

Using IBM Data Virtualization Manager for
z/OS
1.1



Note

Before using this information and the product it supports, read the information in [“Product legal notices” on page 227](#).

This edition applies to Version 1 Release 1 of IBM Data Virtualization Manager for z/OS and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2024-04-01

© **Copyright International Business Machines Corporation 2017, 2022.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© **Rocket Software, Inc. 2017, 2022.**

Contents

- Tables..... vii**

- About this information..... xi**

- How to send your comments to IBM.....xiii**
 - If you have a technical problem..... xiii

- Chapter 1. Virtualizing mainframe data..... 1**
 - Multiple schema support..... 2
 - Prerequisites for adding virtual tables to a schema..... 3
 - Creating schema maps on the server..... 3
 - Moving the existing maps or virtual tables to a schema..... 4
 - Viewing schema maps from the ISPF panel..... 5
 - Using batch JCL jobs..... 5
 - Sample JCL - Batch Data Mapping..... 5
 - Sample JCL - DRDA bind..... 16
 - Sample JCL- Web Services Migration..... 17
 - Using the ISPF application..... 18
 - IBM Data Virtualization Manager for z/OS Interface for ACI.....18
 - Adabas..... 40
 - DB2..... 41
 - IBM Data Virtualization Manager for z/OS Interface for IMS DB: support for DBCTL..... 43
 - VSAM and sequential files.....55
 - Using the Data Mapping Facility..... 60
 - Setting default values for data maps 61
 - Displaying data maps..... 61
 - Viewing individual data elements..... 62
 - Copying data maps..... 64
 - Refreshing data maps..... 65
 - Creating source library maps..... 65
 - Bind or grant DRDA packages.....66

- Chapter 2. Migrating maps..... 67**

- Chapter 3. Using the studio..... 69**
 - Data Virtualization Manager studio overview.....69
 - Perspectives 70
 - DV Data perspective..... 70
 - Services perspective 71
 - Connecting to the Data Virtualization Manager server 72
 - Connecting to the Data Virtualization Manager server.....73
 - Completing the configuration of DRDA access to RDBMS data sources73
 - Locale considerations74
 - Creating server metadata..... 75
 - Creating virtual source libraries 75
 - Creating schemas in the studio.....77
 - Creating virtual tables..... 79
 - Creating virtual views.....105
 - Viewing copybook member name in metadata.....106

Creating Db2 user-defined table functions.....	106
Db2 federation nicknames for distributed environment.....	112
Validating SQL statements.....	114
Generating and executing SQL queries.....	114
Generating code from SQL.....	115
Updating the IMS child segments.....	117
Accessing IT Operational Analytics data.....	118
Accessing SMF data.....	118
Viewing documentation.....	119
Accessing Db2 unload data.....	120
Creating RESTful services.....	120
Setting REST z/OS Connect Web Services preferences.....	121
Connecting to z/OS Connect.....	121
Creating target systems.....	123
Creating Web Services directories.....	124
Creating Web Services and operations.....	125
Web services migration.....	126
Server Trace.....	126
Enabling studio calls in the Server Trace results.....	126
Starting Server Trace.....	127
Filtering Server Trace results.....	127
Using Server Trace Zoom.....	127
Searching Server Trace messages.....	128
Labeling Server Trace messages.....	128
Exporting Server Trace messages.....	129
Importing Server Trace messages.....	130
DV Data preferences.....	130
Data Virtualization Manager preferences.....	130
Admin preferences.....	131
Code generation preferences.....	131
Console preferences.....	132
Dictionary preferences.....	132
Driver preferences.....	132
SQL preferences.....	133
Metadata Discovery preferences.....	134
SSL preferences.....	134
Exporting a virtual table to Software management configuration provider.....	136
Exporting a virtual view to a Software management configuration provider.....	136
Runstats function.....	137
Data Virtualization Manager support for NaturalONE version 9.....	138
About virtualizing large Db2 columns.....	139
Using COBOL copybook map layouts to virtualize large Db2 columns.....	139

Chapter 4. Using JDBC Gateway..... 141

Starting the JDBC Gateway server.....	142
Launching the JDBC Gateway administrative console.....	143
Using the JDBC Gateway administrative console.....	143
Configuring access to data sources using the JDBC Gateway.....	144
Adding JDBC driver information for a data source.....	145
Creating a data source definition entry.....	147
Configuring the Data Virtualization Manager server for JDBC Gateway sources.....	148
Example: Configuring access to Oracle data.....	152
Setting preferences.....	153
Setting JDBC driver preferences.....	154
Setting log preferences.....	154
Setting output preferences.....	155
Troubleshooting.....	155

Chapter 5. SQL DMF supported data types.....	157
Adabas.....	157
COBOL.....	157
IMS - DBD (database description).....	159
Natural conversions.....	159
Natural DDM (data definition module).....	160
SQL Type Support by the IBM Data Virtualization Manager for z/OS interface.....	160
Chapter 6. Supported SQL functions.....	161
ABS.....	162
AVG.....	162
BETWEEN.....	163
BIGINT.....	164
CASE.....	165
CHAR.....	166
CEILING.....	172
COALESCE.....	173
COUNT.....	173
CONCAT.....	174
DATE.....	174
DAY.....	175
DAYOFWEEK.....	176
DAYOFYEAR.....	177
DECIMAL.....	177
DELETE.....	179
DOUBLE.....	179
EXISTS.....	180
FULL OUTER JOIN.....	181
FLOAT.....	181
GROUP-BY.....	182
GROUP_CONCAT.....	183
HAVING.....	184
HEX.....	185
HOUR.....	185
IFNULL.....	186
INNER JOIN.....	186
INSERT.....	187
LARGE INTEGER (INTEGER).....	188
LEFT.....	188
LEFT OUTER JOIN.....	189
LENGTH.....	190
LIKE.....	191
LOWER.....	191
LTRIM.....	192
MAX.....	193
MICROSECOND.....	193
MIN.....	194
MINUTE.....	194
MOD.....	195
MONTH.....	196
ORDER BY.....	196
OUTER JOIN.....	199
RAND.....	200
REAL.....	200
REPLACE.....	201
RIGHT OUTER JOIN.....	203

ROUND.....	203
RTRIM.....	204
SECOND.....	205
SELECT.....	206
SMALLINT.....	208
SUBSTR.....	209
SUM.....	210
SQRT.....	211
TIME.....	212
TIMESTAMP.....	212
TO_CHAR.....	214
TRIM.....	215
UNION.....	216
UNION ALL.....	217
UPDATE.....	218
UPPER.....	222
WHERE.....	222
YEAR.....	223
Accessibility features.....	225
Product legal notices.....	227
Trademarks.....	229
Privacy Policy Considerations.....	230
Terms and conditions.....	230
Index.....	231

Tables

1. To Fingerprint a File.....	6
2. Source to DMF.....	6
3. DBD and PSB Batch Extraction.....	7
4. Source to DMF - Sequential.....	7
5. Source to DMF - To merge Map B into Map A.....	8
6. Source to DMF - To merge a map into a DBD segment.....	8
7. Source to DMF - To remove a map from a DBD segment.....	9
8. Source to DMF - To convert a map to a sequential map.....	9
9. VSAM from Source.....	10
10. To Convert a Map to a VSAM Map.....	10
11. CICS.....	11
12. Adabas - Supported Input Parameters for Extracting an Adabas File.....	12
13. Adabas - Redefine Parameters.....	15
14. Adabas - OVERRIDES Parameters.....	16
15. DRDA bind.....	16
16. Web services migration.....	17
17. Server ACI Facility.....	19
18. Conversions of COBOL data types to ODBC data types.....	34
19. ACI timeout values.....	36
20. FORMAT column types and the SQL equivalent.....	40
21. Server Adabas Data Mapping Facility.....	41
22. Server IMS Data Mapping Facility.....	43
23. Access to file type by interface.....	55

24. Server VSAM/Sequential Data Mapping Facility.....	55
25. Server Data Mapping Facility.....	60
26. Security permissions required to use the migration utility.....	67
27. Data definitions for Adabas.....	157
28. COBOL data definitions used by DMF.....	157
29. PIC S9(_) USAGE COMP-5.....	158
30. PIC 9(_) USAGE COMP-5.....	158
31. Data definitions for IMS - DBD.....	159
32. Data definitions for Natural DDM.....	160
33. VIRTUAL TABLE - EMPL_COMP.....	161
34. VIRTUAL TABLE - EMPLOYEE.....	161
35. Virtual Table - EMP.....	161
36. Virtual Table - JOIN1.....	162
37. Virtual Table - JOIN2.....	162
38. BETWEEN predicate and equivalent search conditions.....	163
39. Between.....	164
40. Case.....	166
41. Exists.....	181
42. Full Outer Join.....	181
43. Group-By.....	183
44. Having.....	185
45. Inner Join.....	187
46. Insert.....	188
47. Left Outer Join.....	190
48. Like.....	191

49. RIGHT OUTER JOIN.....	203
50. SELECT ALL.....	207
51. SELECT - Limited Columns and Row.....	208
52. SELECT - Condition.....	208
53. SUB SELECT	208
54. VIRTUAL TABLE - EMPLOYEE.....	217
55. VIRTUAL TABLE - EMPLOYEE.....	217

About this information

This information supports IBM Data Virtualization Manager for z/OS (5698-DVM) and contains information about using IBM Data Virtualization Manager for z/OS and the IBM Data Virtualization Manager studio, which is a component that is provided with IBM Data Virtualization Manager for z/OS.

Purpose of this information

This document provides an overview of IBM Data Virtualization Manager for z/OS and presents the information you need to access your data sources using the IBM Data Virtualization Manager studio.

Who should read this information

This information is intended for system and database administrators.

How to send your comments to IBM

We appreciate your input on this documentation. Please provide us with any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

Important: If your comment regards a technical problem, see instead [“If you have a technical problem”](#) on page xiii.

Send an email to comments@us.ibm.com.

Include the following information:

- Your name and address
- Your email address
- Your phone or fax number
- The publication title and order number:
 - IBM Data Virtualization Manager for z/OS User Guide
 - SC27-9301-00
- The topic and page number or URL of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM®, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are listed for sending comments. Instead, take one or more of the following actions:

- Visit the [IBM Support Portal \(support.ibm.com\)](https://support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Chapter 1. Virtualizing mainframe data

You can virtualize and access mainframe data using batch processing, the ISPF Server Data Mapping Facility, or the IBM Data Virtualization Manager studio.

- *Batch* is typically used in a production lifecycle for adding and updating maps in your production environment. Batch provides an audit trail for monitoring mainframe changes.
- The *Data Virtualization Manager server ISPF* interface provides interface facilities for accessing data sources and a Data Mapping Facility for creating maps.
- The *IBM Data Virtualization Manager studio* allows you to connect to data sources and map data. In IBM Data Virtualization Manager studio data maps are referred to as virtual tables and virtual collections. For more information, see [“Creating server metadata” on page 75](#).

Virtual tables (maps)

Mapping data means that the source data's definition is used to create a virtual table that matches the definition of the source data. In the IBM Data Virtualization Manager studio data maps are referred to as virtual tables for SQL solutions or virtual collections for NoSQL solutions.

The data definition depends on the programming language that compiles it. For example:

- For COBOL, it is a file definition or data definition.
- For PL/I, it is a Data Control Language (DCL) statement.

The information (length, format, and field elements) is extracted from the data definition and made available to the Data Virtualization Manager server. The data maps refresh process is governed by the **Auto Refresh** parameter that is specified by using the Data Mapping Defaults Options. To access this feature in ISPF, select D Data Mapping from the **Primary Option Menu**. Select 0 Map Defaults from the **Server Data Mapping Facility Menu** and then set the **Auto Refresh** parameter to Yes.

Note: For batch job use the refresh control card in the SYSIN DD

Once created, a data map is called by using a parameter that is passed with an ODBC/JDBC SQL statement. The data map controls the parsing and formatting of the result set, including the names that are assigned to columns. By calling different maps, the Data Mapping Facility (DMF) can return different views or subsets of the data.

Data maps are created by using a series of ISPF panels that allow you to specify a data set containing a compile listing of a program that contains a data definition. The information (length, format, type, and offset, for example) about each field element is extracted from the data definition and made available to Data Virtualization Manager server.

Applications that use Data Virtualization Manager server through a Data Virtualization Manager client, JDBC, and ODBC can use the data maps to manipulate or view the logical or physical data.

Note: The extracts for COBOL and PLI data maps are also available in batch. The AVZMFPAR member is included in the server distributed *hlq.SAVZCNTL* data set as a sample JCL for extracting these types of maps in batch.

When you use the DMF, follow these guidelines:

- Use one server as a test server and another server as a production server.
- Use the DD statement AVZMAPP as part of your initial setup to identify the data sets that contain the maps for your production server.
- For each server, allocate one or more data sets, as needed. To facilitate central control of the production map data set, allocate a “staging” data set for interim maps.

Restrictions for column extraction

The DMF can process up to 7,500 columns for a result set. If more than 7,500 columns are extracted, the extract process continues, but it is recommended that you disable any unwanted columns to reduce the total to 7,500.

Multiple schema support

Create and manage multiple schemas for virtual tables.

With the IBM Data Virtualization Manager for z/OS, you can create and manage multiple schemas. This feature supports the use of multiple virtual schemas for virtual tables in Data Virtualization Manager. You can create schemas in the server using JCL batch files, you can manage the schemas using the JCL batch files and Data Virtualization Manager Studio.

With this feature you can:

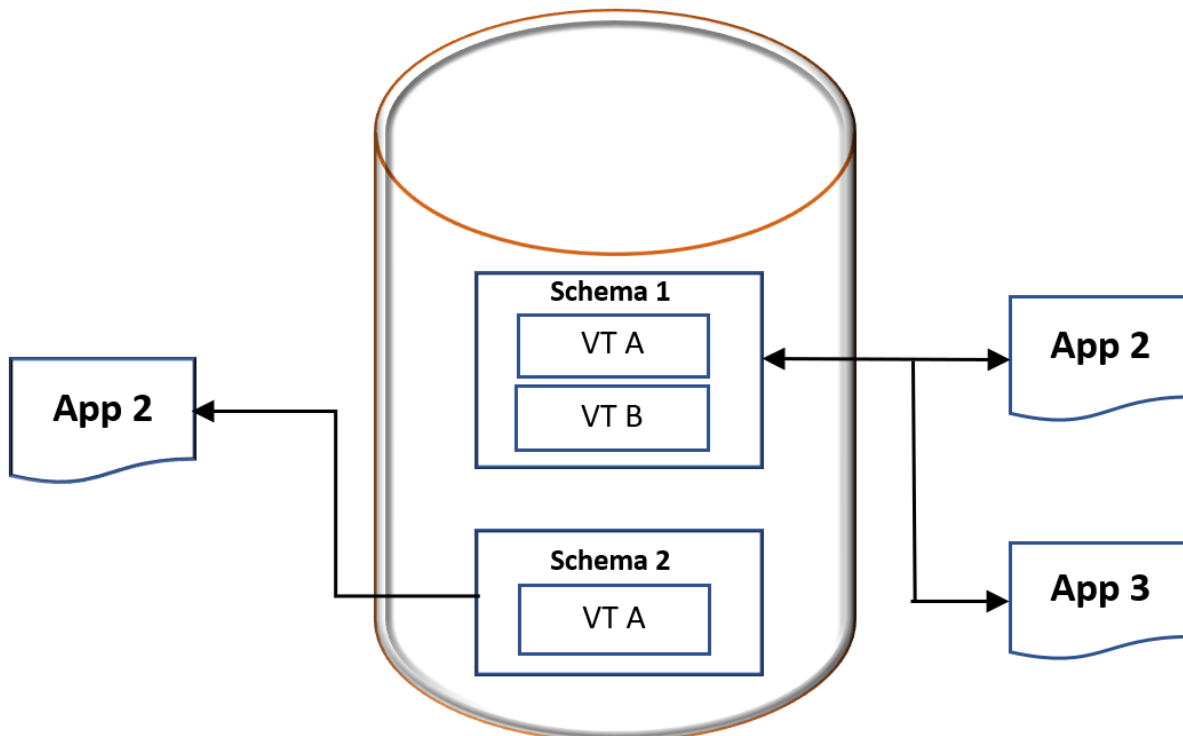
- create multiple virtual tables with the same names in different schemas,
- create tables with same name for different departments or functions in an organization, and
- create virtual tables with the same names that references to different data sources.

Enhanced security

Support for multiple schemas enhances the security by giving you granular control and letting you configure the security at a schema level instead of just at the table level.

You can now additionally configure the security at the schema level and at the schema data set level. You can configure a schema such that only specified users can run queries from the Studio, JDBC, ODBC, DSCClient on the tables that are part of that schema. You can secure the data set of a schema so that only specified users can create virtual tables or have read access to the virtual tables.

The following diagram illustrates an example where two different schemas have virtual tables with similar names and are used by multiple applications:



Related concepts

[“Prerequisites for adding virtual tables to a schema” on page 3](#)

Ensure to follow these prerequisites before adding virtual tables to a schema.

Related tasks

[“Creating schema maps on the server” on page 3](#)

Create new schema maps on the server using the AVZMSCHM member and updating the JCL statements in the control file.

Prerequisites for adding virtual tables to a schema

Ensure to follow these prerequisites before adding virtual tables to a schema.

You can only allocate the virtual tables that you can query using SQL or use with the SQL engine to a schema. You cannot add a virtual table that uses legacy interfaces to a schema. This limitation is because all the legacy interfaces use 31-bit storage, and the schema support requires 64-bit storage.

Here are the prerequisites for creating schemas:

- Every schema must have an associated data set that holds the virtual tables for that schema. The data set is not defined in the started task JCL and is allocated and deallocated dynamically when a server discovers a schema.
- You must associate only one data set per schema.
- You can create a data set by an IEFBR14 step in the batch job or by the studio wizard. If any other process creates a data set, Data Virtualization Manager can use the dataset, but you must ensure that the data set is correct.
- A schema map has a schema name and a map name. The schema name must not exceed 64 bytes. The map name defaults to the first 50 bytes of the schema name if it is not specified. If this default map name is not unique, you might come across an error, and you must provide a map name.
- Currently, you can create only 200 schemas and virtual directories per server.

Related concepts

[“Multiple schema support” on page 2](#)

Create and manage multiple schemas for virtual tables.

Creating schema maps on the server

Create new schema maps on the server using the AVZMSCHM member and updating the JCL statements in the control file.

About this task

A new member called AVZMSCHM is added to the JCL statement in the control file. The job contains instructions of the required modifications and information about the differences in the statement.

Update the JCL with the required information using the SYSIN statement.

Procedure

1. Modify the JCL according to the instructions provided in the h1q.CNTL (AVZMSCHM) member. Update the SYSIN statements as follows:

```
SYSIN DD*
SCHEMA = NAME_OF_YOUR_SCHEMA
SCHEMA DATASET = MAP_DATA_SET_FOR_MAPS_IN_THIS_SCHEMA SCHEMA MAP
SAVE OPTION = SAVE
REFRESH OPTION = REFRESH
```

Option	Description
SCHEMA	Name of your schema that you can use in the SQL query.
SCHEMA DATASET	Data set name for the schema.
SCHEMA MAP	Map name for the schema.

2. Change the existing migration jobs by updating the h1q.CNTL (AVZGNMPM) member. Add a new SYSIN statement as follows:

```
OPT VT SCHEMA=
VT SCHEMA=OLD_SCHEMA,NEW_SCHEMA
VT SCHEMA=OLD_SCHEMA_1,NEW_SCHEMA_1
```

Option	Description
VT SCHEMA	The schema name in which the virtual table must reside. The format is old schema, new schema. Add only one change pair per line.

To move a map that does not have a schema to a schema, use the ADD option as follows:

```
VTSHEMA=ADD,NEW_SCHEMA
```

Note: Only one add statement is allowed per migration.

Related concepts

[“Multiple schema support” on page 2](#)

Create and manage multiple schemas for virtual tables.

Related tasks

[“Moving the existing maps or virtual tables to a schema” on page 4](#)

Move the existing maps or virtual tables to a specific schema using the h1q.CNTL (AVZGNMPM) member. You can also use the standalone utility using the h1q.CNTL (AVZDMUT) member to move existing maps to a schema.

[“Viewing schema maps from the ISPF panel” on page 5](#)

In the ISPF panel, view all the schema maps by navigating from **Data Mapping** to **Map Display**.

Moving the existing maps or virtual tables to a schema

Move the existing maps or virtual tables to a specific schema using the h1q.CNTL (AVZGNMPM) member. You can also use the standalone utility using the h1q.CNTL (AVZDMUT) member to move existing maps to a schema.

About this task

Move the existing maps or virtual tables to specific schemas using either the SYSIN statement in the h1q.CNTL (AVZGNMPM) member or the standalone utility in the h1q.CNTL (AVZDMUT) member.

Procedure

1. Moving the maps using the SYSIN statement in the h1q.CNTL (AVZGNMPM) member. Use the SYSIN statements as follows:

```
SYSIN DD *
VT SCHEMA = ADD,NEW_SCHEMA
MAP=NAMES_OF_MAPS_TO_PUT_IN_THIS_SCHEMA
```

Note: Using the migration job with the same SOURCE and TARGET SSID, the virtual table is copied to the specified schema and the original map is also available in the default schema.

2. Moving the maps using the standalone utility in the `hlq.CNTL(AVZDMUT)` member: You can run the standalone batch utility outside Data Virtualization Manager. Take your current map library and specify the schema map library that you want to use. Ensure that schema map library name matches with schema name that you defined when you created the schema. Anything that you have not defined to move to the schema map library moves to the new map library. If you want to revert the changes, you can restart the server and point to the current map library and redo the steps.

Use the instructions that are described in the job as follows:

```
//STEP1      EXEC PGM=SDBDMUT,REGION=0M
//STEPLIB    DD DISP=SHR,DSN=&LOADLIB
//SYSEXEC    DD DISP=SHR,DSN=&REXXLIB
//DVSMAPP    DD DISP=SHR,DSN=&CURRENT_MAP_LIBRARY
//SCHEMAPP   DD DISP=SHR,DSN=&SCHEMA_MAP_LIBRARY
//NEWMAPP    DD DISP=SHR,DSN=&NEW_MAP_LIBRARY
//SYSPRINT   DD SYSOUT=*
//SYSIN DD*
ADDSHEMA
SCHEMA_NAME MAPNAME
MAPNAME
/*
```

Viewing schema maps from the ISPF panel

In the ISPF panel, view all the schema maps by navigating from **Data Mapping** to **Map Display**.

About this task

View all the schema maps by navigating from **Data Mapping** to **Map Display**. A list of schema maps is displayed. DVSQLE is the default schema. The existing maps are part of the DVSQLE schema. You can open a schema and display the list of maps that are part of the schema.

Note: On the ISPF panel, the structure name field displays the map name.

Using batch JCL jobs

Batch processing is typically used in a production lifecycle for adding and updating maps in your production environment. Batch JCL jobs provide an audit trail for monitoring mainframe changes.

There are different ways to use batch processing:

- You can create a virtual table in the IBM Data Virtualization Manager studio and use a batch job to copy the map into your production environment. For more information on creating virtual tables using IBM Data Virtualization Manager studio, see [“Creating virtual tables” on page 79](#).
- You can use the batch job to create the new virtual table that is then put directly into your production environment.

Note: Using the IBM Data Virtualization Manager studio is the recommended method to create virtual tables.

Sample JCL - Batch Data Mapping

You can use the AVZMFPAR member to extract batch maps for COBOL, PLI, Natural, Sequential Files, DBD, PSBs, ADABAS, CICS, and VSAM data sources, as well as for MFS maps and stored procedures

. You must use a compiled listing to perform the extract.

Note: You must perform a mapping refresh before it shows in the display map command.

Tip: Use the sample batch job in member AVZMFPAR located in your `hlq.SAVZCNTL` data set for extracting these maps in batch.

Table 1 on page 6 through Table 14 on page 16 describe the parameters that can be used in the AVZMFPAR member.

<i>Table 1. To Fingerprint a File</i>		
Required?	Parameter	Description
Required	SSID = AVZS	The target subsystem to use this map.
Required	FUNCTION = FRPT	The function to be performed by the DMF parser. Fingerprint (FRPT). When a file is fingerprinted, the file is scanned to attempt to determine the language type, such as COBOL.
Optional	SOURCE =	The name of the data set that contains the source to parse. Note: It is recommend that, instead of using this parameter, you use the //SOURCE DD statement, which overrides this parameter.

<i>Table 2. Source to DMF</i>		
Required?	Parameter	Description
Required	SSID = AVZS	The target IBM Data Virtualization Manager for z/OS subsystem to use this map.
Required	FUNCTION = STOD	The function to be performed by the DMF parser.
Optional	SOURCE = HLQ.SOURCE.FILE	The name of the data set that contains the source to parse. Note: It is recommended that, instead of using this parameter, you use the //SOURCE DD statement, which overrides this parameter.
Required	START FIELD =	The name of the first field to map.
Optional	END FIELD =	The name of the last field to map.
Optional	OFFSET ZERO = Y/N	Specifies whether to set the Start Search Field offset to zero, even if it is not a group level or the first definition in a group. Defaults to YES.
Optional	SAVE OPTION =	Specifies The DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH

Required?	Parameter	Description
Required	SSID = AVZS	The target IBM Data Virtualization Manager for z/OS subsystem to use this map.
Required	FUNCTION = STOD	The function to be performed by the DMF parser.
Optional	SOURCE = HLQ.SOURCE.FILE	The name of the data set that contains the source to parse. Note: It is recommended that, instead of using this parameter, you use the //SOURCE DD statement, which overrides this parameter.
Optional	END FIELD =	The name of the last field to map.
Optional	OFFSET ZERO = Y/N	Specifies whether to set the Start Search Field offset to zero, even if it is not a group level or the first definition in a group. Defaults to YES.
Optional	V2T	Converts VAR fields to TRUE VAR fields if this option is set. The TRUE VAR fields have a 2 byte length of data field preceding the data.
Optional	FLATTEN	Setting this option to YES will flatten the arrays at run time. Setting this option to NO will instantiate the arrays as separate tables at run time.
Optional	MAP DATASET	The dataset name where the map will be stored.
Optional	MAP NAME	The structure name. The maximum length of the map name is 30 bytes.
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH

Required?	Parameter	Description
Required	SEQ FILE =	Specifies the data set to associate with the map (implies a sequential map for use by the sequential interface).
Optional	SEQ DSN COLUMN NAME =	The sequential request data set column name, if data set name (for PDS(E) data sets), that can be viewed by the client.
Optional	SEQ MEMBER COLUMN NAME =	The sequential request member column name, if member name (for PDS(E) data sets) that can be viewed by the client.

<i>Table 4. Source to DMF - Sequential (continued)</i>		
Required?	Parameter	Description
Optional	SEQ COLUMN NAME SEARCHABLE =	The sequential request DSN and member column names that can be used on the WHERE clause of a SQL statement.

<i>Table 5. Source to DMF - To merge Map B into Map A</i>		
Required?	Parameter	Description
Required	SSID = AVZS	The server subsystem that uses this map.
Required	FUNCTION = MMER	The function to be performed by the DMF parser.
Required	MERGE A =	The map that contains the merged information (Map A of a merge function).
Required	MERGE B =	The name of the map that is merged into Map A.
Optional	NEW MAP NAME =	The name of the new map (structure name). The maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.
Optional	SAVE OPTION =	Specifies The DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH

<i>Table 6. Source to DMF - To merge a map into a DBD segment</i>		
Required?	Parameter	Description
Required	SSID = AVZS	The target server subsystem to use this map.
Required	FUNCTION = MDBD	The function to be performed by the DMF parser.
Required	DBDNAME =	The name of the DBD to link to or unlink from.
Required	SEGMENT =	The name of the segment in the DBDNAME to link to or unlink from.
Required	LINK MAP =	The name of the map to link to the segment.

Table 6. Source to DMF - To merge a map into a DBD segment (continued)

Required?	Parameter	Description
Optional	SAVE OPTION =	Specifies The DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH
Optional	DISABLE DUP = Y/N	Indicates whether to disable duplicates in the DBD.
Optional	DISABLE FILLER = Y/N	Indicates whether to disable filler fields in the DBD.

Table 7. Source to DMF - To remove a map from a DBD segment

Required?	Parameter	Description
Required	SSID = AVZS	The target server subsystem to use this map.
Required	FUNCTION = MDBD	The function to be performed by the DMF parser.
Required	DBDNAME =	The name of the DBD to link to or unlink from.
Required	SEGMENT =	The name of the segment in the DBDNAME to link to or unlink from.
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH

Table 8. Source to DMF - To convert a map to a sequential map

Required?	Parameter	Description
Required	SSID = AVZS	The target subsystem to use this map.
Required	FUNCTION = MTOS	The function that the parser performs.
Required	INPUT MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.

<i>Table 8. Source to DMF - To convert a map to a sequential map (continued)</i>		
Required?	Parameter	Description
Required	SEQ FILE =	The data set associated with the map (implies a sequential map for use by the sequential interface).
Optional	SEQ DSN COLUMN NAME =	The sequential request data set column name, if data set name (for PDS(E) data sets), that can be viewed by the client.
Optional	SEQ MEMBER COLUMN NAME =	The sequential request member column name, if member name (for PDS(E) data sets) that can be viewed by the client.
Optional	SEQ COLUMN NAME SEARCHABLE =	The sequential request DSN and member column names that can be used on the WHERE clause of a SQL statement.
Optional	NEW MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH

<i>Table 9. VSAM from Source</i>		
Required?	Parameter	Description
Required	VSAM FILE = HLQ.VSAM.FILE	The VSAM file to be associated with this map (implies a VSAM map for use by the VSAM or CICS VSAM interface).
Optional	ALT INDEX = Y/N	Indicates that you want to use alternate indexes to access this VSAM map. Default is NO.
Optional	NEW MAP NAME =	The name of this map, which is known as the structure name. Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.

<i>Table 10. To Convert a Map to a VSAM Map</i>		
Required?	Parameter	Description
Required	SSID = AVZS	The target server subsystem to use this map.
Required	FUNCTION = MTOV	The function to be performed by the DMF parser.

<i>Table 10. To Convert a Map to a VSAM Map (continued)</i>		
Required?	Parameter	Description
Required	INPUT MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.
Required	VSAM FILE = HLQ.VSAM.FILE	The VSAM file to be associated with this map (implies a VSAM map for use by the VSAM or CICS VSAM interface).
Optional	ALT INDEX = Y/N	Indicates whether to use alternate indexes to access this VSAM map. Default is NO.
Optional	NEW MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.

<i>Table 11. CICS</i>		
Required?	Parameter	Description
Required	CICS CONN =	The name of the CICS connection to use for this map, if this map is to be used by the CICS VSAM interface.
Required	CICS TRAN =	The name of the CICS transaction to use for this map, if this map is to be used by the CICS VSAM interface.
Required	CICS FCT = or CICS FCT ENTRY =	The name of the CICS FCT entry to use for this map, if this map is to be used by the CICS VSAM interface.
Optional	AIX n FCT = where n is numeric for 1-8.	The name of the CICS FCT entry to use for each IX path found, if this name is to be used by the CICS VSAM interface.
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH



Attention:

- A COBOL listing with OPT(FULL) cannot be processed to produce a virtual table. Keywords for this process define the same elements that you would specify on the ISPF panels.

Creating ADABAS virtual table:

When executing the AVZMBTPA utility without specifying a DDM on the SOURCE DD statement, the batch utility performs an Adabas LF command to read the FDT and generates 2 byte column names. It is recommended to supply a DDM view to provide long names, date, timestamp, scale formatting.

Note: Some 2 byte column names are reserved for SQL, e.g. (AS, IF, IS, IN, ON, OR, TO, TS)

Example:

```
//DMFEXTR1 EXEC PGM=IKJEFT01,PARM=(AVZMBTPA 0'),REGION=0M
//STEPLIB DD DISP=SHR,DSN=your.SAVZLOAD
// DD DISP=SGR,DSN=your.ADABAS.LOAD <= ADALNKR routine
//SYSEXEC DD DISP=SHR,DSN=your.SAVZEXEC
//SOURCE DD DISP=SHR,DSN=SOURCE.DATASET.AND.MEMBER <= DDM LISTING
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//SYSIN DD *
SSID = AVZS
FUNCTION = ADLF
MAP NAME = ADA100_FILE999_SALES
SAVE OPTION = REPLACE
REFRESH OPTION = REFRESH
ADABAS DBID = 100
ADABAS DBNAME = BATCHPARSER
ADABAS FILE NUM = 999
ADABAS SUBSYS = ADAB
FLATTEN = YES
MU COUNT = 3
PE COUNT = 3
CREATE COUNT FIELDS = YES
USE DDM = YES
REDEFINE_BEGIN
REDEFINE_FILE = 999 REDEFINE_FIELD = AA
  (REDEFINE_COLUMN = AA_COLUMN1 REDEFINE_FORMAT = A
  REDEFINE_LENGTH = 10 REDEFINE_OFFSET = 0)
  (REDEFINE_COLUMN = AA_COLUMN2 REDEFINE_FORMAT = A
  REDEFINE_LENGTH = 20 REDEFINE_OFFSET = 10)
REDEFINE_END
BEGIN_OVERRIDES
  FILE = 999, FIELD = P2, FORMAT = P, SCALE = 2
  FILE = 999, FIELD = P8, FORMAT = P, SCALE = 2
  FILE = 999, FIELD = F2, FORMAT = A, LENGTH = 6
  FILE = 999, FIELD = U8, FORMAT = I, LENGTH = 4
  FILE = 999, FIELD = P4, FORMAT = D
  FILE = 999, FIELD = P7, FORMAT = S
END_OVERRIDES
/*
```

Table 12. Adabas - Supported Input Parameters for Extracting an Adabas File		
Required?	Parameter	Description
Required	SSID = AVZS	The Data Virtualization Manager subsystem ID.
Required	FUNCTION = ADLF	The parser function for Adabas.
Required	MAP NAME =	The name of this map, which is known as the structure name. The maximum length is 30 bytes.
Required	ADABAS DBID =	The database ID as shown on the ADAREP.
Optional	ADABAS DBNAME =	The database name as shown on the ADAREP. This name is used for reporting purposes.
Required	ADABAS FILE NUM =	The file number of the Adabas file as shown on the ADAREP.
Required	ADABAS SUBSYS =	The Adabas SVC router name assignment. If not specified, the default is ADAB.
Required	MU COUNT =	The maximum allowed MU (multiple value field) columns generated. Optional when using FLATTEN = N. If not specified, the default is 0.
Required	PE COUNT =	The maximum allowed PE (periodic groups) columns generated. Optional when using FLATTEN = N. If not specified, the default is 0.

Table 12. Adabas - Supported Input Parameters for Extracting an Adabas File (continued)

Required?	Parameter	Description
Optional	CREATE COUNT FIELDS =	<p>If set, the parser generates a count field for all MU and PE fields. The name that is generated for the field is the PE or MU field name plus the letters "_C" (if using the field name in the DDM) or the PE or MU field name plus the letter "C" (if using the field name from the LF command). For example, if you run AVZMBTPA by using the DDM, and the PE or MU name ACCOUNTS, the generated name for the count field is ACCOUNTS_C. If you run AVZMBTPA by using only the LF command, and the PE or MU name is AA, the generated name for the count field would be AAC.</p> <p>Values are:</p> <ul style="list-style-type: none"> • Y for Yes • N for No
Optional	U_2_P =	<p>Indicates whether the extract converts all unpacked format fields to the packed format.</p> <p>Values are:</p> <ul style="list-style-type: none"> • Y for Yes • N for No <p>The default is N.</p> <p>Note: Use this parameter if you anticipate negative Adabas unpacked decimal numbers; otherwise, an alphanumeric representation is returned. For example, -23 would be returned as 02L. Use of this parameter changes the data type from character to numeric.</p>
Optional	B_2_I =	<p>Indicates whether the extract converts all 2-byte, 4-byte, and 8-byte binary file fields to short integer, integer, and big integer formats respectively.</p> <p>Values are:</p> <ul style="list-style-type: none"> • Y for Yes • N for No <p>The default is N.</p>
Optional	DE SEARCH ONLY =	<p>Generates control definitions that allow the client to use WHERE columns that are Adabas descriptors (such as SUPERDE, SUBDE, and HYPERDE).</p> <p>Values are:</p> <ul style="list-style-type: none"> • Y for Yes • N for No <p>The default is N.</p>

Table 12. Adabas - Supported Input Parameters for Extracting an Adabas File (continued)

Required?	Parameter	Description
Optional	SEARCH BY PE INDEX =	<p>Allows the client to target rows that match a particular occurrence of the PE field when searching rows by using the WHERE clause. If not specified, all rows where any occurrence of that PE field matches the value specified are targeted.</p> <p>Values are:</p> <ul style="list-style-type: none"> • Y for Yes • N for No <p>The default is N.</p>
Optional	USE DDM =	<p>Uses the DDM source supplied on the source DD statement to update the Adabas map with long field names and to override the data types as defined in the Adabas FDT. The DDM must be extracted using step DDMEXTR in this JCL.</p> <p>To generate a DDM member, execute a Natural batch job. For example:</p> <pre>//CMPRINT DD DISP=SHR,DSN=h1q.DDMS(DDMEMBER) //CMWKF01 DD DUMMY //CMSYNIN DD * LOGON LIB LIST VIEW ADABAS-DDM FIN</pre> <p>Valid values are:</p> <ul style="list-style-type: none"> • Y for Yes • N for No <p>The default is N.</p>
Optional	SAVE OPTION =	<p>Specifies the DMF import save option. Valid values are:</p> <ul style="list-style-type: none"> • NOSAVE • SAVE • REPLACE <p>It is recommended that you use the SAVE value to prevent overwriting another map. The default is SAVE.</p>
Optional	REFRESH OPTION =	<p>Specifies whether to refresh the map. Valid values are:</p> <ul style="list-style-type: none"> • NOREFRESH • REFRESH <p>The default is NOREFRESH.</p>

Table 12. Adabas - Supported Input Parameters for Extracting an Adabas File (continued)

Required?	Parameter	Description
Optional	SECURITY =	Generates security on the "TABLE DEFINITION". Values are: <ul style="list-style-type: none"> • Y for Yes • N for No The default is N. To define a Data Virtualization Manager Resources for Adabas file security, you must edit and submit one of the following sample jobs (depending on your security type) located in the <i>hlq.SAVZCNTL</i> library: <ul style="list-style-type: none"> • AVZRAVDA for RACF security • AVZAZVDA for CA ACF2 security • AVZTSVDA for CA Top Secret Security
Optional	ADASCRPWD =	The password that is used to access the specified file number. If the IBM Data Virtualization Manager for z/OS Interface for Adabas accepts the password, it passes it to the Adabas control block ADDS 3 field and generates the ADASCRPWD =password statement.
Optional	DBCS =	Specifies which Adabas Alpha/Binary to use to store pure DBCS data without SO/SI characters.

Table 13. Adabas - Redefine Parameters

Required?	Parameter	Description
Optional	REDEFINE_FORMAT = x	The 1-byte format type to be redefined. The rules for redefining a field format must conform to the rules of data type conversions that Adabas permits; otherwise, an Adabas response code might be generated because of a conversion mismatch.
Optional	REDEFINE_PRECISION= nnn	The precision override.
Optional	REDEFINE_COLUMN = xxxxxx...	The 30-character name for the new redefined field that replaces the elements that comprise the original field. For example, if you are redefining field AA as two new fields or columns, the REDEFINE_COLUMN would indicate the new names for the two new fields: AA_PART_1 and AA_PART_2.
Optional	REDEFINE_OFFSET = nnn	The offset of the new redefined field, where <i>nnn</i> is the redefined offset to use.
Optional	REDEFINE_AS_COUNT	This option is used to support the SELECT COUNT(*) statement when there is no unique descriptor (DE, UQ) or fixed-format descriptor (DE, FI).
Optional	SET_AS_PRIMARYKEY	Allows you to set the field that is used as the primary key when there is no unique descriptor (DE, UQ).

Required?	Parameter	Description
Optional	FILE	Adabas file number.
Optional	FIELD	Adabas 2 byte field name.
Optional	FORMAT	Adabas overriding format. The possible conversions are: <ul style="list-style-type: none"> Unpacked into Packed, Unpacked into Integer, Fixed into Alpha Packed into Date, Packed into S (Timestamp), Binary into Integer, Binary into Alpha
Optional	LENGTH	Field length in bytes.
Optional	SCALE	Scale specified for Packed and Unpacked fields.

Sample JCL - DRDA bind

You can use AVZCLBND member to bind or grant access to DRDA packages.

The Table 15 on page 16 table describes the parameters in the sample JCL file.

Required?	Parameter	Description
Required	SSID = AVZS	The target subsystem to use this map.
Required	DB2!=	Db2 for z/OS or Db2 LUW subsystem ID.
Required	FUNCTION=	<ul style="list-style-type: none"> BIND - bind package is created. GRANT - package permission is given to the ID given in the GRANTID field. BIND - GRANT or BOTH - performs both bind and grant operations. By default, BOTH is chosen.
OPTIONAL	REPLACE=	Specifies if the existing packages need to be replaced. Valid values are N and Y. By default, N (no replacement) is chosen.
OPTIONAL	SECTIONS=	Specifies the number of cursors to use while processing results. The default value is 200.
OPTIONAL	COLLECTION	Specifies the collection ID to be assigned to the DB2 packages. The default value is NULLID.

<i>Table 15. DRDA bind (continued)</i>		
Required?	Parameter	Description
OPTIONAL	GRANTID	Specifies the ID for which the package permissions must be granted. The default value is PUBLIC.

Sample JCL- Web Services Migration

You can use AVZSCMG1 member to migrate web services.

The [Table 16 on page 17](#) table describes the parameters in the sample JCL file.

<i>Table 16. Web services migration</i>		
Required?	Parameter	Description
Required	SOURCE SERVER SSID =	Specifies the subsystem ID where the web services are currently located.
Required	TARGET SERVER SSID =	Specifies the subsystem ID where the web services will be migrated to.
Required	TARGET LOADLIB=	Specifies the load library of the target subsystem.
Required	TARGET EXECFB=	Specifies the EXECFB library of the target subsystem.
Required	SOURCE V/DIR=	Specifies the directory where the web services reside.
Required	TARGET MICROFLOW DSN=	Specifies the data set name of the MFL data set on the target system.
Required	TARGET METADATA DSN=	Specifies the data set name of the MMAP data set on the target system.
Required	METADATA EXPORT PDS =	Specifies the pds library where the exported metadata objects will be unloaded to.
Required	MICROFLOW UNLOAD DSN =	Specifies the file where the exported web service microflow objects will be unloaded to.
Required	JCL MEMBER NAME=	Specifies the pds member name in the file allocated to ddname ispfle for the jcl generated.
Optional	WEB SERVICE=	Specifies the web service to migrate.
Optional	TARGET SYSTEM=	Specifies the new name of the target subsystem.

Table 16. Web services migration (continued)

Required?	Parameter	Description
Optional	RESTRICTIP =	Specifies the address of a restricted IP address that must be changed.
Optional	SCHEMA=	Specifies the schema name that must be changed.
Optional	TARGET V/DIR=	Specifies the directory where the web service must be placed.
Optional	TARGET RULESET	Specifies the ruleset name used in the target directory.
Optional	TARGET V/DIR URL =	Specifies the url name used in the target directory. This field is required if target V/DIR is mentioned.
Optional	TARGET NAMESPACE=	Specifies a new host and a port number to be used for the target namespace.
Optional	SAVE OPTION =	Specifies The DMF import save option. Valid values are: <ul style="list-style-type: none"> • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: <ul style="list-style-type: none"> • NOREFRESH (default) • REFRESH

Using the ISPF application

IBM Data Virtualization Manager for z/OS supports access to many data sources.

IBM Data Virtualization Manager for z/OS Interface for ACI

The Advanced Communication Interface (ACI) enables applications that are written in COBOL, Assembler, PL/I, or Natural and running in remote transaction processing (TP) environments to communicate with the desktop.

ACI allows developers to create applications that can run services in their transaction processing (TP) environments. In this case, the IBM Data Virtualization Manager for z/OS Interface for ACI provides access to transactions in a CICS or Batch environment using the ACI API.

This interface also provides data access to JDBC and ODBC clients, web browser clients, and *n*-tier applications. It allows any JDBC- or ODBC-enabled application to use standard JDBC or ODBC facilities to make requests directly to a COBOL, Assembler, PL/I, or Natural program. A relational result set is returned to the application running in its native transaction processing environment.

The ACI Interface Facilities option on the Data Virtualization Manager server - Primary Option Menu provides access to the Server ACI Facility features.

Table 17. Server ACI Facility

Option	Description
ACI Server Definition	Create ACI server map information
Natural Extract	Extract from Natural source
COBOL Extract	Extract from a COBOL listing
PL/I Extract	Extract from a PL/I listing
ACI Map Display	Display ACI server map information
Map Display	Display all map information
Map Copy	Copy maps
Map Refresh	Refresh maps
Active Server Display	Display active ACI servers
ACI Error Create	Create ACI error processing definitions
ACI Error Display	Display ACI error processing definitions
ACI Execution Errors	Display ACI execution errors
CICS Global ACI Count	Monitor CICS global ACI counters
ACI Buffer Pools	Display ACI buffer pool information

ACI server map information

Before you can use the CALL DVS_ACI request for data, you must first define a map to the server by using the ACI Server Definition option in the ACI Facility.

Using this option, you can define an ACI server map. Users can create ACI server maps (service definitions) for the following types of servers:

- CICS servers
- Batch servers
- Stored procedures

A stored procedure is a started task that runs as a stored procedure for ACI.

The map defines and stores the definition of a remote service application. The definitions are retrieved when referenced in the second parameter of the CALL DVS_ACI request.

Defining an ACI server map

ACI server maps can be created in either of the following ways:

- Creating an ACI server map in batch
- Creating an ACI server map by using the **Server ACI Facility** panel

Creating an ACI server map in batch

Use the AVZACMP1 member (located in the *hlq.SAVZCNTL* data set) for sample JCL that you can use to create ACI server maps in batch.

Creating an ACI server map using the Server ACI Facility panel

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. From the Server ACI Facility panel, select **ACI Server Definition** and press Enter.

The following sections guide you through creating an ACI CICS server definition and an ACI batch server definition.

Creating an ACI server definition for CICS

Create an ACI server definition for CICS.

About this task

Use the following procedure to create the ACI server definition for CICS using the Server ACI Facility.

Note: You can use the AVZACMP2 member (located in the *hlq.SAVZCNTL* data set) for sample JCL that you can use to create ACI CICS server data maps in batch.

Procedure

1. From the **Server ACI Extract** menu, select **Create ACI CICS Server Definition** and press Enter.
2. Complete the following fields:

Note: The (R) or (O) at the end of each field indicates whether the field is Required or Optional.

- Server Name: This value must correspond to the service information defined in the service application.
- Server Service Class: This value must correspond to the service information defined in the service application.
- Server Service: This value must correspond to the service information defined in the service application.

Note:

- The combination of the server name, server service class, and server service identifies a service.
- If part of the name is changed while a service is active, all ACI services that are associated with the former name are treated as orphan services because an ACI service with that name no longer exists in the system. The ACI services that are associated with the former name still appear in the active ACI server maps and continue to display until they time out or until they are manually terminated.
- Persistent Connection: Y (Yes) allows persistent connections, and N (No) allows non-persistent connections. The following differences distinguish persistent connections from non-persistent connections:
 - A persistent connection allows ongoing conversational requests and responses. The server is "assigned" to the client until the client issues an end-of-conversation (EOC) request, at which point, the ACI server program deregisters the service and terminates. When another connection requests the same ACI service, a new ACI server is started. This implies that the client and service are in conversation mode.

It is also possible for a persistent ACI server to be reused by different client connections after the EOC request by deregistering and registering.

Note: Reusing persistent connections improves performance and reduces overhead. For information about how to create a program to reuse persistent connections, see [“Reusing persistent connections”](#).

- A non-persistent connection is one in which a single request is issued and a single response is received. The server is available for use by any client on a receive request. The service can be used by any incoming client connection with the Data Virtualization Manager Server.
- Secure this Service: Y (Yes), restricts the connection to the user who has a SAF resource for the ACI service. Only a user ID with a valid resource defined for the ACI service is allowed to start and connect to that service during its life. This field defaults to N (No), which allows any user ID to start and connect to that started ACI service. The format of the resource name is:

```
ACI.aci-mapname
```

Note: To secure a persistent ACI connection, you must edit and submit one of the following sample jobs (depending on your security type) located in your *hlq.SAVZCNTL* library to specify the map name to be used.

- AVZRAACI for RACF security
- AVZA2ACI for CA ACF2 security
- AVZTSACI for CA Top Security security
- Mirror Transaction: The name of the CICS transaction that corresponds to the EXCI mirror program DFHMIRS.
- Connection Name: The name of the CICS connection, as defined in CICS and the Data Virtualization Manager configuration member IN00, for DPL requests in CICS.
- Transaction Name: The user transaction to start in the CICS region.

Note: The Mirror Transaction, Connection Name, and Transaction Name are configured by the user so the transaction can be invoked under CICS. They are previously defined in the Data Virtualization Manager configuration member.

- Unit of Work Participant (for persistent connections only): Indicates whether this transaction can process units of work. If so, the transaction must also support a persistent connection (the **Persistent Connection** field must be set to Y (Yes).
- Maximum UOW Buffer Size (for UOW participants only): The maximum buffer size that UOW transactions can accommodate for any single call (the maximum size of data that the ACI interface can send to the ACI service at any one time). The buffer size is rounded to 1000-byte increments and the maximum is 32,000. If 32,000 is specified, the ACI interface reduces that number to 31,767 bytes at execution time to comply with the maximum size the ACI service can receive at any one time.

Note: The client can send any size data, including SQL_LONGVARCHAR and SQL_LONGVARBINARY data types, which can be greater than 31,767 bytes long. The data that is received from the client is buffered on the Data Virtualization Manager Server until a UOWLAST or UOWONLY request is received [“Query syntax”](#), at which time it is sent to the ACI service in size increments that do not exceed the maximum UOW buffer size value.

- Max Execution Time: The maximum time, in seconds, that an ACI service can run before the ACI service is set to TIMEOUT status, at which point, the client is released and receives notification that the ACI service timed out. If you do not set this value, the client non-activity timer value is used.

For more information about the timeout values, see [“Timeout values”](#).

- Secure Server to Userid: This value defaults to N (No), which allows any user ID to reconnect to that started ACI service. To restrict the connection to the user who started the ACI service, set this value to Y (Yes). Then only the user ID that started the ACI service is allowed to reconnect to the service during its life.
- Auto Start: service Indicates to the Data Virtualization Manager server that it can start this service.

- Max. No. Allowed: The maximum number of concurrent servers of this definition type that can run at any given time.

Note: In cases where CICS is slow or has performance problems, a client request can be submitted to multiple ACI services. To prevent a client request from being submitted to all ACI services, you can limit the number of ACI services that the client request can be submitted to [“Using submission limit checking”](#). When the registration is requested by a CICS program running online (not started by the Data Virtualization Manager server), the Max Allowed setting does not take effect. See [“Running a CICS program not started by Data Virtualization Manager server”](#)
 - Auto Terminate (for non-persistent connections only): A number 0 - 99999 to indicate the number of receives to accept before the system automatically terminates the server and the service deregisters itself. If this field is blank, the default value is 0 (zero). Complete this field only if you specified N (No) for the **Persistent Connection** field.

Note: This field limits the number of times that the server can receive requests after which it is terminated to protect storage resources.
 - Client Non-Activity Timer: The non-activity timer, which has the following functions:
 - It is the amount of time that the client waits for the service to return before it times out.
 - If the max execution time value is not set, it is used for the time that an ACI service can run before timing out and releasing the client.
 - If the maximum wait for server timer value is not set, it is used for the time that a client can wait for an ACI service to be assigned before timing out.
 - For persistent services, it is also the amount of time that the service remains idle waiting for a client to converse with the service (the amount of time that is allowed for a client to interface with a service).

Note: For persistent services only, if the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to SERVER, the Server shutdown non-activity timer value is used for all of the functions that are listed in the client non-activity timer description.

However, the ACIPERSISTTIMEOUT would not yet apply for servers that are still in a pending registration state. Prior to registration, the Data Virtualization Manager uses the max wait for server. If that is not set, the Data Virtualization Manager uses the client non-activity timer. If the timer is not set, the DV uses a default of 15 seconds. For more information about the timeout values, see [“Timeout values”](#).
 - Server Shutdown Non-Activity Timer (for non-persistent connections only): The amount of time the service can be non-active before the Data Virtualization Manager Server requests it to terminate. This field allows the less frequently used servers to stop, freeing up storage for the more frequently used servers, improving the use of available resources.

Note: For persistent services, by default, the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to CLIENT, so this field is not used, and the client non-activity timer value is used. For more information about the timeout values, see [“Timeout values”](#).
 - Maximum Wait for Server Timer: The maximum time that a client can wait for an ACI service to be assigned before the request is timed out and the client is released. If this value is not set, the client non-activity timer value is used. For more information about the timeout values, see [“Timeout values”](#).
 - SDCIFEN Information: If using SDCIFEN as the program associated with the CICS transaction defined in the **Transaction Name** field to pass data to the transaction, enter the SDCIFEN information. The following information is required:
 - The name of the program to which SDCIFEN transfers control.
 - The items to be passed to the transaction using the COMMAREA.
3. Press Enter to complete the ACI server definition. If it was successfully created, the system displays the `Service is now defined` message.
 4. Type the END (or press F3) to return to the **Server ACI Facility** options menu.

5. Select **Map Refresh** to refresh the data maps.

Results

To view the ACI server definitions after creating it, see [“Displaying ACI server map information”](#).

Creating an ACI batch server definition

Procedure

1. From the **Server ACI Extract** menu, select **Create ACI Batch Server Definition** and press Enter.
2. Provide the following information for the Batch ACI server definition.

- Map Name: The name for the map.
- Server Name: This value must correspond to the service information defined in the service application.
- Server Service Class: This value must correspond to the service information defined in the service application.
- Server Service: This value must correspond to the service information defined in the service application.

Note:

- The combination of the server name, server service class, and server service identify a service.
- If any part of the name is changed while a service is active, all ACI services that are associated with the former name are treated as orphan services because an ACI service with that name no longer exists in the system. The ACI services that are associated with the former name still appear in the active ACI server maps display until they time out or until they are manually terminated.
- JCL DSN: Type of JCL DSN if submitting the service as a batch job.
- Console Command: Type of console command if submitting the service as a started task.
- Max Allowed: The maximum number of concurrent servers of this definition type that can run at any given time.
 - In cases where CICS is slow or has performance problems, a client request can be submitted to multiple ACI services. To prevent a client request from being submitted to all ACI services, the IBM Data Virtualization Manager for z/OS Interface for ACI limits the number of ACI services that the client request can submit to as described in [“Using submission limit checking”](#).
 - The Max Allowed setting does not take effect when the registration is requested by a CICS program running online (not started by the server). See [“Running a CICS program not started by Data Virtualization Manager server”](#).
- Persistent Connection: Y (Yes) allows persistent connections, and N (No) allows non-persistent connections. The following differences distinguish persistent connections from non-persistent connections:
 - A persistent connection allows ongoing conversational requests and responses. The server is "assigned" to the client until the client issues an end-of-conversation (EOC) request, at which point, the ACI server program unregisters the service and terminates. A new ACI server is started when another connection requests the same ACI service. This implies that the client, and service are in conversation mode.

It is also possible for a persistent ACI server to be reused by different client connections after the EOC request by deregistering and registering.

Note: Reusing persistent connections improves performance and reduces overhead. For information about how to create a program to reuse persistent connections, see [“Reusing persistent connections”](#).

- A non-persistent connection is one in which a single request is issued and a single response is received. The server is available for use by any client on a receive request. The service can be used by any incoming client connection with the Data Virtualization Manager server.
- Secure this Service: Y (Yes), restricts the connection to the user who has a SAF resource for the ACI service. Only a user ID with a valid resource defined for the ACI service is allowed to start and connect to that service during its life. This field defaults to N (No), which allows any user ID to start and connect to that started ACI service. The format of the resource name is:

```
ACI.aci-mapname
```

Note: To secure a persistent ACI connection, you must edit and submit one of the following sample jobs (depending on your security type) located in your *hlq.SAVZCNTL* library to specify the map name to be used.

- AVZRAACI for RACF security
- AVZA2ACI for CA ACF2 security
- AVZTSACI for CA Top Security security
- Auto Start service: Indicates to the DV that it may start this service.
- Unit of Work Participant (for persistent connections only): Indicates whether this transaction can process units of work. If so, the transaction must also support a persistent connection (the **Persistent Connection** field must be set to Y (Yes).
- Maximum UOW Buffer Size (for UOW participants only): The maximum buffer size that UOW transactions can accommodate for any single call (the maximum size of data that the ACI interface can send to the ACI service at any one time). The buffer size is rounded to 1000-byte increments and the maximum is 32,000. If 32,000 is specified, the ACI interface reduces that number to 31,767 bytes at execution time to comply with the maximum size the ACI service can receive at any one time.

Note: The client can send any size data, including SQL_LONGVARCHAR and SQL_LONGVARBINARY data types, which can be greater than 31,767 bytes long. The data that is received from the client is buffered on the Data Virtualization Manager server until a UOWLAST or UOWONLY request is received (see [“Query syntax” on page 38](#)), at which time it is sent to the ACI service in size increments that do not exceed the maximum UOW buffer size value.

- Auto Terminate (for non-persistent connections only): A number 0 - 99999 to indicate the number of receives to accept before the system automatically terminates the server and the service deregisters itself. If this field is blank, the default value is 0 (zero). Complete this field only if you specified N (No) for the **Persistent Connection** field.

Note: This field limits the number of times that the server can receive requests after which it is terminated to protect storage resources.

- Secure Server to Userid: This value defaults to N (No), which allows any user ID to reconnect to that started ACI service. To restrict the connection to the user who started the ACI service, set this value to Y (Yes). Then only the user ID that started the ACI service is allowed to reconnect to the service during its life.
- Client Non-Activity Timer: The non-activity timer, which has the following functions:
 - It is the amount of time that the client waits for the service to return before it times out.
 - If the max execution time value is not set, it is used for the time that an ACI service can run before timing out and releasing the client.
 - If the maximum wait for server timer value is not set, it is used for the time that a client can wait for an ACI service to be assigned before timing out.
 - For persistent services, it is also the amount of time that the service remains idle waiting for a client to converse with the service (the amount of time that is allowed for a client to interface with a service).

Note: For persistent services only, if the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to SERVER, the Server shutdown non-activity timer value is used for all of the functions that are listed in the client non-activity timer description.

However, the ACIPERSISTTIMEOUT would not yet apply for servers that are still in a pending registration state. Before registration, the DV uses the max wait for server. If that is not set, the DV uses the client non-activity timer. If the timer is not set, the DV uses a default of 15 seconds. For more information about the timeout values, see [“Timeout values” on page 36](#).

- **Server Shutdown Non-Activity Timer (for non-persistent connections only):** The amount of time the service can be non-active before the Data Virtualization Manager server requests it to terminate. This field allows the less frequently used servers to “die,” freeing up storage for the more frequently used servers, improving the use of available resources.

Note: For persistent services, by default, the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to CLIENT, so this field is not used, and the client non-activity timer value is used. For more information about the timeout values, see [“Timeout values” on page 36](#).

- **Maximum Wait for Server Timer:** The maximum time that a client can wait for an ACI service to be assigned before the request is timed out and the client is released. If this value is not set, the client non-activity timer value is used. For more information about the timeout values, see [“Timeout values” on page 36](#).
3. Press Enter to complete the ACI server definition. If it was successfully created, the system displays the *Service is now defined* message.
 4. Type the END command (or press F3) to return to the **Server ACI Facility** menu.
 5. Select **Map Refresh** to refresh the data maps.

Results

To view the ACI server definition after you create it, see [“Displaying ACI server map information”](#).

Extracting ACI data map information

You can extract information from a Natural source, a COBOL source, or a PL/I listing. This extraction provides information about the characteristics of the program's input and output requirements.

Extracting a map from a Natural listing

You can extract a Natural listing by using either of the following methods:

- Using the AVZMF PAR member
- Using the DMF Parser

Using the AVZMF PAR member

To use this method, run the AVZMF PAR member that is located in your *hlq.SAVZCNTL* data set as a sample JCL for extracting Natural maps.

For information about the available parameters in the AVZMF PAR member, see [“Using batch JCL jobs”](#).

Using the DMF Parser

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. From the **Server ACI Facility** menu, select **Natural Extract** and press Enter.
The **DMF Map Creation Utility** panel displays.
3. Specify the following information:
 - **Source Library Name:** The data set and member name that contains the source code for the map you want to create.
 - **Nat PGM:** The program name of your batch Natural nucleus.

- **Load Lib:** The name of the library where the Natural nucleus resides. If you are required to concatenate multiple libraries to resolve all modules that are used during Natural execution, the library names may be used.
- **PARM:** The Natural nucleus parameters that are required by your installation.
- **TEMP DSN:** The name of a temporary data set that is used as a work file by this execution.
- **Temp DSN Space:** Define this value large enough to contain the Natural object listing.
- **Logon:** The Natural library to be logged on to.
- **List:** The Natural object to be listed.
- **ADARUN:** Defines the Adabas execution requirements if not linked in the Natural nucleus. After you enter the information about the panel, press Enter. The system displays a second DMF Map Creation Utility panel.

4. Provide the following information

- **Start Field:** The field name where the map starts building.
- **End Field:** The field name where the map stops building. If not specified, the first field that is at the same level as the Start Field stops the build process.
- **Map Name:** The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
- **Use Offset Zero:** If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
- **Edit Object Listing:** Edit the object listing before the data is parsed and the data map is created.
- **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem.

Press Enter. The batch Natural nucleus is run to list the object you selected. Then, the ISPF editor may be invoked, depending on the Edit Object Listing selection, so that you can delete or modify information in the object listing.

5. Delete or modify information in the object listing, as appropriate. You can delete lines, fields, or information you do not want to be extracted. Leave any data elements that you want to be extracted in the editor.

The first three lines in the ISPF editor must be deleted, even if all of the other information is required for the extract. Line 1 must be the first line as input into the extract; therefore, preceding lines must be deleted. If this is not done, the `Not Valid Source` message appears.

6. Type the END command (or press PF3) after you complete all of your edits. The data remaining in the ISPF editor is parsed and the data map is created. An `Extract Successful` message appears on the extract screen.
7. Type the END command (or press PF3) to return to the **Server ACI Facility** menu.
8. Select **Map Refresh** to add your map to the map display list.

Extracting a map from a COBOL source or COBOL/PLI listing

You can extract a map from a COBOL source, or a COBOL or PLI listing, using either of the following methods:

- Using the AVZMF PAR member
- Using the DMF parser

Using the AVZMF PAR member

Run the AVZMF PAR member that is located in your `hlq.SAVZCNTL` data set as a sample JCL for extracting COBOL and PL/I maps.

For information about the available parameters that are located in the AVZMF PAR member, [“Using batch JCL jobs”](#).

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. From the **Server ACI Facility** menu, select **COBOL Extract** and press Enter.
The **DMF Map Creation Utility** panel displays.
3. Specify the following information:
 - **Source Library Name:** The data set name and member name that contain the source code for the map you want to create.
 - **Start Field:** The field name where the map starts building.
 - **End Field:** The field name where the map stops building. If not specified, the first field that is at the same level as the Start Field stops the build process.
 - **Map Name:** The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
 - **Use Offset Zero:** If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
 - **Convert Var to True:** Select Y (Yes) to convert VAR fields to TRUE VAR fields. TRUE VAR fields are fields that have a 2-byte length of data field that precedes the data.
 - **Flatten Arrays:** Determines whether arrays are flattened. Valid values depend on the product:
 - For IBM Data Virtualization Manager for z/OS SQL, you can specify C (COMPATIBLE) or Y (YES).
 - For IBM Data Virtualization Manager for z/OS Streams, you can specify C (COMPATIBLE) only.
 - For IBM Data Virtualization Manager for z/OS SQL 92, you can specify C (COMPATIBLE), Y (YES), or N (NO).

Note: The C (COMPATIBLE) value is provided for backwards compatibility with an older mapping architecture. When C is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
 - **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem.
4. Type the END command (or press PF3). An Extract Successful message appears on the extract screen.
5. Type the END command (or press PF3) to return to the **Server ACI Facility** menu.
6. Select **Map Refresh** to add your map to the map display list.

Displaying ACI server map information

Once the ACI servers are defined, you can view them using the ACI Map Display option. The **ACI Data Mapping Block** panel displays ACI data mapping information only. The data maps displayed in this panel represent the service, or remote application, characteristics.

About this task

To access the **ACI Data Mapping Block** panel:

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **ACI Map Display** from the **Server ACI Facility** menu. Press Enter.
The system displays the **ACI Data Mapping Block** panel.

Note: Some active ACI server information can be returned in a result set using a simple query with the IBM Data Virtualization Manager for z/OS driver. For more information, see [“Using a query” on page 32.](#)

Available commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
P	Prints map.
S	Shows map.
D	Disables map.
E	Enables map.
M	Modifies/displays map.

The M command can be used to display or modify an ACI server definition. If no fields are changed on the panel that is displayed, the map is not saved. If any field is changed, the map is saved and a refresh is required to make the changes active. Changes cannot be viewed until a refresh is done.

Column names

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description
STRUCTURE NAME	The ACI server map name.
STATUS	The active status of the ACI server.
MAX. NO. SERVERS	The maximum number of services that can be running concurrently.
ACTIVE SERVERS	The number of services that are currently running.
HIGH WATER SERVER USAGE	High water marks concerning service usage.
REGISTER COUNT	The number of times a service registers. Incremented when a REGISTER completes.
DEREG COUNT	The number of times a service deregisters. Incremented when a DEREGISTER completes.
SEND COUNT	The number of buffers a service has sent to Data Virtualization Manager server. Incremented when a SEND completes.
RECEIVE COUNT	The number of requests a service has received from a client. Incremented when a RECEIVE or RCV ON SND completes. Note: RECV READY is not counted in this count because a RECV READY means that the program is waiting for a client. As soon as the client connects, a RECEIVE occurs and this is when the count is incremented.

Column name	Description
TIMEOUT COUNT	The number of times a client has timed out waiting for a server (see “Timeout values” on page 36). Incremented when a client request (CALL DVS_ACI) times out while waiting for an ACI server to be available.
ABEND COUNT	The number of times a server has terminated abnormally. Incremented when an ACI server service abends.
WAIT COUNT	The number of times a client has been waiting for an available server. Incremented each time a CALL DVS_ACI request waits for an ACI service (for example, a WAITING FOR THE SERVER occurrence).

Note: The counts that are displayed on this panel are reset when the server is restarted.

If an error definition is defined see [“Displaying CICS global ACI counters”](#) on page 33, the columns that are described in the following table also contain information.

Column name	Description
SUSPEND COUNT	The number of times a server has been suspended because of an error.
LAST SUSPENDED DATE TIME	The last time a server was suspended.
SUSPEND ERROR	The error that caused the server to abend.
SUSP_SEC REMAINING TOTAL	The time (in seconds) until the server resumes.
TOTAL ERRORS	The total number of errors that are received by the server.
INACTIVE TIMEOUTS	The number of inactive timeouts.
MODIFICATION DATE TIME	The date and time the map was modified.
USERID	The user ID of the map creator.

Displaying all map information

The **Data Mapping Block** panel displays all data maps that are defined to this Data Virtualization Manager server.

About this task

To display map information:

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **Map Display** from the **Server ACI Facility** menu. Press Enter.

The system displays the **Data Mapping Block** panel. Several panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

Available commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
D	Disables the map so it is unavailable for use.
E	Enables the map for use.
K	Deletes a map, making it unavailable for use.
P	Prints the associated control block for the selected row.
S	Displays the associated control block for the selected row.
X	Displays the map elements for the selected row.

Column names

The following table provides a description and sort name (if available) for each column name on the ISPF panels.

Column name	Description	Sort name
STRUCTURE NAME	The data map name.	NAME
TYPE	The type of data map.	TYPE
STATUS	The status of the service: <ul style="list-style-type: none">• Enabled• Disabled• Deleted	STATUS
MR	The MapReduce status. Y indicates enabled and N indicates disabled.	MR
LANGUAGE	The language type from which this map was generated.	LANGUAGE
AT	Attachments (OPDWs) present in the map (Yes/No)	AT
MODIFICATION DATE TIME	The creation date and time of this map.	DATE
USERID	The user ID of the map creator.	USERID
CREATION DATASET	The data set from which the map was extracted.	DATASET

Copying ACI maps

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI**.

2. Then, select **Map Copy** from the **Server ACI Facility** panel. Press Enter.
The system displays the Move/Copy Utility panel.
3. Type one of the following commands in the **Option** field:
 - C to copy
 - CP to copy and print
 - M to move
 - MP to move and print
4. In the From ISPF Library fields, provide the information for the data set, including values for the Project, Group, and Type information. If the data set is partitioned, type a member name in the **Member** field:
 - To move, copy, or promote a single member, type the member name.
 - To move, copy, or promote all members, type * (asterisk).
 - To request a member selection list, leave the member name blank or specify a pattern.

Alternatively, for any other partitioned or sequential data sets, you can specify the From Other Partitioned or **Sequential Data Set** field. Type the data set name and volume serial number. Press Enter.

Note: If you forget to enter a password for a data set that requires one, or if you enter the password incorrectly, the system prompts you in standard TSO (line) mode. On TSO/TCAM systems, you may need to press the CLEAR key before responding to the password prompt. If you enter the password incorrectly or encounter any other problems, you may be prompted again to enter the password until you reach a system limit of attempts.

Displaying active ACI server information

You can display active ACI server information in either of the following ways:

- [“Using the active server display” on page 31](#)
- [“Using a query” on page 32](#)

The information that displays represents active services or remote applications that are running in the system. These services are registered to this Data Virtualization Manager server instance and are assigned a server ID. The server name is the same as that defined in the service definition.

Using the active server display

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **Active Server Display** from the **Server ACI Facility** panel display and press Enter. The system displays the **ACI Servers** panel.

Two panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

Note: Some active ACI server information can be returned in a result set by using a query. For more information, see [“Using a query” on page 32](#).

Available commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
P	Prints the map.

Line commands	Description
S	Shows the map.
K	Terminates the map.

Column names

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description
SERVER ID	The server ID.
SERVER NAME	The name of the server as defined in the service definition.
STAT	<p>The status of the service:</p> <ul style="list-style-type: none"> • 0: Waiting for work from a client. • 1: Busy or assigned conversationally to a client. • 2: Registered but not assigned. • 3: Deregistered but not released. • 5: Waiting for the program to terminate or reset. • 6: EOC issued waiting for service to process command. • 7: Start issued waiting for service to register. <p>Note: Status 4 is not used.</p>
LAST ACTIVE	<p>This value depends on the status of the service:</p> <p>For status 0, this is the number of seconds that the service has been idle.</p> <p>For other status values, this is the number of seconds that the service has been in use.</p>
CONN ID	The CICS connection name or load balancing name
TRAN ID	The CICS transaction name running in the CICS region for this service.
TASK ID	The CICS task ID running in the CICS region for this service.
MAXIMUM LAST ACTIVE	<p>The high-water mark for the LAST ACTIVE count.</p> <p>Note: The MAXIMUM LAST ACTIVE column displays on the next panel. Use the RIGHT scroll commands (or PF11 key) to scroll to the right.</p>
AVG SEND BUFFER	The send buffer size in bytes.
AVG RECV BUFFER	The receive buffer size in bytes.

Using a query

Using the driver, your application can return active ACI server information in a result set by using a query.

About this task

The syntax of the query is:

```
CALL DVS_INFO('ACTIVEACISERVERS','optional-filters')
```

where:

- ACTIVEACISERVERS (Required) causes the query to return a result set with all active ACI servers listed.
- *optional-filters* (Optional) Specifies a filter for the query. Valid filters are:
 - NAME (*server-name*): Obtains results for the server name specified. For example:

```
CALL DVS_INFO('ACTIVEACISERVERS', 'NAME(SDCIFEN)')
```

- CONNECTION (*connection-name*): Obtains results for the CICS connection name or load balancing name specified. For example:

```
CALL DVS_INFO('ACTIVEACISERVERS', 'CONNECTION(EXCS)')
```

- PERSIST (YES|NO|ALL): Obtains results for servers with the persistent status specified:
 - YES selects persistent servers.
 - NO selects non-persistent servers.
 - ALL (Default) selects all servers (persistent and non-persistent).

For example:

```
CALL DVS_INFO('ACTIVEACISERVERS', 'PERSIST(ALL)')
```

Displaying CICS global ACI counters

With the CICS Global ACI Count option, you can display the current values of MAXTASKS and the number of ACI services that are running for each CICS that is running ACI services. You can also display the last service to update the counter. If the counter is determined to be inaccurate, the counter can be updated by typing over the counter-value.

About this task

To display CICS Global ACI Counters:

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **CICS Global Count** from the **Server ACI Facility** panel and press Enter. The system displays the **Global ACI Counters Display** panel.

Use the LEFT and RIGHT scroll commands (or PF keys) to shift between the two panels that display the global ACI counters.

Note: The maximum number of ACI services started is managed globally among all Data Virtualization Manager servers. This limits the number of ACI services even when the ACI configuration would otherwise allow more services to start. Refer to the DEFINE CONNECTION statement, MAXTCUSHION parameter. This defines a value that is used to further limit the number of ACI services that can be started. The MAXTCUSHION value is subtracted from the MAXTASKS value found in CICS, and used to reserve some tasks for non-ACI work in CICS.

Use the X line command to view all of the ACI Servers for a CICS APPLID.

Converting program data types to ODBC

COBOL conversions

COBOL conversions describe how COBOL data types are converted to ODBC data types.

COBOL	ODBC
Alphanumeric	SQL_CHAR
Floating Point	(If 4 bytes) SQL_FLOAT (If 8 bytes) SQL_DOUBLE
Integer	(If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
Numeric	SQL_NUMERIC
Packed	SQL_DECIMAL

Natural conversions

Natural conversions describe how Natural data types are converted to ODBC data types.

Natural	ODBC
A-Alphanumeric	SQL_CHAR
B-Binary	(If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
C-Attribute Control	N/A
D-Date	*SQL_DECIMAL
F-Floating Point	(If 4 bytes) SQL_FLOAT (If 8 bytes) SQL_DOUBLE
I-Integer	(If 1 byte) SQL_BINARY (If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
L-Logical	SQL_BINARY
N-Numeric	SQL_NUMERIC
P-Packed	SQL_DECIMAL
T-Time	*SQL_DECIMAL

Note: Although the IBM Data Virtualization Manager for z/OS Interface for Adabas supports the conversion of ODBC date and time to the Natural date and time format, the IBM Data Virtualization Manager for z/OS Interface for Natural only allows the passing of the internal format for date and time (P6 and P12, respectively).

Reusing persistent connections

Persistent connections can be reused so that the ACI service can be used by different client connections. The number of times that a service can be reused is controlled by the application, not by the IBM Data Virtualization Manager for z/OS Interface for ACI.

Once the client issues the end-of-conversation (EOC) request, the ACI service and its CICS task normally become unavailable. For the persistent service to be reused, the program must perform one of the following actions:

- Deregister and register again. Then, go into the RECV READY state.
- Enter the RECV READY state by using `CONV-ID = 'NEW'`. In this case, the IBM Data Virtualization Manager for z/OS Interface for ACI implicitly issues a Deregister/Register on the client's behalf.

Note: Although the ACI service is reused, the server ID of the service is changed each time because of the Deregister/Register calls.

Using submission limit checking

The ACI interface limits the number of ACI services that can be submitted for the client request. When a request to start an ACI server is received from a client, the ACI interface attempts to start the ACI server or waits for a short interval depending on the following criteria:

- The active ACI server queue is searched for an available ACI server that matches the ACI service definition, as configured in the data map. If found, that server is assigned to this request and processing continues.

For more information about creating an ACI service definition, see [“Defining an ACI server map” on page 19](#)

- If either of the following situations occur, the request waits for a short interval:
 - The number of active ACI servers is greater than the Max Allowed setting of the ACI service definition, which specifies the maximum number of concurrent servers allowed.
 - The number of start attempts for this request is greater than the maximum allowed (five).

Note: The first criterion that is met determines the action taken.

If no start attempts have been made for this request, a start attempt is made. The current registration count for this ACI service definition is saved.

If the wait interval is less than a second, the request waits for another short interval.

If a start attempt is made, but the current registration count for this ACI service definition has changed, another requestor may have obtained control of the ACI server. A new ACI server is started and the current registration count for this ACI service definition is saved.

For other situations, the request waits for a short interval as defined. The interval time that a request waits starts at 0.25 seconds and doubles for each waiting interval until it reaches a maximum of 5 seconds.

Note: The `WAITING FOR SERVER` message does not appear until after the interval reaches five seconds.

The maximum amount of time that a request waits for an ACI server to be assigned is defined by the Maximum Wait For Server Timer value in the ACI service definition; otherwise, the Client Non-Activity Timer value is used.

When the maximum amount of time is reached for the client to wait for an available server, the request terminates with an error, depending on whether any servers are active for this ACI service definition:

- If at least one server is active for this ACI service definition:

```
DVS_ACI ERROR HAS OCCURRED RC -1062; TIMEOUT EXCEEDED, ALL SERVERS ARE BUSY
```

- If no servers are active for this ACI service definition:

DVS_ACI ERROR HAS OCCURRED RC -1081; NO ACI SERVICE AVAILABLE / CANNOT START ACI SERVICE - CHECK FOR SERVER FAILURE

Timeout values

Timeout values describe the amount of time to wait before timing out in the ACI server definition, how each timeout value is set, and the resulting error message that is returned to the requesting application.

<i>Table 19. ACI timeout values</i>		
Event Description	Method of control	Client error code returned
The timeout for a client waiting for an available server (the amount of time that the client waits for a service connection).	<ul style="list-style-type: none"> Maximum Wait for Server Timer If the Maximum Wait for Server Timer value is not specified, the Client Non-Activity Timer value is used. 	<p>If no server is active:</p> <pre>DVS_ACI ERROR HAS OCCURRED RC -1081; NO ACI SERVICE AVAILABLE / CANNOT START ACI SERVICE - CHECK FOR SERVER FAILURE</pre> <p>If a server is active, but unavailable:</p> <pre>DVS_ACI ERROR HAS OCCURRED RC -1062; TIMEOUT EXCEEDED, ALL SERVERS ARE BUSY</pre>
The timeout value for a client waiting for a server to return (the time that is allowed for a service to complete a unit of work before the result is sent to the client).	Client Non-Activity Timer	<pre>DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT</pre>
The maximum server execution time (the time that is allowed for a server to run).	<ul style="list-style-type: none"> Max Execution Time If the Max Execution Time is not specified, the Client Non-Activity Timer is used. 	<pre>DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT</pre>
The timeout value for an idle server waiting for a client to make a request (the amount of time the service can be non-active before Data Virtualization Manager server requests the service to terminate).	<ul style="list-style-type: none"> Non-Persistent Connections: Controlled by the Server Shutdown Non-Activity Timer. Persistent Connections: Controlled by the Client Non-Activity Timer. 	

Note: For persistent connections, the method of controlling timeout values depends on the value of the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter:

- If the parameter ACIPERSISTTIMEOUT = CLIENT (default) is defined, the client non-activity timer value is used.
- If the parameter ACIPERSISTTIMEOUT = SERVER is defined, the server shutdown non-activity timer value is used for all of the client non-activity timer functions.

Handling interrupted connections

Interrupted connections affect the following items:

- ACI service status
- Client error codes

ACI service status

When the ACI service is busy in status 1, but the connection is interrupted while issuing a CALL DVS_ACI, the IBM Data Virtualization Manager for z/OS Interface for ACI ensures that the situation is handled appropriately by marking the connections as timed out. This allows the server to clean up and deregister. The server is placed in status 5, which indicates that it is waiting for the application to terminate or to reset.

The ACI service remains in status 5 until the application responds by using a SEND/RECEIVE command. Once the SEND/RECEIVE is received, the application receives a TIMEOUT error code (#ETBCB-ERROR-CODE = TIMEOUT). The application then issues a DEREGISTER, and the ACI service is cleaned up.

Client error codes

The client receives an appropriate error code, depending on which of the following occurrences caused the interrupted connection:

- **Connection Timing Out:** If the application reaches the timeout setting while waiting for a server to return or waiting for server execution (see “Creating an ACI server definition for CICS” on page 20 for description) while issuing a CALL DVS_ACI, the application receives the following message:

```
DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT
```

Note: In the case of persistent services, subsequent calls to this service get the following message:

```
DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT
```

The service that is assigned to the client must be terminated so the client can restart another persistent service and start a new conversation. Once the service is terminated, any subsequent calls to this service receive the following message:

```
DVS_ACI ERROR HAS OCCURRED RC -1071; CONVERSATION HAS NOT BEEN ESTABLISHED OR IS TIMED OUT BY SERVICE
```

The client must start a new conversation.

- **Terminated Connection:** If the connection was terminated, the client receives the following message:

```
Host Communication Failed
```

Connections can be terminated by the following methods:

- Data Virtualization Manager server FAILxxxxxTIME parameter, which terminates the connection if the connection exceeds the value specified.
- Kill line command of the Remote User program (accessed from the Data Virtualization Manager server - Primary Option Menu).

Running a CICS program not started by Data Virtualization Manager server

CICS programs that are not started by Data Virtualization Manager server can register with Data Virtualization Manager server by using the SDBRTX table.

Note: The Max Allowed setting in the ACI service map does not take effect when registration is requested by a program that is not started by Data Virtualization Manager server because this setting limits the number of CICS transactions (the number of programs) that can be started by Data Virtualization Manager server. The MAX NO SERVERS and MAX ACTIVE SERVERS counts in the ACI server maps display do not apply for this type of registration scenario.

When registration is requested by a CICS program that is not started by Data Virtualization Manager server, registration process performs the following actions:

- Determines the Data Virtualization Manager server subsystem. The SDBRTX table is checked to see if an entry with the transaction name matches the transaction name under which the program is running:

- If a match is found, the registration goes to the Data Virtualization Manager server subsystem specified in this entry.
- If no match is found, the subsystem name on the default entry is used.
- Determines the ACI service. The ACI service is determined in the following ways:
 - If ACIDEFAULTCONNNAME ((ACI DEFAULT CONNECTION NAME) is not set, the IBM Data Virtualization Manager for z/OS Interface for ACI bypasses connection name checking. The first ACI service with a triple name that matches the triple name that is specified by the program is used for the registration process.
 - If ACIDEFAULTCONNNAME is set, the IBM Data Virtualization Manager for z/OS Interface for ACI enables connection name checking, which means that an ACI service is used for the registration process only if the triple name matches the triple name that is specified by the program and the connection name matches the value of ACIDEFAULTCONNNAME. If no match is found, the registration request receives an ACI error code of 01000100.

Query syntax

The syntax of a query is:

```
CALL DVS_ACI('function','datamaps','data1',..., 'dataN')
```

where

function is SEND, SOC, EOC, UOWFIRST, UOWMIDDLE, UOWLAST, or UOWONLY:

- SEND: The data strings are sent to the server defined in the server data map.
 - Note:** The SEND function implies that you receive information in return.
- SOC: Start of conversation. This function is required for persistent servers. It is used to obtain an existing service or to start a server and lock a server from use by the client.
- EOC: End of conversation. This function is required for persistent service. It is used to notify the service that it is no longer registered to a client.
- UOWFIRST: Indicates that this message is the first part of a unit of work (UOW). The messages are accumulated on the Data Virtualization Manager server until a request indicates a function of UOWLAST is received.
- UOWMIDDLE: Indicates that this message is not the first part or the last part of a UOW. The messages are accumulated on the Data Virtualization Manager server until a request indicates a function of UOWLAST is received.
- UOWLAST: Indicates that this message is the last part of a UOW. When this is received, the Data Virtualization Manager server processes the entire UOW, sending the messages to the ACI service in the size increments it desires.
- UOWONLY: Indicates that this is a one-message UOW. When this is received, the Data Virtualization Manager server processes the UOW, sending the messages to the ACI service in the size increments it desires.

Note: For a UOW, the size of the message segments that are sent by the client are not dependent on the size that the ACI service can accept. Any size segment can be sent by the client by using the SQL_LONGVARCHAR and SQL_LONGVARBINARY data types.

datamaps are the data maps (up to three data map names):

- (Required) Specifies the map that defines the server to which the request is being assigned. The map name is required.
- (Optional) Client map in (CMI). Defines the client input (data1-dataN) presented to the server. The CMI represents the data format expected by the service.

If CMI is coded, the data parameters data1-dataN are validated as described in [“Data validation”](#).

Note:

- For special considerations on passing numeric data with CMI, see [“Passing numeric data”](#).
- CMI is not supported for UOW calls. If a CMI is specified for a UOW call, the following message is generated:

```
DVS_ACI ERROR HAS OCCURRED RC -1086; INPUT DATA MAP NOT ALLOWED FOR UNIT
OF WORK TRANSACTIONS
```

- (Optional) Server map output (SMO). Describes the data as presented by the server. If SMO is coded, the data buffer output from the source is presented as a result set described by the SMO data map.

If the SMO is not specified, the Data Virtualization Manager server cannot determine the maximum size of the row in the result set until the first SEND call of each CALL DVS_ACI invocation is made. Once the first SEND call is issued, the Data Virtualization Manager server uses the length of the first SEND call to establish the maximum size of the row in the result set.

Note the following guidelines:

- If an optional CMI and SMO are specified, separate them by commas.
- If a CMI is omitted and an SMO is specified, use a comma as a placeholder for the CMI.
- You can run a simple CALL statement to return metadata for the CMI or SMO.

data1-dataN describes the data input to the server. If more than one data area is coded, a CMI is required.

CMI considerations

When passing an ACI input map (CMI), remember the following considerations:

- Data validation
- Passing numeric data

Data validation

If CMI is coded, the data parameters data1-dataN are validated in the following ways:

- If only one data parameter is given, the CMI is used for validation of data types only; that is, numeric fields are numeric.
- If more than one data parameter is given, the CMI is used to validate and buffer the data components as input to the server.

Passing numeric data

The following considerations exist for passing numeric data with a CMI:

- **Packed Decimal Fields.** If a field is defined as Packed Decimal in the ACI input map, the following guidelines apply:
 - If the value passed has a scale that is too long, the size of the scale in the ACI input map is used. The IBM Data Virtualization Manager for z/OS Interface for ACI allows the value if the adjusted precision is less than or equal to the precision in the ACI input map, and the scale is truncated.
 - Although the IBM Data Virtualization Manager for z/OS Interface for ACI allows you to pass a string to a Packed Decimal field, it does not allow the decimal point to be specified in the string (the decimal is based on what is defined in the ACI input map). Also, if the length of the string exceeds the precision of the field, the leading digits in the string are truncated. For these reasons, it is not recommended to pass a string value to a packed field.
- **SmallInt Fields.** The IBM Data Virtualization Manager for z/OS Interface for ACI allows a value that is passed as an integer to a field defined in the ACI input map as SmallInt. Ensure that the value is less than or equal to 32767. Otherwise, the data is truncated.

Using a CALL statement to obtain map metadata

The IBM Data Virtualization Manager for z/OS Interface for ACI allows users to view metadata information for CMI and SMO maps on the client with a simple CALL statement. This allows a user to pass all the input

parameters with the correct data types as required by the CMI or SMO without having to go into the server ISPF panels to locate this information.

The format of the CALL statement is:

```
CALL DVS_MAP('DESCRIBE',mapname')
```

where *mapname* is the name of the map.

The CALL statement returns a result set with a single column named FORMAT. This column contains details on the fields of the map. The following table describes the FORMAT column types and their SQL equivalents.

<i>Table 20. FORMAT column types and the SQL equivalent</i>	
FORMAT types	SQL types
CHARACTER	SQL_CHARACTER
NUMERIC	SQL_NUMERIC
DECIMAL	SQL_DECIMAL
INTEGER	SQL_INTEGER
SMALLINT	SQL_SMALLINT
FLOAT	SQL_FLOAT
DOUBLE	SQL_DOUBLE
DATE	SQL_DATE
TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP
VARCHAR	SQL_VARCHAR
LONGVARCHAR	SQL_LONGVARCHAR
BINARY	SQL_BINARY
VARBINARY	SQL_VARBINARY
LONGVARBINARY	SQL_LONGVARBINARY
UNICODE	SQL_UNICODE
UNICODE_VARCHAR	SQL_UNICODE_VARCHAR
UNICODE_LONGVARCHAR	SQL_UNICODE_LONGVARCHAR

Adabas

The IBM Data Virtualization Manager for z/OS Interface for Adabas allows ODBC, JDBC, and Web clients to access Adabas data in a relational model by using simple SQL-based queries. This interface can be used with traditional client/server applications, desktop productivity tools that use ODBC, and 2-tier and 3-tier Web implementations. Using the IBM Data Virtualization Manager for z/OS Interface for Adabas, any ODBC- or JDBC-enabled application can use standard ODBC or JDBC facilities to make SQL requests directly to Adabas. The result is a relational result set, with no host programming required.

The Adabas Interface Facilities option on the Data Virtualization Manager server - Primary Option Menu provides access to the Server Adabas Data Mapping Facility features.

<i>Table 21. Server Adabas Data Mapping Facility</i>	
Option	Description
Map Defaults	Set map options
Map Create	Create maps
Map Display	Display all map information
Map Copy	Copy maps
Map Refresh	Refresh maps

Creating Adabas virtual tables using the Data Mapping Facility in batch

To extract and import Adabas data, use the sample JCL in the AVZMFPAR member.

Member AVZMFPAR, which is in the *hlq.SAVZCNTL* data set, contains sample JCL for extracting Adabas virtual tables.

For information about the available parameters in the AVZMFPAR member, see [“Using batch JCL jobs”](#).

DB2

The IBM Data Virtualization Manager for z/OS Interface for DB2 offers access to DB2-z/OS data, providing maximum performance for organizations that need to integrate DB2 data with distributed or Web applications without sacrificing flexibility, reliability, or security.

Regardless of how the data is initially represented, the IBM Data Virtualization Manager for z/OS Interface for DB2 can integrate DB2 data and stored procedures without custom coding. In addition, one Data Virtualization Manager server can access many DB2 subsystems.

The DB2 Interface Facilities option on the Primary Option Menu provides access to the Server Database Control feature. The Server Database Control application allows you to view and modify the product Server Database table. This table maps database names to entries in the Link table, which can be displayed using the Link Control application. You can associate a database name with a new host name (link) using a line command.

Database control program

The Database control program allows you to view the DB2 databases and group attachment names known to the server, and to reset the logging request queue. The entries in this table are referenced for DB2 thread collection.

Invoking the DB2 control program

Procedure

From the Primary Option Menu, select **DB2** and press Enter.

The Database Control program displays the first of two connections control facility panels. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

Available commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
C	Clears the pending logging requests.

Line commands	Description
F	Formats database information for the selected row.
P	Prints the associated control block for the selected row.
S	Displays the control block for the selected row.

Column names

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
SUBSYSTEM NAME	The name of the database as it will be referred to in application programs.	NAME
SUBSYSTEM TYPE	The type of database management system.	TYPE
DATABASE VERSION	The version of the database management system	VERSION
DATABASE STATUS	The status of the database management system.	STATUS
MEMBER OF GROUP	Database is a member of group attachment.	GROUP
COMPLETED REQUESTS	The number of completed requests for the database management system.	COMPLETED REQUESTS
PENDING REQUESTS	The number of pending requests for the database management system.	PENDING REQUESTS
SSCT ADDRESS	The address of the Subsystem Communication Table (SSCT) for this database management system.	SSCT ADDRESS
RIB ADDRESS	The address of the Release Information Block (RIB) for this database management system.	RIB
DB MODE	Database operational mode. Valid values are: <ul style="list-style-type: none"> • CM: compatibility mode. • ENFM: enable new function mode. • NFM: new function mode. 	DB MODE

IBM Data Virtualization Manager for z/OS Interface for IMS DB: support for DBCTL

IMS support for DBCTL accesses IMS data by using DL/I data calls through the CCTL (coordinator controller). The CCTL provides communications for the DBCTL environment and consists of a subsystem that contains a database resource client (DRA).

The DBTCL (database control) is an environment that allows full-function and data entry databases (DEDBs) to be accessed from a management system.

The IMS Interface Facilities option on the Primary Option Menu provides access to the Server IMS Data Mapping Facility features.

Option	Description
Facilities	General IMS Facilities Menu
IMS Data Mapping	Create IMS Map Information
ODBA	Open Database Access Menu

Choosing a connectivity method

The IBM Data Virtualization Manager for z/OS Interface for IMS DB allows access to IMS data when used with the Data Virtualization Manager client, JDBC, or ODBC.

Using the IBM Data Virtualization Manager for z/OS Interface for IMS (CCTL/DBCTL), you can access data by using the method that is described in the following section.

SQL access to IMS DB

The IBM Data Virtualization Manager for z/OS Interface for IMS CCTL/DBCTL allows you to access IMS data by using SQL.

- Logical DBDs are not supported.
- The IBM Data Virtualization Manager for z/OS does not necessarily use the first PCB. It finds the best PCB within the PSB that will satisfy the query. If it is a SELECT statement, it will use a PROCOPT=GO PCB in preference to one with update capability.
- IMS can have separate positioning for each PCB. The IBM Data Virtualization Manager for z/OS can use multiple PCBs within a PSB. For each cursor opened on a database, IBM Data Virtualization Manager for z/OS will require a PCB in the PSB to use. If you open and hold open more than one cursor on the database concurrently, an additional PCB is required in the PSB for each open cursor.

Note: If there are not enough PCB available within the PSB to support the number of cursors open at the same time in the database, you will receive the following error message:

Unable to find available PCB.

If you set the **setAutoCommit** parameter to FALSE, to avoid being unable to find an available PCB, you should consider closing a cursor as soon as possible after the completion of the SQL associated with the cursor.

The process of enabling access to an IMS database involves extracting database information and issuing a query. For more information, see [“Using the method for SQL access to IMS DB” on page 51.](#)

Extracting database information

You can extract information about the database from the following sources:

- IMS Database Description (DBD)
- Program Specification Block (PSB)
- Segment detail definitions

Data Virtualization Manager server maintains segment detail definitions in the Virtualization Facility. The primary segment information can be obtained from the IMS DBD for a specific database. The DBD contains segment definitions, which can be viewed as individual segment descriptions. Segment definitions contain information that describes the relationships between segments (parent/child relationships), as well as the information access path.

IMS Database Description (DBD)

To access an IMS database, the IBM Data Virtualization Manager for z/OS Interface for IMS DB/SQL requires that the Database Description (DBD) be extracted to create a DMF data mapping entry for every DBD/segment combination.

Program Specification Block (PSB)

Program Specification Block (PSB) controls the access to IMS databases, and each PSB has one or more Program Control Blocks (PCBs) that define the access to specific databases and database segments within the IMS subsystem. To enable SQL access, a PSB must exist or be built that contains PCBs with the necessary access to the IMS data mapped in virtual tables. The source for IMS PSBs is extracted using the Data Studio tool and related to IMS virtual tables for SQL access.

While executing an SQL query, a client connection to the server defines an implicit database transaction. In IMS, a PSB schedule defines an IMS transaction. For update transactions, only one PSB can be scheduled at any point of time to ensure transactional consistency. When auto-commit is on, every SQL statement is committed, PSB is terminated, and IMS transaction is completed. When auto-commit is off, any updates issued, which are a part of a transaction, must be supported by the PSB scheduled for the transaction. Attempts to perform an update within a single transaction that would require a different PSB (from one already scheduled) returns an error.

If you are updating multiple IMS databases within a single transaction (with auto-commit off), all virtual tables participating in the transaction must use the same PSB and that PSB must include PCBs for all databases to be updated by the transaction.

The implicit database connection is ended under the following conditions:

- After each SQL statement execution when auto-commit is on. In this case, each execute instruction commits the transaction.
- After commit/rollback request when auto-commit is off.
- After the client is disconnected when auto-commit is off. In this case, a commit is issued.

Note: The auto-commit flag is set in the application code and not set within Data Virtualization Manager.

When you extract the PSB, remember the following considerations:

- Segment sensitivity considerations. Access is allowed to all segments contained in the first PCB for a PSB.
- Field sensitivity considerations. If field sensitivity is defined, WHERE clauses are allowed in the query.
- PCB considerations. Based on the SQL statement, the first DB PCB that has the necessary segment sensitivity and the processing options is selected for database access.

Segment detail definitions

Sometimes, database segments are not defined fully in the DBD. Segment layout detail definitions can be obtained from other sources, such as COBOL copybooks. To use segment detail definitions, they must be extracted to create DMF entries, which must be linked to the associated DBD segment.

When extracting the segment detail definitions, remember the following considerations:

- **Field Sensitivity:** If field sensitivity is defined, WHERE clauses are allowed in the query.
- **REDEFINES:** Redefinitions are used to change the information that is accessed by the IBM Data Virtualization Manager for z/OS Interface for IMS DB/SQL into a customized format, depending on how the information is to be presented.

For example, assume PART-KEY is redefined as PART-PREFIX and PART-NUMBER:

```

01 PART-REC
03 PART-KEY PIC X(17).
03 PART-KEY-DETAIL REDEFINES PART-KEY.
05 PART-PREFIX PIC X(02).
05 PART-NUMBER PIC X(15).
03 FILLER

```

In this case, the following SELECT statement is valid for column selection:

```
SELECT PART-PREFIX, PART-NUMBER FROM DI21PART.DFSSAM03_PARTROOT
```

- **OCCURS:** The Data Mapping Facility does not support OCCURS clauses that contain the DEPENDING ON clause. When the OCCURS clause is used, it appends a numeric suffix to the corresponding column.

For example, if you executed the following OCCURS clause on PART-PREFIX:

```
05 PART-PREFIX OCCURS 3 TIMES
```

You would see the following column names:

```

PART-PREFIX-1
PART-PREFIX-2
PART-PREFIX-3

```

Database information

Database information is contained in the following parts:

- DBD: DI21PART
- PSB: DFSSAM03

DI21PART and DFSSAM03 are samples to demonstrate how IMS support works. The samples represent how data shown in a hierarchical model is virtualized in tables.

Database definition (DBD)

This example is the DI21PART DBD of the PART sample database, represented in an IMS view in [Figure 1 on page 46](#).

```

DBD NAME=DI21PART, ACCESS=(HISAM, VSAM)
DATASET DD1=DI21PART, DEVICE=3380, OVFLW=DI21PARO,
        SIZE=(2048, 2048), RECORD=(678, 678)
SEGM   NAME=PARTROOT, PARENT=0, BYTES=50, FREQ=250
FIELD  NAME=(PARTKEY, SEQ), TYPE=C, BYTES=17, START=1
SEGM   NAME=STANINFO, PARENT=PARTROOT, BYTES=85, FREQ=1
FIELD  NAME=(STANKEY, SEQ), TYPE=C, BYTES=2, START=1
SEGM   NAME=STOKSTAT, PARENT=PARTROOT, BYTES=160, FREQ=2
FIELD  NAME=(STOCKEY, SEQ), TYPE=C, BYTES=16, START=1
SEGM   NAME=CYCCOUNT, PARENT=STOKSTAT, BYTES=25, FREQ=1
FIELD  NAME=(CYCLKEY, SEQ), TYPE=C, BYTES=2, START=1
SEGM   NAME=BACKORDR, PARENT=STOKSTAT, BYTES=75, FREQ=0
FIELD  NAME=(BACKKEY, SEQ), TYPE=C, BYTES=10, START=1
DBDGEN
FINISH
END

```

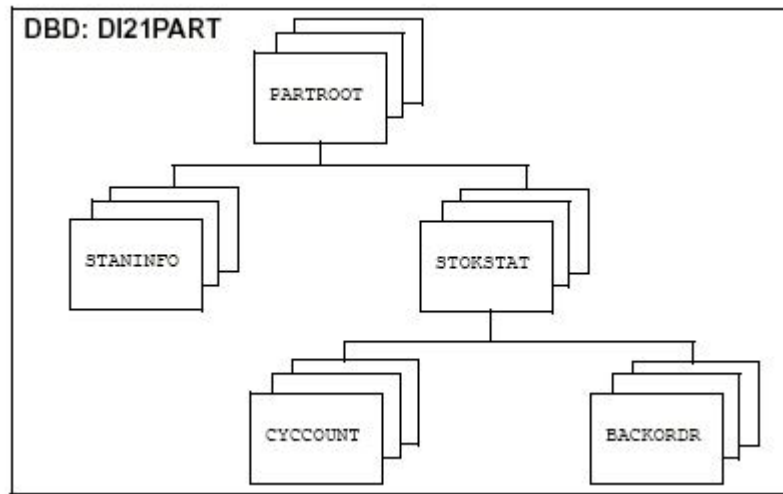


Figure 1. IMS database representation

Program Specification Block (PSB)

This example is the DFSSAM03 PSB of the PART sample database:

```

DBPCB01 PCB      TYPE=DB, DBDNAME=DI21PART, PROCOPT=G, KEYLEN=43
          SENSEG  NAME=PARTROOT, PARENT=0
          SENSEG  NAME=STANINFO, PARENT=PARTROOT
          SENSEG  NAME=STOKSTAT, PARENT=PARTROOT
          SENSEG  NAME=CYCCOUNT, PARENT=STOKSTAT
          SENSEG  NAME=BACKORDR, PARENT=STOKSTAT
          PSBGEN  LANG=COBOL, PSBNAME=DFSSAM03
          END
  
```

Extracting the data

After the maps of the DBD and PSB are extracted, you can use the Data Mapping Facility to navigate through the data.

Because IMS does not maintain a catalog that describes client information for each segment, Data Virtualization Manager server maintains the information in the Data Mapping Facility. An IMS database segment map definition is created based on the SQL statement processing requirements.

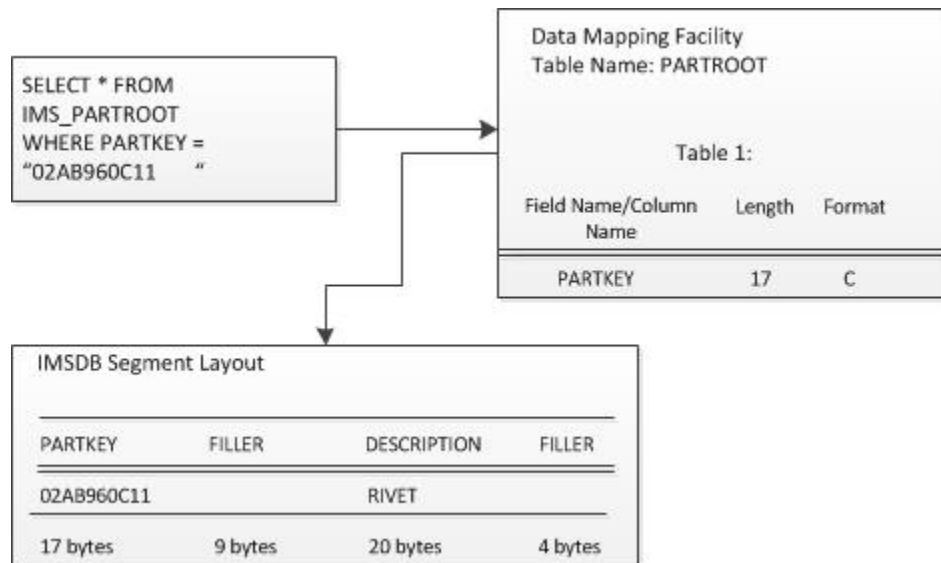


Figure 2. Using the Data Mapping Facility with the IBM Data Virtualization Manager for z/OS Interface for IMB DB/SQL

Data access paths

Data can be accessed in or across hierarchical boundaries. For the DBD shown, all of the SELECT statements that are shown in this section are valid.

The database representation of the DBD shown can be combined with the PSB and divided into specific data paths.

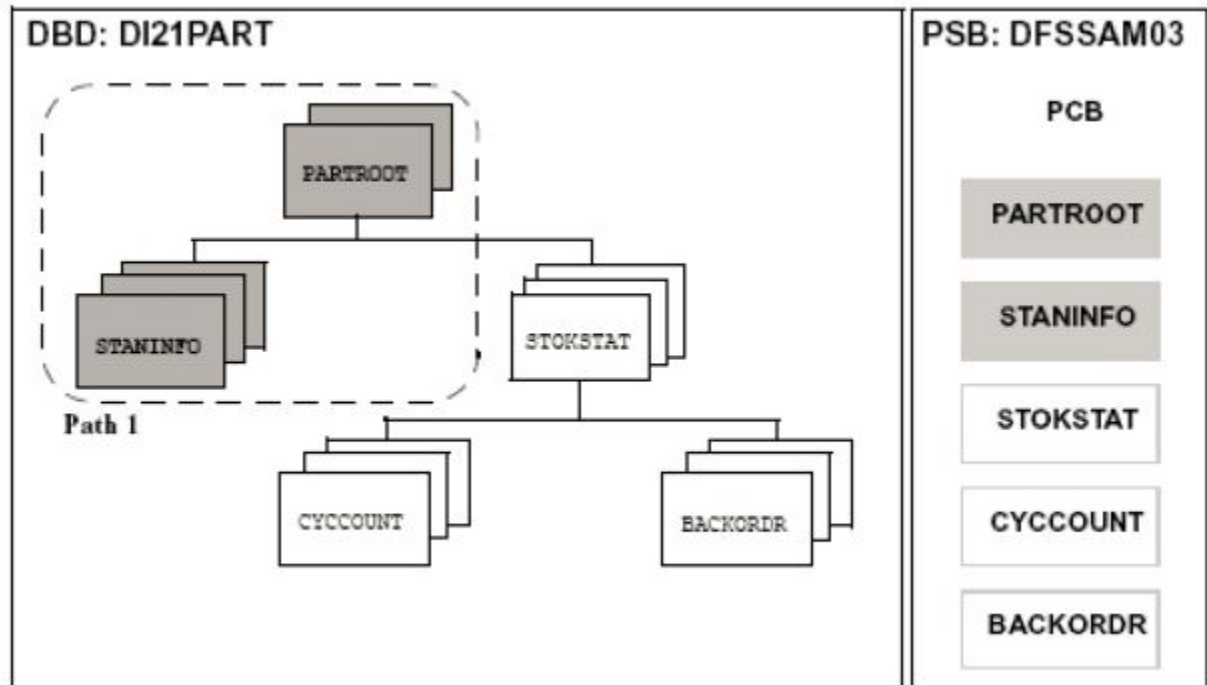


Figure 3. Data access path 1

The following SELECT statements are valid for the data access path that is shown in [Figure 3](#):

```
SELECT * FROM IMS_PARTROOT
SELECT * FROM IMS_PARTROOT P, IMS_STANINFO C
WHERE P.CHILD_ID = C.PARENT_ID
AND P.PARTKEY='02AB960C11'
```

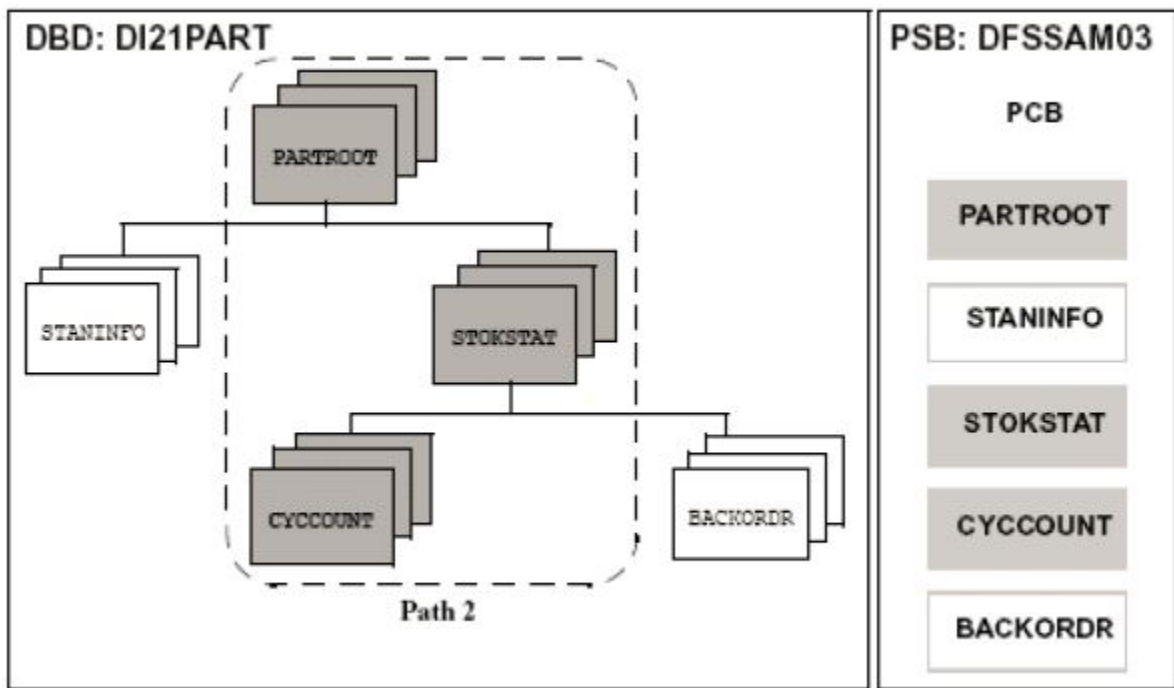


Figure 4. Data access path 2

The following SELECT statements are valid for the data access path that is shown in [Figure 4](#):

```

SELECT * FROM IMS_PARTROOT
SELECT * FROM IMS_STOKSTAT
SELECT * FROM IMS_CYCCOUNT
SELECT * FROM IMS_PARTROOT P, IMS_STOKSTAT C
  WHERE P.CHILD_ID = C.PARENT_ID
  AND PARTKEY='02AB960C11'
SELECT * FROM IMS_PARTROOT P, IMS_STOKSTAT C1,
  IMS_CYCCOUNT C2
  WHERE P.CHILD_ID = C1.PARENT_ID
  AND C1.CHILD_ID = C2.PARENT_ID
  AND PARTKEY='02AB960C11'

```

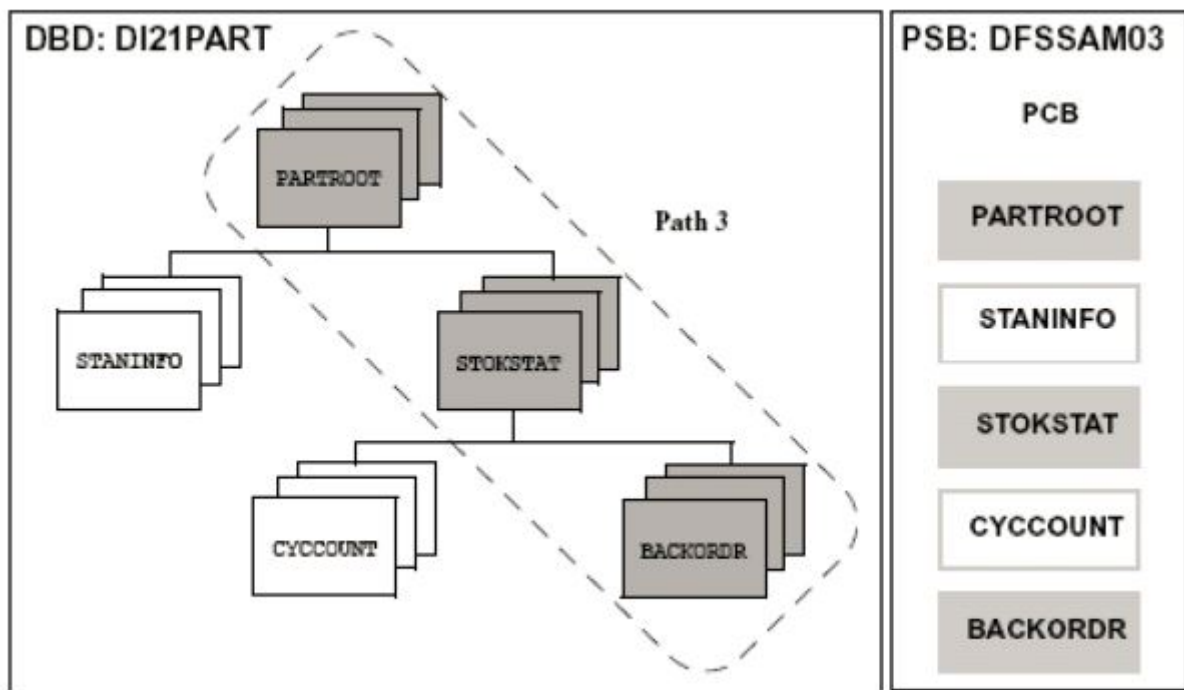


Figure 5. Data access path 3

The following SELECT statements are valid for the data access path that is shown in Figure 5:

```
SELECT * FROM IMS_PARTROOT
SELECT * FROM IMS_STOKSTAT
SELECT * FROM IMS_BACKORDR
SELECT * IMS_PARTROOT P, IMS_STOKSTAT C
  WHERE P.CHILD_ID = C.PARENT_ID
  AND PARTKEY='02AB960C11'
SELECT * FROM IMS_PARTROOT P,
  IMS_STOKSTAT C1,IMS_BACKORDR C2
  WHERE P.CHILD_ID = C1.PARENT_ID
  AND C1.CHILD_ID = C2.PARENT_ID
  AND PARTKEY='02AB960C11'
```

The following statements are not valid because they produce a Cartesian product (or Cartesian join):

```
SELECT * FROM IMS_PARTROOT, IMS_STANINFO
SELECT * FROM IMS_PARTROOT, IMS_STOKSTAT,DI21PART.
  DFSSAM03_CYCCOUNT
```

Running a statement that produces a Cartesian product results in a 1002 error code.

Note: To select from two different tables, a WHERE clause must be specified.

Selecting data

The IBM Data Virtualization Manager for z/OS Interface for IMS DB/SQL code parses the SELECT statement, optimizes it, and processes the data by using the path that is determined by the optimizer. The optimizer examines the SELECT criteria, and combines and sorts it. It also validates the access path.

For generic selections (SELECT *), all enabled columns in the data map for the segments listed in the FROM clause are returned to the client. Selected columns can be requested from any segment in a given path.

PSB security checking

The IBM Data Virtualization Manager for z/OS Server interface for IMS DBCTL/ODBA and IMS-Direct supports PSB authorization. For DBCTL/ODBA requests, Data Virtualization Manager issues a SAF call to verify the Data Virtualization Manager connected user-id has READ access to the PSB prior to scheduling

the PSB in IMS. For IMS-Direct requests, this SAF call is issued even though a PSB is not scheduled in IMS, so PSB level security is consistent regardless of the IMS data access used by the server. To activate this security checking, set the **IMSPSBSECURITY** parameter to yes in the Data Virtualization Manager server's IN00 file.

Creating a data map from SQL

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **IMS** and press Enter.
2. From the Server IMS Control Facility menu, select **IMS Data Mapping** and press Enter.
3. Select **Generate a View of an IMS/ DB DBD and Segment** from the menu and press Enter.
4. Provide the following information:
 - **Source Library Name:** The data set name and member name that contain the source code for the map you are creating.
 - **Start Field:** The field name where the map starts building.
 - **End Field:** The field name where the map stops building. If this name is not specified, the first field that is at the same level as the Start Field stops the build process.
 - **Case Sensitive:** If the Start Field or End Field are case-sensitive, set this value to Y (Yes) to preserve the case.
 - **Map Name:** The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
 - **Use Offset Zero:** If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
 - **Convert Var to True:** Set this value to Y (Yes) to convert VAR fields to TRUE VAR fields. TRUE VAR fields have a 2-byte data length field preceding the data.
 - **Flatten Arrays:** Determines how arrays are processed. Valid values depend on the data source:
 - Flatten arrays into a single fixed table at runtime (Y)
 - Return arrays into separate tables at runtime (N)
 - Flatten arrays now (C)
 - **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task.
 - **DBD Name:** The name of the DBD for which you are creating a view.
 - **Segment Name:** The name of the Segment, from the specified DBD, for which you are creating a view.
 - **PSB Name:** The name of the PSB to use to access the specified segment.
 - Note:** If you leave this field BLANK, the product automatically selects a PSB name based on the SQL query.
 - **PCB Name:** The name of the PCB, from the specified PSB, to use to access the specified segment.
 - Note:** If you leave this field BLANK, the product automatically selects a PCB name based on the SQL query.
5. If either the DBD Name or Segment Name field is BLANK, the system displays a panel allowing you to choose a name from a selection list. Select a DBD Name or Segment Name from the Selection List.
6. Press Enter. If the operation is successful, the Create Successful message appears on the panel.

Using the method for SQL access to IMS DB

The IBM Data Virtualization Manager for z/OS Interface for IMS DB provides SQL access. Use the Data Mapping Facility to define maps. Maps are defined once, and then updated/replaced, if needed.

You can extract a map from a source by using either of the following methods:

- [“Using the AVZMF PAR member” on page 51](#)
- [“Using the DMF parser”](#)

Using the AVZMF PAR member

Use the AVZMF PAR member that is located in your *hlq*.SAVZCNTL data set as a sample JCL to virtualize DBC, PSB, and COBOL maps.

For more information about the parameters in the AVZMF PAR member, see [“Using batch JCL jobs”](#).

Using the DMF parser

Procedure

1. From the Primary Option Menu, select **IMS** and press Enter.
2. From the **IMS Control Facility** menu, select **IMS Data Mapping** and press Enter.
3. Select **Mapping Defaults** from the menu and press Enter.
4. Make sure the Parser Version option is set to N (New).
5. Extract by using a DBD source:
 - a. Return to the **IMS Mapping Option** panel, and select the **Extract using DBD Source** option. Press Enter.
 - b. Provide the following information:
 - **Source Library Name:** The data set name and member name that contain the source code for the map you are creating.
 - **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task. The map name is the DBD or PSB name. This name also is used as the member name for the map in the mapping data set, if possible.
 - c. Press Enter. If the extract completes with no errors, the **Create Successful** message appears on the panel.
6. Extract the data map by using the PSB source.
 - a. Return to the **IMS Mapping Options** panel and select **Extract Using PSB Source**. Press Enter.
 - b. Provide the following information:
 - **Source Library Name:** The data set name and member name that contain the source code for the map you are creating.
 - **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task. The map name is the DBD or PSB name. This name also is used as the member name for the map in the mapping data set, if possible.
 - c. Press Enter. If the extract completes with no errors, the **Create Successful** message appears on the panel.
7. Optional: Add segment detail definitions to the extracted DBD:
 - a. Return to the **IMS Mapping Options** panel and select **Extract COBOL from listing**. Press Enter.
 - b. Provide the following information:
 - **Source Library Name:** The data set name and member name that contain the source code for the map you want to create.

- **Start Field:** The field name where the map starts building.
 - **End Field:** The field name where the map stops building. If this name is not specified, the first field that is at the same level as the Start Field stops the build process.
 - **Map Name:** The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
 - **Use Offset Zero:** If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
 - **Flatten Arrays:** Determines whether arrays are flattened. Valid values depend on the product:
 - For IBM Data Virtualization Manager for z/OS SQL, you can specify C (COMPATIBLE) or Y (YES).
 - For IBM Data Virtualization Manager for z/OS Streams, you can specify C (COMPATIBLE) only.
 - For IBM Data Virtualization Manager for z/OS SQL 92, you can specify C (COMPATIBLE), Y (YES), or N (NO).

Note: The C (COMPATIBLE) value is provided for backward compatibility with an older mapping architecture. When C is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
 - **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAP DD statement for the subsystem in the server started task.
- c. Press Enter. If the extract completes with no errors, the Extract Successful message appears on the panel. Both the map library and Data Virtualization Manager server contain the mapped structure definition.
8. Merge the other maps into the DBD maps to add the segment detail definitions from the COBOL listings to the DBD segments (see [“Merging maps into a DBD map”](#)).
9. Display the maps to make sure they were all created successfully (see [“Displaying maps”](#)).

Merging maps into a DBD map

Procedure

1. From the Primary Option Menu, select **IMS** and press Enter.
2. From the IMS Control Facility Menu, select **IMS Data Mapping** and press Enter.
3. From the panel, select **Merge Other Maps into a DBD map** and press Enter.
4. Enter information in the **DBD Map Merge Utility** panel.
 - Provide the information for the Map Data Set Library, including values for the Project, Group, Type, and Member fields (optional) for the DBD data map. Otherwise, you can use the **Other Map Data Set Name** field to specify another data set for the DBD data map.
 - To disable duplication fields, select the **Disable duplicate fields** parameter.
 - To disable FILLER fields, select the **Disable FILLER fields** parameter.

Press Enter.

 - If you specified a member name, that member is selected and the system displays the Data Map Linkages panel.
 - If you did not specify a member name, the system displays a **Selection List** panel.
5. Select a member:
 - a. From the **Selection List** panel, type one of the following commands in front of the member name:
 - B: Browse the member
 - E: Edit the member
 - S: Select the member

Note: You can process one or multiple members.

- b. Type the END command to process the members.
6. From the **Data Map Linkages** panel, in the LINK NAME column, type the names of the data maps that were extracted from the COBOL listing to link with the DBD segments. Press Enter.
7. For each DBD segment that is linked to a data map, the Data Map link established message appears in the MESSAGE column.

Note: To force a mapping update, you must delete or leave the link name blank, and press Enter to process. After you see the Warning: No Linked Data Map defined message, you can rekey the link name and press Enter to pick up the revised map layout. If you performed these steps and are unable to pick up the new definition, you must perform a Map Refresh. You can also set the option Auto Refresh to Y (Yes) on the panel prior to the map extract.

8. Type the END command to process the links. The system returns to the **IMS DBD Map Links** panel. If the linking completes with no errors, the Create Successful message appears on the panel.
9. Return to the **IMS Mapping Options** panel and select **Map Refresh** from the menu.
 - a. Press Enter for a map refresh to add your map to the map display list. If the refresh completes with no errors, the Refresh Successful message appears in the upper right corner of the panel.

Displaying maps

Displaying all maps is useful to make sure that maps are created correctly.

Procedure

1. Return to the **IMS Mapping Options** panel and select **Display IMS DB DBD Maps** from the menu and press Enter.

The system displays the DBD maps. For more information about the available line commands and column descriptions, see the following sections.

2. Return to the **IMS Mapping Options** panel and select **Display IMS DB PSB Maps** from the menu and press Enter.

For more information about the available line commands and column descriptions see the following sections.

3. Return to the **IMS Mapping Options** panel and select **Display IMS DB COBOL/PLI Extract Maps** from the menu. Press Enter.

The system displays the PSB maps.

These examples show the information that displays for existing data maps. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

• Available Commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
D	Disables the map causing it to be unavailable for use.
E	Enables the map for use.
K	Deletes a map, also making it unavailable for use.
P	Prints the associated control block for the selected row.
S	Displays the associated control block for the selected row.

Line commands	Description
X	Displays map elements for the selected row.

- **Column names**

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column names	Description	Sort name
STRUCTURE NAME	The member names in the map data set.	NAME
TYPE	One of the following types of structure: <ul style="list-style-type: none"> – ADABAS – Input – Output – Screen – LPTBL – Header – USER 	TYPE
STATUS	Status of this map (Enabled, Disabled, or Deleted)	STATUS
LANGUAGE	Language type this map was generated from (for example, Adabas, COBOL, DB2, Natural, VSAM). Determined at the time of the extract. The extracted map is independent of language type.	LANGUAGE
AT	Attachments (OPDWs) present in the map (Yes/No)	AT
MODIFICATION DATE TIME	The date and time the map was modified. Used only for informational purposes.	DATE
USER ID	The user ID of the map creator. Used only for informational purposes.	USERID
CREATION DATASET	The data set that the map was extracted from.	DATASET

VSAM and sequential files

The IBM Data Virtualization Manager for z/OS Interface for VSAM provides seamless, real-time controlled access to VSAM files, CICS-assigned KSDS VSAM files, RRDS VSAM files, and sequential files, including flat files and partitioned data sets (PDSs) as shown in the following table.

Table 23. Access to file type by interface

Interface	VSAM			QSAM
	ESDS	KSDS/IAM	RRDS	
VSAM (read-only)	YES	YES	YES	YES
VSAM CICS (read/write)	YES	YES	YES	NO
VSAM RLS (read/write)	YES	YES	YES	YES

Using the IBM Data Virtualization Manager for z/OS Interface for VSAM, any ODBC- or JDBC-enabled application can use standard ODBC or JDBC facilities to make SQL requests to VSAM and sequential files and return a result set. No host programming is required.

The VSAM/Sequential Interface Facilities option on the Primary Option Menu provides access to the Server VSAM/Sequential Data Mapping Facility features.

Table 24. Server VSAM/Sequential Data Mapping Facility

Option	Description
Map Defaults	Set the defaults for the mapping facility
Extract VSAM	Extract from a VSAM file
Extract Seq	Extract from a Sequential file
Map Display	Display all map information
Map Copy	Copy maps
Map Refresh	Refresh maps
VSAM File Control	Displays the status of VSAM files used in the system

Using the Data Mapping Facility (DMF)

Use the Data Mapping Facility (DMF) to create data maps for VSAM and sequential file access.

Creating data maps for VSAM file access

You can extract a map from a VSAM file. These instructions also apply if you are extracting a VSAM file through CICS by selecting **CICS / CICS Data Mapping / Extract VSAM** from the Primary Option Menu.

You can extract a VSAM data map by using either of the methods:

- Using the AVZMF PAR member
- Using the DMF parser

Using the AVZMF PAR member

To extract VSAM maps in batch and to extract VSAM maps with alternate indexes, run the AVZMF PAR member that is located in your *hlq.SAVZCNTL* data set as sample JCL. A compiled listing is required to perform the extract. Also, a COBOL listing with OPT(FULL) cannot be processed to produce a map. Keywords for this process define the same elements that you specify on the ISPF panels.

Note: Perform a Map Refresh before updating the display with the IBM Data Virtualization Manager for z/OS display map command.

For more information about the VSAM parameters that are located in the AVZMFPAR member, see [“Using batch JCL jobs”](#).

Using the DMF parser

To extract a VSAM data map using the DMF parser, follow these steps.

Procedure

1. From the Primary Option Menu, select **VSAM/Sequential** and press Enter.
2. From the **VSAM/Seq Data Mapping Facility** panel, select **Map Defaults** and press Enter.
3. Make sure the Parser Version is set to N (New).
4. Return to the VSAM/Seq Data Mapping Facility Display.
5. Select **Extract VSAM** from this menu and press Enter.
6. Provide the following information:
 - **Source Library Name:** The data set name and member name that contain the source code for the map being created.
 - **Start Field:** The field name where the map starts building.
 - **End Field:** The field name where the map stops building. If not specified, the first field that is at the same level as the Start Field stops the build process.
 - **Map Name:** The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set if possible.
 - **Use Offset Zero:** If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
 - **Flatten Arrays:** Determines whether arrays are flattened. Valid values depend on the product:
 - For IBM Data Virtualization Manager for z/OS SQL, specify C (COMPATIBLE) or Y (YES).
 - For IBM Data Virtualization Manager for z/OS Streams, specify C (COMPATIBLE) only.
 - For IBM Data Virtualization Manager for z/OS SQL 92 Engine, specify C (COMPATIBLE), Y (YES), or N (NO).

Note: The C (COMPATIBLE) value is provided for backward compatibility with an older mapping architecture. When C is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
 - **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem.
7. Press Enter. The system displays the VSAM Extract panel.
8. Provide the following information:
 - **For read-only VSAM files allocated to the Data Virtualization Manager server address space:** In the VSAM DSN field, type the VSAM data set name (DSN) for the data that you want to access. The DSN is dynamically allocated during the execution of the query.

Note: To create the sample VSAM file, use the sample *hlq.SAVZCNTL(DEFSTAFF)*.
 - **For READ/WRITE VSAM files via CICS:**
 - The FCT for this VSAM cluster.
 - The CICS connection name, as defined in the Data Virtualization Manager server Initialization EXEC.
 - The mirror transaction name or the transaction ID, as defined in CICS.
 - The name of the Post-Read Exit and Pre-Write Exit routines if you are using the exit processing feature.

- Type Y or N to indicate whether to use alternate indexes for this file. For this example, specify Y (Yes).
- 9. Press Enter. If you selected Y for alternate indexes, the VSAM Extract panel appears.
- 10. Specify the name of up to eight alternate indexes and press Enter. If the extract completes with no errors, the `Extract Successful` message appears on the panel.
- 11. Select **Map Refresh** from the **VSAM/Seq Data Mapping Facility** menu to refresh the data maps.

Using alternate indexes for a VSAM cluster

The IBM Data Virtualization Manager for z/OS Interface for VSAM supports VSAM alternate indexes by defining a data map that contains the following items:

About this task

- For read-only VSAM files allocated to the Data Virtualization Manager server address space, the data map is the path name in the base VSAM cluster.
- For read/write access to VSAM files by using CICS, the data map is the base cluster ID and an alternate index path ID as known to CICS.

The DMF allows for the same or different views in a VSAM file by changing the map name.

Procedure

1. From the Primary Option Menu, select **VSAM/Sequential** and press Enter.
2. From the VSAM/Seq Data Mapping Facility panel, select **Extract VSAM** and press Enter.
3. Provide the following information:
 - **Listing Library:** Type the information for the listing, including values for the Project, Group, Type, and Member fields. Alternatively, you can use the **Other Partitioned Data Set Containing Listing** field to specify the data set.
 - **Map Library:** Type the information for the map output data set, including values for the Project, Group, Type, and Member fields. Alternatively, you can use the **Other Partitioned Data Set To Contain Map** field to specify another data set for the map output.
4. Provide the following information in the Listing Search Criteria fields:
 - **Start Search Field:** This is used to search the listing data set for the starting point of the language-dependent data declaration. The search criteria must be unique enough to find the specific declaration to be mapped. For best results, use the fully qualified name of the declaration as it appears in the listing.
 - **End Search Field:** If this field is blank, extraction starts with the level number of the line found and continues until an equal or higher level is processed. If you enter a value in this field, extraction continues until the ending search string is found in the listing.
 - **Offset Zero:** (Y/N) Indicates whether to set the Start Search Field offset to zero, even if it is not a group level or the first definition in a group.
5. Press Enter. The system displays the VSAM Extract panel.
6. Indicate whether to use alternate indexes for this file. Specify Y (Yes) to allow the use of alternate indexes on this file.
7. Indicate whether to treat this file as an IAM file. The default is N (No); however, if the file is an IAM file, it is still treated as an IAM file.
8. Press Enter. The system displays the following panel.
9. Provide the following information:
 - For read-only VSAM files allocated to the Data Virtualization Manager server address space, the path name of the VSAM alternative index. You can add up to 10 alternative index names.
 - For read/write access to VSAM files by using CICS, the path name of the VSAM alternative index and the CICS FCT name. You can add up to 8 alternative index names.

Defining multiple VSAM logical records in the same file

If you are using the IBM Data Virtualization Manager for z/OS Interface for VSAM support of multiple logical records in the same file, you must define different views in the VSAM file. You create different maps that contain a different view for each of the logical records.

The following examples show two logical records from two different views in the same VSAM file. One view contains demographic information, and a second view contains account information. The RECORD_TYPE column specifies the view that contains the record.

Normally, a COBOL application that reads this data reads the record's content by using a record type (or view) indicator and then uses the redefinition of the record layout. If the COBOL program uses a redefine of the data area, the data map that is extracted contains the redefined columns. The application checks the content of RECORD_TYPE and uses the appropriate columns to view the data.

An alternative to this approach is to define the views in two separate data mapping definitions. Both data maps refer to the same file, but each has a different table name to distinguish its view in the VSAM data set. Using the preceding example, the data map DEMOGRAF can contain definitions for ACCOUNT_NUMBER, RECORD_TYPE, NAME, and ADDRESS. The data map ACCOUNT can contain ACCOUNT_NUMBER, RECORD_TYPE, and ACCOUNT_BALANCE. The application can issue the following queries to obtain all rows (records) in each view:

```
SELECT * FROM DEMOGRAF WHERE RECORD_TYPE = 1
SELECT * FROM ACCOUNT WHERE RECORD_TYPE = 2
```

To alternate the views, the application can run the following statements, where the &VALUE information is substituted from the previous query ACCOUNT_NUMBER column:

```
SELECT * FROM DEMOGRAPH WHERE RECORD_TYPE = 1
SELECT * FROM ACCOUNT WHERE ACCOUNT_NUMBER = "&VALUE" AND RECORD_TYPE = "2"
```

Creating data maps for sequential file access

You need to define a sequential file before you can access sequential files. You can define and extract this map by using either of the following methods:

- Using the AVZMFPAR member
- Using the DMF parser

Using the AVZMFPAR member

To extract sequential maps in batch and to extract sequential maps with alternate indexes, run the AVZMFPAR member that is located in your *hlq.SAVZCNTL* data set as sample JCL.

A COBOL listing with OPT(FULL) cannot be processed to produce a map. Keywords for this process define the same elements that you specify on the ISPF panels.

Note: You must perform a Map Refresh before it shows in the IBM Data Virtualization Manager for z/OS display map command.

For more information about the sequential parameters that are located in the AVZMFPAR member, see [“Using batch JCL jobs”](#).

Using the DMF parser

About this task

To extract a sequential data map using the DMF parser, follow these steps.

Procedure

1. From the Primary Option Menu, select **VSAM/Sequential** and press Enter.
2. From the VSAM/Seq Data Mapping Facility panel, select **Extract Seq** and press Enter.
3. Provide the following information:

- **Source Library Name:** The data set name and member name that contain the source code for the map you want to create.
 - **Start Field:** The field name that is used to start building the map.
 - **End Field:** The field name that is used to stop building the map. If not specified, the first field that is at the same level as the Start field stops the build process.
 - **Map Name:** The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set if possible.
 - **Use Offset Zero:** If the Start field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
 - **Flatten Arrays:** Determines whether arrays are flattened. Valid values depend on the product:
 - For Data Virtualization Manager server SQL, specify C (COMPATIBLE) or Y (YES).
 - For Data Virtualization Manager server Streams, specify C (COMPATIBLE) only.
 - For Data Virtualization Manager server SQL 92 Engine, specify C (COMPATIBLE), Y (YES), or N (NO).

Note: The C (COMPATIBLE) value is provided for backwards compatibility with an older mapping architecture. When C is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
 - **Map Data Set Name:** The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task.
4. Press Enter. The system displays the Sequential Extract panel.
 5. Provide the following information:
 - **For flat files:** The data set name in the **Enter DSN** field.
 - **For PDSs:** The data set name in the **Enter DSN** field. In addition, if you want to create a data map that includes columns for viewing or searching the data set name and/or PDS member name, provide the following information:
 - To view the data set name, in the DSN Column Name field, type a column name that represents the data set name information.
 - To view the PDS member name, in the Member Column Name field, type a column name that represents the member name information.
 - If you want to search by the data set name or PDS member name columns, specify Y (Yes) to indicate that the columns are allowed to be used in search criteria.

Note: If you do not specify the appropriate information to search by data set name or member name, a query returns information for all PDS members of all of data sets, without any indication of the corresponding member name or data set name.
 6. Press Enter. If the extract completes with no errors, the Extract Successful message appears on the panel.
 7. Return to the VSAM/Seq Data Mapping Facility and select **Map Refresh** to refresh the data maps.

Query syntax

The following syntax shows the query for each type of data file:

- **VSAM data (read-only)**

```
select (5) * from vsam1
```

- **VSAM for CICS data (read/write)**

```
select (5) * from filea
```

- **Sequential files**

```
select * from flatfile
```

Using a CALL statement to obtain map metadata

The IBM Data Virtualization Manager for z/OS Interface for VSAM and Sequential Files allows users to view metadata information for VSAM or sequential file data maps on the client with a simple CALL statement. The syntax of the call is:

```
CALL DVS_MAP('DESCRIBE','mapname')
```

where *mapname* is the name of the map.

This call returns a result set with a single column named FORMAT. The FORMAT column contains details on the fields of the map. The FORMAT column types and their SQL equivalents are shown in the following table.

FORMAT column types	SQL equivalent
CHARACTER	SQL_CHARACTER
NUMERIC	SQL_NUMERIC
DECIMAL	SQL_DECIMAL
INTEGER	SQL_INTEGER
SMALLINT	SQL_SMALLINT
FLOAT	SQL_FLOAT
DOUBLE	SQL_DOUBLE
DATE	SQL_DATE
TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP
VARCHAR	SQL_VARCHAR
LONGVARCHAR	SQL_LONGVARCHAR
BINARY	SQL_BINARY
VARBINARY	SQL_VARBINARY
LONGVARBINARY	SQL_LONGVARBINARY
UNICODE	SQL_UNICODE
UNICODE_VARCHAR	SQL_UNICODE_VARCHAR
UNICODE_LONGVARCHAR	SQL_UNICODE_LONGVARCHAR

Using the Data Mapping Facility

You can use the Data Mapping Facility to set default maps to display, copy, or refresh data maps, to view individual items in a data map, to generate RPC skeletons, and to create source library definitions.

The Data Mapping option on the Primary Option Menu provides access to the Data Mapping Facility features.

Option	Description
Map Defaults	Set the defaults for the mapping facility
Map Display	Display all map information
Map Copy	Copy maps

Table 25. Server Data Mapping Facility (continued)

Option	Description
Map Refresh	Refresh maps
VSAM File Control	Displays the status of VSAM files used in the system
Source Library Management	View or create source library definitions

Setting default values for data maps

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Defaults** and press Enter.
3. Type Y (Yes) or N (No) for Auto Refresh. Press Enter.

Y means that the storage data maps are automatically refreshed after changes.

N requires a manual refresh by using the Map Refresh option.

Auto Refresh can incur significant overhead if you have several changes to make and you exit after each change. Either make all changes before exiting, or turn off Auto Refresh and use the Map Refresh option when you are finished.

If you set this value to Y, you do not need to perform a Map Refresh before the HTML generation. If you set it to N, you must perform a Map Refresh before and after the HTML generation.

Results

The Profile Saved message appears, indicating that the data set name is saved in the user profile pool.

Displaying data maps

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Display** and press Enter.

The **Schema Listing** panel displays.

3. Type **X** next to the schema to view the **Data Mapping Block** panel for the schema.

The **Data Mapping Block** panel displays.

4. Use the available line commands to perform the appropriate functions. The following commands are available:

- P – Prints map
- S – Shows map
- D – Disables map
- E – Enables map
- K – Deletes map
- X – Displays map

Type the command name and press Enter.

Results

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
STRUCTURE NAME	The member names in the map data set.	NAME
TYPE	TYPE <ul style="list-style-type: none">• ADABAS• Input• Output• Screen• LPTBL• Header• USER	TYPE
STATUS	The status of the map (Enabled, Disabled, or Deleted).	STATUS
MR	Map Reduce (Yes/No)	MR
LANGUAGE	The language of the extracted map. This value is determined at the time of the extract.	LANGUAGE
AT	Attachments (OPDWs) present in the map (Yes/No)	AT
MODIFICATION DATE TIME	The date and time the map was last modified.	DATE
USERID	The user ID of the map creator. Used only for informational purposes.	USERID
NOTE	Comments	

Viewing individual data elements

About this task

To display the contents of a data map, use the following instructions.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Display** and press Enter.
The **Schema Listing** panel displays.
3. Type X next to the schema to view the **Data Mapping Block** panel for the schema.
The **Data Mapping Block** panel displays.
4. Type X next to the structure to view individual data elements of that structure. Press Enter.
The system displays the Data Elements for the structure.
5. Use the available line commands to perform the appropriate functions. Available commands:

- P – Prints map
- S – Shows map
- D – Disables map
- E – Enables map
- C – Changes map

Type the command name and press Enter.

Results

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column Name	Value	Description
FIELD NAME	1-50 characters	The name of the field.
COLUMN NAME	1-18 characters	The name of the column heading. During map extract, column names were created using the field names and translating any dash characters to underscores. The map editor can be used to make column names more meaningful for users.
STATUS	<ul style="list-style-type: none"> • Enabled • Disabled 	The status of the map.
LEVEL	1-nnn	The level in relation to other elements. This is maintained for informational purposes only.
LENGTH	1-65635	The length of the data element.
FORMAT	<ul style="list-style-type: none"> • Char • Bin • Packed • Decimal • Date, Time • Group 	The format of the data element.
OFFSET	1-65635	An offset is maintained as the relative position 0 (zero) displacement from the beginning of the structure.
PRECISION	1-65635	The element precision.
SCALE	1-65635	The element scale.
LINKED STRUCTURE	1-8 characters	The related structure name.
LINKED COLUMN	1-32 characters	The related structure column name.
FILL CHAR	1 character	The default fill character.
FILL DATA	1-200 characters	The default fill data.

Column Name	Value	Description
ORIGINAL STATEMENT	1-80 characters	The originating statement from which the elements were extracted. For items that were entered using the editor, these are not available.

Copying data maps

Data maps may be copied, or copied then edited to create new maps.

About this task

Use the following instructions to copy data maps.

Note: We recommend using the batch migration jobs for moving maps from one environment to another.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Copy** and press Enter.
The **Move/Copy Utility** panel displays.
3. Use the available line commands to perform the appropriate functions.

- C — Copy
- CP — Copy and Print
- M — Move
- MP — Move and Print

Type the command name and press Enter.

4. Project, Group, and Type are used for source code management. In the From ISPF Library fields, specify the following information:
 - Project
 - Group
 - Type
 - Member (if the data set is partitioned). You can perform the following actions:
 - To move, copy, or promote a single member, type the member name.
 - To move, copy, or promote all members, type * (an asterisk).
 - To request a member selection list, leave the member name blank or specify a pattern

Alternatively, for other partitioned or sequential data sets, you can specify the From Other Partitioned or Sequential Data Set field. Type the data set name and volume serial (volume serial number).

Note: If you do not enter a correct password for a data set that requires one, the system prompts you in standard TSO (line) mode. On TSO/TCAM systems, it may be necessary to press the CLEAR key before you respond to the password prompt. If you enter the password incorrectly or encounter other problems, you may be prompted again to enter the password until you reach a system limit of attempts.

Press Enter.

Refreshing data maps

Refreshing a data map may be required when changes to the underlying data structure occur. When you refresh a map, the Data Mapping Facility checks the library for modifications, and then refreshes in-core map tables from the library.

About this task

Use the following instructions to refresh a data map.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Refresh** and press Enter.
If the refresh is completed with no errors, the Refresh Successful message appears on the Server Mapping Facility options menu.

Creating source library maps

You can create new source library maps.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Server Data Mapping Facility** panel, select **Source Library Management** and press Enter.
3. From the **Data Mapping Facility** menu, select **Create Source Library Map** and press Enter.
The **Source Library Management** panel displays.
4. Enter information for the Source Library Definitions.
 - Type the name of the list of Source Libraries.
 - Enter Y or N to specify whether to Replace an existing definition.
 - Enter information to specify the Data Set Source Library or Natural Source Library. Data Set supports all libraries except Natural.

Press Enter.

The system displays the **Source Library** panel and shows the new source library map.

Results

The following table describes each column name on the ISPF panels.

Column name	Description
NAME	The source library name.
DESCRIPTION	A description of the source library.
REPLACE	Replace an existing definition. Yes or No.
DATA SET NAME	The name of the PDS/Sequential file that contains the source code.
NATURAL LIBRARY	The name of the Natural library.
ADABAS DBID	The Adabas database ID.
ADABAS FILE	The FUSER or FDIC file number.

Column name	Description
SERVER TYPE	The generic ACI server to run query: <ul style="list-style-type: none"> • B – BATCH • C – CICS

Displaying source library maps

You can view current source library maps.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Display Source Library Map** and press Enter.
The **Source Library** panel displays.
3. Use the available line commands to perform the appropriate functions. Available commands:
 - P – Prints map
 - S – Shows map
 - D – Disables map
 - E – Enables map

Type the command name and press Enter.

Bind or grant DRDA packages

Use the AVZCLBND member that is located in your *hlq.SAVZCNTL* data set as a sample JCL to bind DRDA packages to a connected user or grant access to the packages to a connected user. The binding is needed before any DDF connected subsystem can be used.

For information about the available parameters in the AVZCLBND member, see [“Sample JCL - DRDA bind”](#) on page 16.

Chapter 2. Migrating maps

Use the Map Migration utility to move your virtual table maps from a development environment to a test or production environment or from one release to another.

Before you begin

Before using the Map Migration utility, make sure that the following prerequisites have been met:

- **IBM Data Virtualization Manager studio requirements**

If migrating DB2 virtual tables, target systems used by each table must be defined in the target server using one of the following definitions:

- If you want to use the same target system name, define the target system name on the target server.
- If you want to use a different target system name, then define the new target system name, and use the `TSYS=OLD_TSYS,NEW_TSYS` parameter in the AVZGNMPPM batch migration utility.

- **Data Virtualization Manager server requirements**

Make sure that both the origin and destination servers have been started.

- **Data Virtualization Manager server security requirements**

The following table summarizes the security permissions required to use the migration utility:

	JCL library	Map export PDS	Server map data set
	The location where the JCL resides.	The PDS library to which the exported metadata objects are unloaded.	The AVZMAPP DD data set, which must be the first data set in the concatenation if the parameter NEW_MAP_DSN is not set.
Batch user ID	UPDATE	CREATE READ	N/A
Server user ID	N/A	UPDATE	UPDATE READ

About this task

The Map Migration utility facilitates change control of the virtual table maps. Change control is the process of moving the virtual table maps defined in a development environment to a test or production environment or from one release to another.

You can use the AVZGNMPPM member located in your `hlq.SAVZCNTL` data set for migrating virtual table maps. See the AVZGNMPPM member for a list of parameters available for use when migrating virtual table maps.

You can use the AVZGNMPPM member to perform the following tasks:

- Migrate one or multiple virtual table maps from one server to another.
- Change the virtual table map definition using the optional parameters. See the comments in the sample job for more details.

Procedure

1. Customize the migration utility job, AVZGNMPM, for the requirements at your site.
2. Submit the AVZGNMPM batch job. Utility job AVZGNMPM extracts the contents of the maps, stores the metadata objects in the map export PDS library, and creates the batch job that is used to rebuild the maps on the target server.
3. Submit the batch JCL that was created in the previous step to rebuild the maps on the target server.

Results

The utility extracts the content of the map export PDS and rebuilds the map on the target server.

Chapter 3. Using the studio

This section presents the information you need to access to your mainframe data using the Data Virtualization Manager studio.

Data Virtualization Manager studio overview

This topic introduces the Data Virtualization Manager studio and the function it provides in IBM Data Virtualization Manager for z/OS.

Data Virtualization Manager enables users to have seamless access to mainframe data without needing to know technical details, such as how the data is formatted or where it is located. Data Virtualization Manager provides data integration without the complexity and cost associated with extracting, transforming, and loading the data.

Data Virtualization Manager studio is an Eclipse-based user interface that communicates with the Data Virtualization Manager server. You use the studio to create metadata objects, such as virtual source libraries, virtual tables, virtual collections and virtual views on the host Data Virtualization Manager server. These server metadata objects provide the information that is necessary to access your mainframe data and off-host RDBMS data sources by virtualizing your data.

Getting started with the studio

The following list highlights the steps to start using Data Virtualization Manager studio:

- Before you can begin using the Data Virtualization Manager studio, you must first open the DV Data perspective from the **Window** menu, and then connect to the Data Virtualization Manager server on the mainframe that has the data that you want to access.
- To get access to the mainframe data, use the Data Virtualization Manager studio to create the following components on the Data Virtualization Manager server:
 - *Virtual source libraries*: A virtual source library is a reference to a library that already exists on the mainframe. Virtual source libraries point to the information (metadata) that is required to virtualize the source data. You create virtual source libraries using the **Virtual Source Library Wizard**.
 - *Virtual tables*: A *virtual table* is a map to the data that you want to access from the data source. After the virtual table is created, you can use it to generate and execute the SQL. The resulting SQL is used to read and extract the mapped data from the mainframe. You create virtual tables using the **New Virtual Table Wizard**. To get access to your data from programs or applications, you generate the code from the SQL using the **SQL Code Wizard**.
 - *Virtual views*: Optionally, you can use virtual tables to create virtual views from which you can generate SQL queries. A virtual view is the SQL SELECT statement that contains the columns from the source data that are used to read data directly from the data source. In some cases, creating a virtual view is more convenient than regenerating and editing the SQL. Virtual views are also used if your virtual table is missing columns or if you want to join different types of data. You create virtual views using the **New Virtual View Wizard**.
 - *Virtual collections*: When you create a virtual table, a virtual collection for NoSQL access to data is automatically created. You generate the JavaScript from the virtual collection and use that JavaScript to read and extract the data from the mainframe. Edits that you make to either the virtual table or collection, are automatically applied to both. To edit the contents of a virtual table, you must edit the SQL virtual table.

Note: Because the virtual table and virtual collections wizards require that you to enter the same information, only the virtual table wizards are documented.

Perspectives

The perspective that you choose determines the views and editors that become available in the workbench.

A perspective is an arrangement of views and editors in the workbench. You use perspectives to accomplish a specific task or set of tasks. When you open a perspective, the menu items, tool bars, views, editors, and wizards that are associated with that perspective become available in the workbench.

Opening perspectives

From the **Window** menu, you can open a perspective by selecting **Open Perspective** and selecting the perspective that you want to use from the drop-down menu.

DV Data perspective

The **DV Data** perspective provides the default views, editors, and wizards that you use to perform tasks that are associated with accessing your mainframe data.

Use this perspective to perform the following tasks:

- Explore mainframe resources and view metadata.
- Create and manage data sources.
- Generate and modify SQL queries.
- Create virtual tables from SQL.
- Create virtual views for use with complex SQL queries.
- Generate the code to use in your applications from SQL.

Views

The following views are available with this perspective:

Views	Description
Active Connections	Lists the open JDBC connections between the studio and one or more servers. The current active connection is used by the SQL Editor to issue SQL queries over that JDBC connection. You can create new or delete existing server connections.
Server view	Lists data resources, stored procedures, and metadata. You can perform tasks on selected objects in the tree. Explorer views include the following tabs: <ul style="list-style-type: none">• Client: Lists information that is related to data sources and application development on your local machine.• Server: Lists the Data Virtualization Manager server to which you have connected, view resources, or perform tasks.• Network: Lists the host and server connections within your network. You can choose to view or modify existing host and server connection settings.• Favorites: Lists shortcuts to the mainframe resources that you frequently access.

Views	Description
Server Trace	Applies labels to Server Trace messages for use when searching within the Server Trace view.
Lists	Use to display details for each tree node or object that is selected in an Explorer view.
Search	Use to search for a text string within the Server Trace results.
Server Trace	Use to set and gather server diagnostic information for support purposes.
Server Trace Import	Use to import Server Trace (. isx) files.
SQL Results	Use to display the result set returned from a SQL query in the SQL Results tab, and the resulting trace information in the SQL Messages tab.
NoSQL for MongoDB results	Use to display the results returned from a query.
Studio Navigator	Use to list shortcuts to key task views, wizards, and editors.
Virtualization Facility	Use to display virtual table mapping details.

Editors

The following text editors are available with this perspective:

Editors	Description
Data Source	Use to edit existing connection definitions.
Edit SQL	Use to compose SQL statements and to invoke queries against the server.
NoSQL for MongoDB	Use to edit Mongo JavaScript scripts.
Virtualization Facility	Use to edit metadata settings related to virtual tables and virtual views.

Wizards

This perspective includes wizards that guide you through tasks, such as:

- Setting the server connection.
- Creating virtual source libraries.
- Creating virtual tables and views for SQL access to data.
- Creating virtual collections for NoSQL access to data.
- Generating application code from SQL.

Services perspective

The **Services** perspective provides the default views, wizards, and editors that you use to perform tasks that are associated with creating and managing services.

The **Services** perspective allows you to manage the implementation of a Service-Oriented Architecture (SOA) with your mainframe applications. This perspective provides the tools needed to build, develop, and transform mainframe applications into web services and components. With the **Services** perspective, users can easily explore resources on the mainframe, and browse mainframe application screens, define

target systems, create Web Services, and related components, and generate components that capture mainframe application logic.

The following views are available from the **Services** perspective.

Views	Description
Services Navigator View	<p>Provides shortcuts to the following:</p> <ul style="list-style-type: none"> • Services – A shortcut to the Services folder on the Server tab. • New Target System – Starts the Target System Wizard that is used to create a target system on the Data Virtualization Manager server. The Web Services operations use the target system to map to your mainframe resources. • New Web Services Directory – Starts the Web Services Directory Wizard that is used to create a Web Services directory. The Web Services directly identifies the PDS on the mainframe that is used to get the metadata libraries. • z/OS Connect Configuration – Starts the z/OS Connect Configuration Wizard that is used to create the sample XML fragments for use in a z/OS server.xml file.
Explorer	<p>Displays the following tabs and folders:</p> <ul style="list-style-type: none"> • Client tab (local machine view) <ul style="list-style-type: none"> – zOSConnect folder – Contains the generated SAR files and the server.xml file. – zServices.Projects folder – Contains related zServices project files such as Web Services downloads. • Server tab <ul style="list-style-type: none"> – Web Services folder – Contains the Web Services and operations that are executed to get the mainframe data. – Target Systems folder – Contains the target systems that Web Services operations will use to map to mainframe resources.
Editors	Description
Editors are used to modify Web Services component details.	<ul style="list-style-type: none"> • Target System Editor • Virtual Directories Editor • Web Services Editor • Web Service Operations Editor

Connecting to the Data Virtualization Manager server

To access data on the mainframe, connect IBM Data Virtualization Manager studio to the Data Virtualization Manager server that is running on a z/OS mainframe instance.

Connecting to the Data Virtualization Manager server

Use the IBM Data Virtualization Manager studio to connect to the Data Virtualization Manager server that is running on an instance of z/OS®.

Before you begin

Before you can connect to the Data Virtualization Manager server, the server must be configured and started.

Procedure

1. From the IBM Data Virtualization Manager studio menu, click **Window > Open Perspective > DV Data**.
2. On the **Server** tab, click **Set Server**.
3. In the **Set Server** dialog box, complete the following:
 - **Host:** Select or enter the TCP/IP host name or IP address of the mainframe system on which the Data Virtualization Manager server is deployed.
 - **Port:** Enter the port number that the Data Virtualization Manager server uses. The default is 1200.
 - **Userid:** Enter your mainframe user ID.
 - **User Password:** Enter your password or password phrase for the mainframe user ID.
4. Click **OK**.

Completing the configuration of DRDA access to RDBMS data sources

To complete the configuration of DRDA access to RDBMS data sources, you must bind packages on the Data Virtualization Manager server, and grant users the authority to use those packages.

Before you begin

You must know the host name and the port number of the Data Virtualization Manager server and your log on credentials. Your log on credentials must have the authority to bind packages and grant privileges.

About this task

Perform the following task for each RDBMS data source that you want to access.

Procedure

1. From the studio, click **Window > Open Perspective > DV Data**.
2. On the **Server** tab, click **Set Server**.
3. In the **Set Current Server** dialog box, complete the following fields:

Option	Description
Host	Enter the TCP/IP host name or IP address of the mainframe system.
Port	Enter the port number that is used to communicate with the Data Virtualization Manager server. The default is 1200.
Userid	Enter the mainframe user ID.
User Password	Enter the password for the mainframe user ID.

4. Click **OK**.
5. On the **Server** tab, expand **SQL > Data > Other Subsystems**.
6. Right-click the subsystem and select **BIND/GRANT Packages**.
7. On the **BIND/GRANT Packages** page, complete the following fields:

Field	Action
Package Prefix	Enter the two character prefix to assign to the package. The package prefix must match the prefix that is defined on the mainframe server. If you change the default prefix (DS), you must also change it in the <i>hlq.SAVZEXEC (AVZSIN00)</i> file.
Number of Cursors	Enter the number of cursors to use to process results. The default is 200.
Collection	Enter the value to use to bind packages. The default is NULLID . This value is normally determined by the DB2 Administrator.
Table Qualifier	Enter the value to use to qualify unqualified SQL. This value is normally determined by the DB2 Administrator.
Owner UserId	Enter the user ID of the package owner. This value is normally determined by the DB2 Administrator.
Grant to	Set only when granting authority for the target DB2 server. The default is PUBLIC .
Bind Package	Binds the product packages. This is the default setting.
Grant Execute	Grants execute permissions on the package to the user ID that is specified in the Grant to field.
Replace Packages	Replaces an existing package for the specified subsystem. Select this option only if the package already exists.

8. Depending on the options that you select, additional dialog boxes and messages might be displayed.
9. Review the results in the **Results** text box and click **BIND/GRANT**.

Locale considerations

You can modify the data source connection definitions to use different local code pages.

Before you begin

You have the option to change the default code page (US/English IBM 1047) that the studio uses to perform character data translations between the native Java character encoding (UTF-8) and the mainframe.

Procedure

1. To configure the data source connection definition, in the **Active Connections** view, close all open connections.
2. On the **Client** tab, expand **Data Virtualization Manager > Data Sources > JDBC > Default Config File**.
3. Right-click the data source that you want to modify and click **Edit**.
4. In the **Data Source Editor**, click the **Connection String** tab.
5. Add or modify the Charset setting to use the appropriate EBCDIC code page. For example, Charset=IBM037.
6. If LGID=ENC exists in the connection string, delete it to avoid a conflict with the Charset setting.
7. Close the **Data Source Editor**.
8. When prompted, click **Yes** to save the data source definition.
9. To change the default Charset that the studio uses when creating connection definitions, from the **Window** menu select **Preferences**, expand **DV Data > Driver**.
10. In **Connection Overrides**, enter the new Charset setting and click **Apply**.
11. On the **Server** tab, expand **SQL > Data**.

12. Right-click the data source to which you want to connect and select **Create Connection Definition (DSN)**.
13. Accept the default name that is displayed or enter a new DSN name and click **OK**.
14. In the **Data Source Editor**, click the **Connection String** tab and confirm that the new Charset setting displays in the connection string.

Results

When running queries using the new data source definition, the character data (including language specific glyphs) that you chose is displayed in the **SQL Results** view.

Creating server metadata

Using the Data Virtualization Manager studio, you create the server metadata that provides the information necessary to virtualize your data. Server metadata includes virtual source libraries, virtual tables, virtual collections, and virtual views.

Creating virtual source libraries

Virtual source libraries point to the information that IBM Data Virtualization Manager for z/OS needs in order to access some types of mainframe data.

Before you begin

A virtual source library is a server metadata object that references a source library that exists on the Data Virtualization Manager (host) server. The members of the source library contain layout information specific to a type of data, for example a COBOL or PL/I copybook (copybook), Adabas Data Definition Module (DDM) views, IMS Database Definition (DBD) files, or IMS Program Specification Block (PSB) files. Virtual source libraries provide a reusable catalog of the host's data source libraries.

Note: When creating a virtual source library, the current user must have read access to the host data source library.

About this task

Virtual source libraries are a prerequisite to creating virtual tables for the following types of data sources:

- Adabas
- IMS
- IBM MQ
- Sequential
- VSAM, VSAM CICS and IAM
- zFS and HFS
- DBMS
- IDMS
- Logstream

When creating the virtual source libraries you specify the following data set (PDS/PDSE) names based on the type of data that you want to access:

- To access Adabas data, you specify the name of the PDS/PDSE that contains the Data Definition Module (DDM) views that have been set up for the Adabas data in your environment.
- To access IMS data, you may need to create multiple virtual source libraries that reference multiple types of source libraries. You may create a separate virtual source library that references the IMS DBD files, the IMS PSB files, and the copybooks that describe the layout of each IMS segment. In each case, you specify the PDS/PDSE that is specific to the source library.

- To access IBM MQ data, you specify the name of the PDS/PDSE that contains the copybook that describes the data written to the queue.
- To access sequential data, you specify the name of the PDS/PDSE that contains the copybook that describes the structure of the sequential data records.
- To access VSAM, VSAM CICS, and IAM data, you specify the name of the PDS/PDSE that contains the copybook that describes the structure of the VSAM, VSAM CICS, and IAM data records.
- To access z/FS and HFS data, you specify the name of the PDS/PDSE that contains the copybook that describes the structure of the records in the data file.

Procedure

1. On the **Server** tab, expand **Admin > Source Libraries**.
2. Right-click **Create Virtual Source Library** and select **Create Virtual Source Library**.
3. Select the **Data Set** wizard and click **Next**.
4. On the **Virtual Source Library** page, complete the following fields:

Field	Action
Name	Enter a unique, meaningful name for the virtual source library you are creating.
Description	Enter an optional description for the virtual source library.
Library Name	Enter the name of the PDS/PDSE that contains the layout information for the data you want to access.

5. Click **Finish**.

Results

The new virtual source library is displayed in the **Source Libraries** folder.

Extracting DBDs and PSBs using IMS Catalog Node

The IMS Catalog Node introduces the ability to discover and extract IMS information from the IMS Catalog database in the IBM Data Virtualization Manager for z/OS studio.

Before you begin

The IMS Catalog node appears in the IBM Data Virtualization Manager for z/OS server when the following conditions are met in the server configuration:

- DBCTL or ODBA access is configured in the server IN00 file
- IMS Catalog maps are installed in the AVZMAPP DD concatenation.

For more information on how to install virtual table and virtual view maps for IMS catalog access, see [Installing virtual table and virtual view maps for IMS catalog access](#).

About this task

The IMS catalog implemented as a standard IMS database contains information about the DBDs and PSBs. The studio supports extracting IMS Database Definition (DBD) and IMS Program Specification Block (PSB) source from the catalog database as an alternative to the pre-existing DBD and PSB extract from the source libraries. Before creating a virtual tables for IMS, you need to extract DBDs and PSBs. The studio supports extracting DBD and PSB source from the catalog database as an alternative to the pre-existing DBD and PSB extraction from the source libraries.

Procedure

To extract DBDs and PSBs carry out the following tasks.

Extracting Database Definitions (DBDs)

Use IMS Catalog Node to extract IMS Database Definitions (DBDs).

Procedure

1. Under the **Server** tab, expand **Discovery > IMS Catalog node > ABCsubsystem** and select **DBD**.
2. Right-click the required **DBD version** and select **Extract DBD source**.

Or,

- a) Right-click and select **View DBD source**.
- b) Click **Extract DBD source**.

If the DBD source is already extracted, then **Extract Source** button appears disabled.

The DBD Extract Source dialog box appears.

3. Click **Finish** to extract the DBD sources into the IMS product map library. The extracted source will be available under data virtualization facility.

If the DBD source is already extracted, then **Finish** button appears disabled.

Extracting Program Specification Blocks (PSBs)

Use IMS Catalog Node to extract IMS Program Specification Blocks (PSBs).

Procedure

1. Under the **Server** tab, expand **Discovery > IMS Catalog Node > ABCsubsystem** and select **PSB**.
2. Right-click the required **PSB version** and select **Extract PSB source**.

Or,

- a) Right-click and select **View PSB source**.
- b) Click **Extract PSB source**.

If the PSB source is already extracted, then **Extract Source** button appears disabled.

The PSB Extract Source dialog box appears.

3. Click **Finish** to extract the PSB sources into the IMS product map library. The extracted source will be available under data virtualization facility.

If the PSB source is already extracted, then **Finish** button appears disabled.

Creating schemas in the studio

To create virtual tables with similar names and have enhanced security create schemas.

About this task

Create schemas using the **Create New Schema Wizard** that is specific to the type of data that you want to access. You can create multiple schemas within a database to logically categorize the relational objects.

Procedure

1. From the studio, click **Window > Open Prespective > DV Data**.
2. On the **Server** tab, click **Set Server**.
3. In the **Set Server** dialog box, complete the following fields:

Option	Description
Scan For More Servers...	Searches for all the available servers.
Host	Enter the TCP/IP host name or IP address of the mainframe system.

Option	Description
Port	Enter the port number that is used to communicate with the Data Virtualization. The default value of the Manager Server is 1200.
User ID	Enter the mainframe user ID.
User Password	Enter the password for the mainframe user ID.

4. Click **OK**.
5. On the **Server** tab, expand **SQL > Data > ABCD** node, where *ABCD* is the name of your server.
Or,
On the **Server** tab, expand **SQL > Data** node.
6. Right-click and select **Create New Schema**.
Create New Schema Wizard appears.
7. In the **Schema Name** box, enter the name of the schema.
8. In the **High-Level Qualifier** box, enter the value for your high-level qualifier.
If *hlq* is defined in the server, this field appears disabled and displays the value defined for the *hlq* in the server. For more information see, [“Creating schema maps on the server” on page 3](#).
9. In the **Dataset name** box, enter the name of the dataset.
By default, the studio generates the dataset name as **<High Level Qualifier>D<Current Date>T<Current Time>SCHEMA**.

The dataset name is editable if the *hlq* is **not defined** in the server. If the *hlq* is defined in the server, this field appears disabled and displays value defined for the dataset in the server.

Note: Each schema has a dataset defined.
10. In the **Schema Map Name** box, enter the name of the schema map. This field is auto populated with the name entered in **Schema Name** box in step 7. You can edit the map name as per your requirement.
11. Click **Finish**.
The **Finish** button is enabled only after populating all the mandatory fields.

Accessing the schema from studio and creating virtual table

Access the schemas and you can create virtual tables.

About this task

After creating schemas, you can view the schemas available in the data. Also, view the available virtual tables in the schema and create new virtual tables under the required schema.

Procedure

1. From the studio, click **Window > Open Perspective > DV Data**.
2. On the **Server** tab, click **Set Server**.
3. In the **Set Server** dialog box, complete the following fields:

Option	Description
Scan For More Servers...	Searches for all the available servers.
Host	Enter the TCP/IP host name or IP address of the mainframe system.

Option	Description
Port	Enter the port number that is used to communicate with the Data Virtualization. The default value of the Manager Server is 1200.
User ID	Enter the mainframe user ID.
User Password	Enter the password for the mainframe user ID.

4. Click **OK**.
5. On the **Server** tab, expand **SQL > Data > SSID node** node, where *SSID node* is the name of your server. A list of available schemas appear.
6. Click **+** to expand the schema and view the virtual tables in the schema. Right-click and select **Create Virtual Table** to create a new virtual table. For more information, see [“Creating virtual tables” on page 79](#).

Creating virtual tables

To access your data, create a virtual table or virtual view that maps to your source data and that matches the definition of the source data structure on the mainframe.

From the virtual table or virtual view, you generate the SQL that is used to read and access the mapped data from the mainframe. You create virtual tables using the **New Virtual Table Wizard** that is specific to the type of data that you want to access. Some virtual tables, including SMF virtual tables, are made available during the product installation.

The following high-level procedures must be completed prior to creating a virtual table:

- Start the IBM Data Virtualization Manager studio.
- Open the DV Data perspective.
- Connect to the Data Virtualization Manager server.
- Run the **Create Source Library** wizard to create a virtual source library to map to your mainframe data. This procedure is not required to create virtual tables for RDBMS data.

Virtual table tasks

When a virtual table is selected on the **Server** tab, you can perform the following tasks:

- **Edit**: Edit the virtual table properties in the editor.
- **Copy** and **Paste**: Copy the virtual table and paste the copy under the **Virtual Tables** node.
- **Disable**: Disable the virtual table on this server.
- **Delete**: Delete the virtual table from the server.
- **Create Virtual View**: Create a virtual view from the virtual table.

Key and index information

To view a summary of key and index information for an existing virtual table, select the virtual table on the **Server** tab and from the **Window** menu, select **Show View > Properties**. The properties for the selected table are displayed in the **Properties** view.

If a virtual table includes columns that have a primary key or an index, the column is notated using the following symbols:

- Key symbol – This column is associated with a primary key.
- Superscript numeral 1 – This column is associated with a unique index, but does not have an associated primary key.
- Superscript asterisk – This column is associated with a non-unique index.

This primary key and index information is also highlighted when you browse RDBMS tables under the **Other Subsystems** tree.

You can control the identification of primary keys and indexes by setting the parameter **SQLLENDEFERIDXDISC** and using the settings in [SQL preferences](#).

Virtual collections for NoSQL data access

When you create a virtual table, a virtual collection for NoSQL access to data is automatically created. You can generate the JavaScript from the virtual collection, and use that script to read and extract the data from the mainframe. Because the virtual table and collections wizards require the same information be entered, only the virtual table wizards are documented. Edits that you make to either the virtual table or collection, are automatically applied to both. To edit the contents of a virtual table, you must edit the SQL virtual table.

Creating virtual tables for Adabas data

Create a virtual table that maps to the Adabas data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

Have the Adabas database ID and password, the file number, and the subsystem name available.

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **ADABAS** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Arrays Handling	Enable one of the following array management options: <ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime: This relates to multiple occurring (MU) fields and periodic (PE) groups. • Return arrays into separate tables at runtime: This relates to multiple occurring (MU) fields and periodic (PE) groups. A subtable is generated for each array. Subtables only support read access.

5. On the **ADABAS Details** page, complete the following fields and click **Next**:

Field	Action
DB ID	Enter the Adabas database ID.
File Number	Enter the number of the file to use.

Field	Action
Adabas Password	If the file is password-protected, enter the password. This password is stored and encrypted in the virtual table so that future queries use the same password to access the data.
SubSystem	Enter the name of the Adabas subsystem.
Max MU Count	Enter the maximum number of times to repeat the MU field. The default is 10.
Max PE Count	Enter the maximum number of times to repeat the PE field. The default is 10.
Create Count Field	Select this check box to index every MU or PE field so that the index (count) field created precedes the repeating field. This index field tells the caller how many repeating fields are being used.
Secure	Select this check box to choose the Adabas file ID number to be used for file name security.
DE Search only	Select this check box if you want the utility to generate control definitions that allow the client to only use WHERE columns that are Adabas descriptors (such as superde, subde, and hyperde).
Search by PE index	Select this check box to allow the client to target rows that match a particular occurrence of the PE field when searching rows using the WHERE clause. If this parameter is not specified, all rows where any occurrence of that PE field match the value specified will be targeted.
Unpacked to Packed	Select this check box to convert all unpacked format fields to packed format.
Binary to Integer	Select this check box to convert all 2-byte and 4-byte binary fields to short integer and integer formats, respectively.
Binary to Packed	Select this check box to map the binary fields in the Adabas file to SQL decimal columns (numeric packed decimal format) in the generated virtual table. Note the following points: <ul style="list-style-type: none"> • If the precision of the Adabas binary field allows for the possibility of a numeric value that would cause data overflow when converted to SQL decimal, the column in the virtual table will be mapped to SQL binary instead. This means that Adabas fields with precision greater than 12 will continue to be mapped to SQL binary. • If you select the Binary to Integer check box and the Binary to Packed check box, the precision of the Adabas binary field will determine if it gets mapped to an SQL integer (that is, 2-byte or 4-byte fields) or a decimal type.
Advanced	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

6. Optional: On the **Data Definition Module** page, if you have a Natural Data Definition Module (DDM) listing of the file, you can complete the following to get additional metadata information:

Field	Action
Available Source Libraries	From the list of Available Source Libraries , select the virtual source library that contains the data structure definition that you want the virtual table to use.

Field	Action
Source Library Members	Select the names of each virtual source library member that represents the data structure that you want to include. The green arrow next to a DDM indicates that it is a suggested member, not that it is selected. Use Filter patterns to filter the list.

7. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Expand the source file to verify that it displays the expected data layout.
Start Field	This field is not supported for Adabas because the entire data layout is used.
End Field	This field is not supported for Adabas because the entire data layout is used.

8. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Important: Use caution when using the `BASE_KEY` in `WHERE` predicates, (for example, `[PARENT TABLE].BASE_KEY = [CHILD TABLE].PARENT_KEY`) when joining the parent table with a child subtable, since this will result in a table scan of the entire Adabas file. It is recommended instead to use the `CHILD_KEY` (for example, `[PARENT TABLE].CHILD_KEY = [CHILD TABLE].PARENT_KEY`).

Creating virtual tables for RDBMS data sources

Create virtual tables that map to RDBMS data sources, such as Db2 for z/OS, Db2 LUW (Linux, UNIX, and Windows), Oracle, and Microsoft SQL Server.

About this task

It is recommended that you create a virtual table for each RDBMS table from which you want to access data. Creating a virtual table for each RDBMS table allows you to perform joins across data that may originate from different DRDA accessible RDBMS subsystems or to perform joins between your RDBMS data and other types of virtualized data, such as IMS or VSAM data.

This wizard allows you to create multiple virtual tables at a time if the selected source tables belong to the same RDBMS subsystem. In this wizard, a view is treated the same as a table; each table or view is mapped to a virtual table.

Db2 data access method:

When virtual tables are created for access to Db2 for z/OS data, an option is available to select the access method. *Db2 Direct* is a Data Virtualization Manager server access method that reads the data in the Db2 VSAM linear data sets directly instead of accessing the data through traditional Db2 APIs. For more information, see "Db2 for z/OS data access methods" in the *Installation and Customization Guide*.

Note: The data access method options are not displayed if the Data Virtualization Manager server does not support Db2 Direct.

Procedure

1. On the **Server** tab, explore the RDBMS metadata information by expanding the **SQL > Data > Other Subsystems** node, and then navigating down the appropriate subtree. The hierarchy begins with the subsystem, followed by the schema, and then the tables and views.
2. Select a single table or view from the tree, or use the following techniques to select multiple tables or views:
 - To select more than one individual node, hold down the Ctrl key and click each node to be included.

- To select a range of tables (or views), click the first table in the range, and then hold the Shift key and select the last table in the range. All tables within the range will be included.
- To select a group of nodes, click the parent node. All of the children under the parent node will be included. For example, select the **Tables** node to include all tables belonging to that schema. Or, select the schema node to include all tables and views under that schema.

You can use a combination of these techniques. For example, you can select two schema nodes to create virtual tables for all tables and views belonging to those two schemas.

3. Right-click the selected items and select **Create Virtual Table(s)**. The **New Virtual Tables Wizard** launches.
4. On the **New Virtual Tables for DBMS access** page, complete the following fields:

Field	Action
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Naming Pattern	Specify the format to use for the generated virtual table names. Use the following variables to create naming patterns that are derived from the RDBMS metadata: <ul style="list-style-type: none"> • {Subsystem}: Subsystem name • {Schema}: Source schema name • {Table}: Source table name
Virtual Target System	Select a virtual target system from the drop-down list. A virtual target system points to the RDBMS subsystem that contains the data that you want to access using the current virtual table. If there are no virtual target systems in the drop-down list, click Create Target System to create one. By using virtual target systems, you can easily change the name of the RDBMS subsystem that is referenced in the virtual tables. For example, you create a virtual target system called TSDSN1, and specify that it will access the RDBMS subsystem DSN1. Then, you create 50 virtual tables that access data in the RDBMS source TSDSN1 (that is, pointing to DSN1). If it becomes necessary to change the name of the RDBMS source DSN1, you only have to change it in a single place by editing the virtual target system. These target systems can be located under the SQL > Target Systems > DBMS node in the server view tree.
<ul style="list-style-type: none"> • Use traditional DB2 access (read/write, transactional integrity) • Use DB2-Direct access (read-only, high performance bulk data access) 	<p>Select the access method to use when accessing Db2 for z/OS data.</p> <p>Choose Use traditional DB2 access (read/write, transactional integrity) to use Db2 APIs such as DRDA, CAF, and RRSAP. This is the default selection.</p> <p>Choose Use DB2-Direct access (read-only, high performance bulk data access) to use Db2 Direct.</p> <p>Note: These options are available only when creating virtual tables for access to Db2 for z/OS data and if the Data Virtualization Manager server supports Db2 Direct.</p>
Post-Read Exit Name	To manipulate the data after reading it from the source file, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.

Field	Action
Pre-Write Exit Name	To manipulate the data before writing it to the source file, enter the name of the pre-exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing before a record is read from the data source.
Advanced	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

5. In the results table, review the list of selected entries. Modify the selections as needed.

Tip: Use the check box in the header row of the table to control the selection of all entries.

6. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Creating virtual tables for IMS data

Create a virtual table that maps to the IMS data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

The Program Specification Block (PSB) and Database Definition (DBD) source members, and the copybooks for each segment must exist in the virtual source libraries defined to the server. For details, see [“Creating virtual source libraries ”](#) on page 75.

To use the IMS Direct feature, the IMSDIRECTENABLED parameter must be enabled in the Data Virtualization Manager server IN00 file.

About this task

When an IMS SQL query is run, the SQL Engine for the server will determine if the request is best executed using IMS Direct (native file support) or if IMS APIs are required. The determination is based on the database and file types supported as well as the size of the database.

Procedure

1. Expand the **SQL > Data > SSID** node, where **SSID** is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **IMS** wizard and click **Next**.
4. On the **New IMS virtual Table(s)** page, create metadata for an IMS virtual table by completing the following steps:
 - a) Choose a DBD by doing one of the following steps:
 - Select a **DBD** from the drop-down list.
 - If your DBD does not appear in the drop-down list, click **Extract DBD** to create the requisite metadata. The **New IMS DBD Metadata Wizard** launches. See [“Using the IMS DBD Metadata wizard”](#) on page 85.
 - b) Choose a PSB by doing one of the following steps:

- Select a **PSB** from the drop-down list.
 - If your PSB does not appear in the drop-down list, click **Extract PSB** to create the requisite metadata. The **New IMS PSB Metadata Wizard** launches. See [“Using the IMS PSB Metadata wizard”](#) on page 86.
- c) Click **Create Virtual Table** to create a virtual table for an IMS segment in the selected DBD and PSB. The **New Virtual Table Wizard** launches. See [“Using the IMS Virtual Table wizard”](#) on page 87.

Note: Both the DBD and PSB must be defined for this button to be enabled.

5. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Using the IMS DBD Metadata wizard

Use the **New IMS DBD Metadata Wizard** to create DBD server metadata.

About this task

This wizard is used to create server metadata containing information extracted from the selected DBD source. This DBD metadata is a prerequisite for creating IMS virtual tables. The name of each DBD map will be determined from the contents of the DBD source.

Procedure

1. On the **New DBD Metadata** page, complete the following fields and click **Next**:

Field	Action
Metadata Library	From the drop-down list, select the target library where the DBD metadata will be stored (for example, <i>hlq.USER.MAP</i>). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.

2. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of Available Source Libraries , select the virtual source library that contains the DBD source member.
Source Library Members	Select the DBD that you want to use and click Download to copy the member from the mainframe to your desktop. Use Filter patterns to filter the list.
Downloaded Source Files	Review the list of downloaded members and ensure that the check box for the DBD that you want to use has been selected.

3. On the **Data Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Expand the source file to verify that it displays the expected database definition (DBD).
Start Field	Accept the default root start field, or if multiple DBD nodes are present in the source tree, you can click on one of the DBD nodes to indicate that you only want to map that one DBD.
End Field	End Field selection is disabled when extracting DBD source.

4. On the **IMS Server configuration** page, complete the following fields:

Field	Action
<ul style="list-style-type: none"> • Use IMS/DBCTL (read/write, transactional integrity) • Use IMS-Direct (read-only, high performance bulk data access) 	<p>Select the IMS protocol to use.</p> <p>Choose Use IMS/DBCTL (read/write, transactional integrity) to use IMS API calls.</p> <p>Choose the default option Use IMS-Direct (read-only, high performance bulk data access) to enable IMS Direct for the DBD. To use this feature, IMS Direct must also be enabled in the Data Virtualization Manager server IN00 file. You must select this option for the DBD to be able to enable IMS Direct for a virtual table.</p>
IMS ID Override (used with IMS-Direct only)	Specify the IMS ID of the IMS subsystem to use when multiple IMS subsystems are defined for use with IMS Direct. This value will override the default IMS ID in the DBD map.
Advanced	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

5. Click **Finish**.

What to do next

Return to the **New IMS Virtual Table(s)** page and define the IMS PSB. See [“Creating virtual tables for IMS data”](#) on page 84.

Using the IMS PSB Metadata wizard

Use the **New IMS PSB Metadata Wizard** to create PSB server metadata.

About this task

This wizard is used to create server metadata containing information extracted from the selected PSB source. This PSB metadata is a prerequisite for creating IMS virtual tables. The name of each PSB map will be determined from the contents of the PSB source.

Procedure

1. On the **New PSB Metadata** page, complete the following fields and click **Next**:

Field	Action
Metadata Library	From the drop-down list, select the target library where the PSB metadata will be stored (for example, <i>hlq.USER.MAP</i>). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.

2. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of Available Source Libraries , select the virtual source library that contains the PSB source member.

Field	Action
Source Library Members	Select the PSB that you want to use and click Download to copy the member from the mainframe to your desktop. Use Filter patterns to filter the list.
Downloaded Source Files	Review the list of downloaded members and ensure that the check box for the PSB that you want to use has been selected.

3. On the **Data Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Expand the source file to verify that it displays the expected program specification block (PSB).
Start Field	Accept the default root start field, or if multiple PSB nodes are present in the source tree, you can click on one of the PSB nodes to indicate that you only want to map that one PSB.
End Field	End Field selection is disabled when extracting DBD source.

4. Click **Finish**.

What to do next

Return to the **New IMS Virtual Table(s)** page and create the virtual table. See [“Creating virtual tables for IMS data”](#) on page 84.

Using the IMS Virtual Table wizard

Use the **New Virtual Table Wizard** to create a new IMS virtual table.

About this task

This wizard is used to map an IMS segment using a copybook representation to produce a new IMS virtual table.

Procedure

1. On the **New IMS Virtual Table** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	Select one of the following options: <ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime (Y): This option supports both OCCURS and OCCURS DEPENDING ON statements.

Field	Action
	<ul style="list-style-type: none"> • Return arrays into separate tables at runtime (N): This option supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables support SQL read access only.

2. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of Available Source Libraries , select the virtual source library that contains the data structure definition that you want the virtual table to use.
Source Library Members	Select the PDS members that represent the data structures to include and click Download to copy the members from the mainframe to your desktop. Use Filter patterns to filter the list.
Downloaded Source Files	Select one or more previously downloaded members.

3. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	<p>Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows:</p> <ul style="list-style-type: none"> • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, Enable End Field Selection must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, Enable End Field Selection must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is not selected, and select a different start field in the Source tree.
Use Offset Zero	<p>Select this checkbox to start the selected structure from offset Zero (0).</p> <p>By default, the selected structure starts from zero and the Use Offset Zero checkbox is selected.</p> <p>Clear the Use Offset Zero checkbox to calculate the offset from the start.</p>
Enable End Field Selection	Use this field to control selection of the start field and end field values in the Source tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is selected, and select a different end field in the Source tree.

4. On the **IMS Information** page, complete the following fields:

Field	Action
Segment Name	From the drop-down list, select the segment name.
• Use IMS/DBCTL (read/write,	Select the IMS protocol to use.

Field	Action
transactional integrity) <ul style="list-style-type: none"> • Use IMS-Direct (read-only, high performance bulk data access) 	Choose the default option Use IMS/DBCTL (read/write, transactional integrity) to use IMS API calls. Choose Use IMS-Direct (read-only, high performance bulk data access) to enable IMS Direct on the virtual table. To use this feature, IMS Direct must also be enabled for the selected DBD and enabled in the Data Virtualization Manager server IN00 file.
Post-Read Exit Name:	To manipulate the data after reading it from the source file, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.
Advanced	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

5. Click **Finish**.

What to do next

Return to the **New IMS Virtual Table(s)** page and if necessary create the next virtual table. See [“Creating virtual tables for IMS data”](#) on page 84.

Creating virtual tables for Logstream

Use the **New Virtual Table Wizard** to create a new Logstream virtual table.

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **Logstream** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	Enable one of the following array management options:

Field	Action
	<ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime: This relates to multiple occurring (MU) fields and periodic (PE) groups. • Return arrays into separate tables at runtime: This relates to multiple occurring (MU) fields and periodic (PE) groups. A subtable is generated for each array. Subtables only support read access. • Flatten arrays now: If you select this option, you cannot change array-handling after you save the virtual table.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	Select the source library that contains the data structure to use.
Source Library Members	Select the PDS members that represent the data structures to include and click Download to copy the members from the mainframe to your desktop. Use Filter patterns to filter the list.
Download Source Files	Select one or more previously downloaded members.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	<p>Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows:</p> <ul style="list-style-type: none"> • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, Enable End Field Selection must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, Enable End Field Selection must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is not selected, and select a different start field in the Source tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the Source tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is selected, and select a different end field in the Source tree.

7. Optional: On the **Virtual Table Redefines** page, accept the default table redefines or expand **Redefine** to modify your selection, and click **Next**.
8. Click **Finish**.
9. On the **Data Source Details** page, complete the following data source fields and click **Next**:

Field	Action
Data Set Name	Enter the Logstream data set name you want to use.
Post-Read Exit Name	To manipulate the data after reading it from the source file, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server

Field	Action
	and is used to perform additional processing after a record is read from the data source.
Advanced	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

10. Click **Finish**.

Modifying a Logstream virtual table

Once a Logstream virtual table is created, Data Virtualization Manager lets you modify it.

About this task

You can modify the following values in the created virtual table.

- Source dataset name.
- Post-Read exit name.
- Pre-Write exit name.
- MapReduce thread count.

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. Expand **Virtual Tables** and choose the virtual table that you want to modify.
3. Right-click the virtual table and click **Edit**.
4. Provide new values in **Dataset Name:**, **Post-Read Exit Name:**, or **Pre-Write Exit Name:**.
5. Click **Advanced** to change **Thread Count** values in MapReduce.
6. Save the changes.

Creating virtual tables for IBM MQ

Create a virtual table that maps to the IBM MQ data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

Before creating the virtual table, verify that the MQ queue exists and that the copybook exists in the source library. If you use delimited data, configure support for delimited data processing. See "Configuring delimited data support" in the *Installation and Customization Guide*.

About this task

Data in MQ queues is described using COBOL or PLI data descriptions taken from copybooks or programs.

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **MQ** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Arrays Handling	Enable one of the following array management options: <ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements. • Return arrays into separate tables at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables only support SQL read access.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	Select the source library that contains the data structure to use.
Source Library Members	Select the PDS members that represent the data structures to include and click Download to copy the members from the mainframe to your desktop. Use Filter patterns to filter the list.
Downloaded Source Files	Select one or more previously downloaded members.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows: <ul style="list-style-type: none"> • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, Enable End Field Selection must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, Enable End Field Selection must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is not selected, and select a different start field in the Source tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the Source tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is selected, and select a different end field in the Source tree.

7. On the **MQ Details** page, complete the following fields:

Field	Action
Queue Manager Name	Enter the IBM MQ queue manager name. The name is a four-character subsystem name.
Queue Name	Enter the IBM MQ queue name. The name can contain a maximum of 48 characters and must comply with z/OS data set naming standards.
Post-Read Exit Name	To manipulate the data after reading it from the queue, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.

8. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Creating virtual tables for VSAM, VSAM CICS, and IAM data

Create a virtual table that maps to the VSAM, VSAM CICS, and IAM data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

You must have the VSAM or VSAMCICS cluster name available (*sourcelibrary.copybook.filename*).

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **VSAM/VSAMCICS** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	Enable one of the following array management options: <ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements. • Return arrays into separate tables at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables only support SQL read access.

Field	Action
	<ul style="list-style-type: none"> • Flatten arrays now: If you select this option, you cannot change array-handling after you save the virtual table.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of Available Source Libraries , select the virtual source library that contains the data structure definition that you want the virtual table to use.
Source Library Members	Select the PDS members that represent the data structures to include and click Download to copy the members from the mainframe to your desktop. Use Filter patterns to filter the list.
Download Source Files	Select one or more previously downloaded members.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	<p>Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows:</p> <ul style="list-style-type: none"> • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, Enable End Field Selection must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, Enable End Field Selection must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is not selected, and select a different start field in the Source tree.
Use Offset Zero	<p>Select this checkbox to start the selected structure from offset Zero (0).</p> <p>By default, the selected structure starts from zero and the Use Offset Zero checkbox is selected.</p> <p>Clear the Use Offset Zero checkbox to calculate the offset from the start.</p>
Enable End Field Selection	Use this field to control selection of the start field and end field values in the Source tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is selected, and select a different end field in the Source tree.

7. Optional: On the **Virtual Table Redefines** page, accept the default table redefines or expand **Redefine** to modify your selection, and click **Next**.

8. Complete the following VSAM related fields:

Field	Action
Cluster Name	Enter the cluster name for the VSAM data set, and click Validate . The server searches the catalog on the mainframe to confirm that the data set exists. If the data set exists, a dialog displays the data set type.

Field	Action
FCT Name (VSAMCICS only)	Enter the name of the VSAM file that is defined for CICS.
CICS Connection Name (VSAMCICS only)	Enter the name of the CICS connection that is configured on the Server.
Mirror Transaction Name (VSAMCICS only)	Enter the name of the Mirror Transaction that is defined for CICS.
Post-Read Exit Name	To manipulate the data after reading it from the source file, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.
Pre-Write Exit Name	To manipulate the data before writing it to the source file, enter the name of the pre-exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing before a record is read from the data source.
Alternate Indexes	If the VSAM file has been defined to include alternate indexes, you can click Get to add index information to the virtual table, or you can click Delete to remove the information. Alternate indexes are used to improve query performance when the search criteria includes columns that are not part of the primary index. Alternate indexes have an indirect relationship to the cluster name, but they must be defined separately. If you are using a KSDS VSAM or ESDS cluster, you can specify alternative indexes that are associated with the cluster.
Advanced (VSAM only)	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

9. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Creating virtual tables for sequential data

Create a virtual table that maps to the sequential data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

Before creating the virtual table, verify that the data set name exists and that the copybook exists in the source library.

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.

3. Under **Wizards**, select the **Sequential** wizard and click **Next**.

4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	<p>Enable one of the following array management options:</p> <ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements. • Return arrays into separate tables at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables only support SQL read access. • Flatten arrays now: If you select this option, you cannot change array-handling after you save the virtual table.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	Select the source library that contains the data structure to use.
Source Library Members	Select the PDS members that represent the data structures to include and click Download to copy the members from the mainframe to your desktop. Use Filter patterns to filter the list.
Download Source Files	Select one or more previously downloaded members.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	<p>Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows:</p> <ul style="list-style-type: none"> • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, Enable End Field Selection must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, Enable End Field Selection must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is not selected, and select a different start field in the Source tree.

Field	Action
Use Offset Zero	Select this checkbox to start the selected structure from offset Zero (0). By default, the selected structure starts from zero and the Use Offset Zero checkbox is selected. Clear the Use Offset Zero checkbox to calculate the offset from the start.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the Source tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is selected, and select a different end field in the Source tree.

- Optional: On the **Virtual Table Redefines** page, accept the default table redefines or expand **Redefine** to modify your selection, and click **Next**.
- On the **Data Source Details** page, complete the following data source fields and click **Next**:

Field	Action
Treat the Dataset as GDG	Select the Treat the Dataset as GDG checkbox to treat the data set name as GDG data set name. On selecting Treat the Dataset as GDG checkbox, the Member Name text box will appear disabled.
Data Set Name	Enter the data set name you want to use. The following data set types are supported: <ul style="list-style-type: none"> PDS or PDSE: Specify the partitioned data set name. This requires that you also enter a Member name prior to validating that the member name exists on the host. Physical sequential: Specify the sequential data set name and click Validate to verify that the data set name exists on the host. Generation Data Groups (GDG): Specify the GDG data set using the GDG syntax. For example: <i>hlq.DATA.SEQ(-1)</i>. You can also specify a base GDG name so that all generations of the GDG will potentially be accessed. Click Validate to verify that the data set name exists on the host.
Member	If you selected a PDS or PDSE for the Data Set Name , you must also enter the member name to use. Click Validate to verify that the member name exists on the host.
Post-Read Exit Name	To manipulate the data after reading it from the source file, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.
Advanced	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

- Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Creating virtual tables for zFS and HFS file system data

Create a virtual table that maps to file data that you want to access on a zFS or HFS file system and from which the SQL used to access the data is generated and executed.

Before you begin

Before creating the virtual table, verify that the PDS members that represent the data structures for the data you want to virtualize already exist in the source library.

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **zFS** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	Enable one of the following array management options: <ul style="list-style-type: none">• Flatten arrays into a single fixed table at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements.• Return arrays into separate tables at runtime: This supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables only support SQL read access.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Download Folder	Verify that the appropriate download folder is displayed.
Available Source Libraries	Select the source library that contains the data structure to use.
Source Library Members	Select the PDS members that represent the data structures to include and click Download to copy the members from the mainframe to your desktop.
Downloaded Source Files	Select one or more previously downloaded members. Selecting previously downloaded members is optional.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	<p>Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows:</p> <ul style="list-style-type: none"> • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, Enable End Field Selection must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, Enable End Field Selection must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is not selected, and select a different start field in the Source tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the Source tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure Enable End Field Selection is selected, and select a different end field in the Source tree.

7. On the **zFS Virtual Table Details** page, complete the following fields:

Field	Action
Pathname	<p>Enter the path name of the zFS file.</p> <p>If the absolute path name of the zFS file is less than 255 characters in length, you must include the root slash "/" in the path name. For example, /u/tsado/data/stuff.txt.</p> <p>If the absolute path name of the zFS file is greater than 255 characters in length, you must enter the relative path name. The relative path name starts with the name of the target system to indicate the top-level directory and does not include the leading root slash. For example, data/stuff.txt, where "data" is the name of the target system.</p>
Target System	<p>If you plan to map several zFS files under the same zFS directory location, specify a target system to use.</p> <p>You can click Create to add a new path name to use, or if a relative path name is already specified in the Pathname field, you must select an existing target system from the drop-down list.</p> <p>If you choose to create a new target system, complete the following fields and click Finish:</p> <p>Name – Enter the name for the new target system.</p> <p>CCSID – Enter the CCSID of the character set in which the zFS file data is encoded. The default setting is EBCDIC 1047.</p> <p>Base Pathname – Enter the absolute path name under which the zFS file resides. Typically, this is the path name of the zFS subdirectory that contains your zFS file. At runtime, the server will determine the location of the zFS file by concatenating the path name with the value specified in the virtual table Pathname field. The server does not insert additional slash (/) separators when concatenating the target system path name and the virtual table path name. If the target system path name represents a complete directory name, include the trailing slash (/tmp/).</p>

Field	Action
Advanced	When reading large volumes of data from tables, click Advanced to display and configure the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

8. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Creating virtual tables for CA IDMS data

Create virtual tables that map to the CA IDMS data that you want to access and from which the SQL used to access the data is generated and executed.

Before you begin

The Data Virtualization Manager server must be configured for CA IDMS access, and the CA IDMS central version referenced by the data server SYSCTL DD statement must be active.

About this task

CA IDMS schema records are mapped using the CA IDMS data dictionary. Each record is mapped as a separate virtual table using the COBOL names to derive the SQL column names. In addition to records, schema sets can be mapped as well. Virtual tables created for CA IDMS sets serve as correlation tables between CA IDMS records so SQL joins can navigate the CA IDMS schema.

Procedure

1. On the **Server** tab, explore the CA IDMS metadata information by expanding the **Discovery > IDMS** node, and then navigating down the appropriate subtree. The hierarchy begins with the data dictionary, followed by the CA IDMS schema, the CA IDMS subschema, and then the associated records and sets.
2. Select one or more records, as follows:
 - To select individual records, hold down the Ctrl key and click each record to include.
 - To select a range of records, click the first record in the range, and then hold the Shift key and select the last record in the range. All records within the range will be included.
 - To select all child records under a parent, click the parent record.
3. Right-click the selected records and select **Create Virtual Table(s)**. The **New Virtual Tables Wizard** launches.

Note: You can map the relevant CA IDMS sets in the wizard.

4. On the **Create IDMS virtual tables** page, complete the following **Common Virtual Table Settings**:

Field	Description
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Arrays Handling	Select one of the following options:

Field	Description
	<ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime (Y): This option supports both OCCURS and OCCURS DEPENDING ON statements. • Return arrays into separate tables at runtime (N): This option supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables support SQL read access only.
Virtual Table Naming Patterns	<p>Specify the format to use for the generated virtual table names. You can specify different patterns for records and sets. Use the following variables to create naming patterns that are derived from the IDMS metadata:</p> <ul style="list-style-type: none"> • {SubSchema}: Subschema name • {Record}: Record name • {Set}: Set name
Prune IDMS record field suffix from column names	Select this option to remove the IDMS record field suffix from the column names.

5. In the table that lists the IDMS records, review the list of selected entries. Modify the selections as needed.

Tip: Use the check box in the header row of the table to control the selection of all entries.

6. To map the sets, click **Fetch Related IDMS Sets**. The studio collects additional metadata from the server and displays the relevant items in the table that lists the IDMS sets.

7. In the table that lists the IDMS sets, review the list of selected entries. Modify the selections as needed.

8. To disable MapReduce, click **Advanced** and select **Disable MapReduce**.

9. Click **Finish**.

Results

The studio creates the virtual tables (the metadata maps) on the server.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Creating virtual tables for VSAM and sequential access using ADDI

Create virtual tables that map VSAM and sequential data for COBOL applications by using information made available through IBM Application Discovery and Delivery Intelligence (ADDI).

Before you begin

The Data Virtualization Manager server must be configured to access one or more ADDI projects hosted on Microsoft SQL Server. The studio recognizes ADDI when virtual views and target system maps are installed. Map recognition is based on target systems starting with the string TSIAD and virtual views starting with the name IADV_. For more information on configuring the server, see the *Installation and Customization Guide*.

About this task

To create the virtual tables that are used to access VSAM and sequential data for COBOL applications, information is queried in the ADDI project. Information is retrieved about the z/OS data sets and the COBOL copybooks used to access the z/OS data sets.

The following restrictions and considerations apply:

- Virtual table creation is restricted to data sets in the ADDI project that are processed by COBOL programs using JCL. Data sets accessed using CICS as well as other databases (such as IMS, CA IDMS, or Adabas) are not supported.
- When retrieving data sets from the ADDI project, the studio provides a list of all data sets discovered in the ADDI project that correspond to copybook information. If the data set does not have a corresponding copybook, the data set will not be presented in the studio.
- When creating virtual tables in the studio, duplicate records may appear in the generated list. (Duplicate records have the same project and copybook record names but different ID values.) This is due to multiple copies of the same copybook existing in the ADDI project. The studio provides a feature that compares the definitions of the records and allows you to remove any duplicates.
- When mapping COBOL copybooks containing REDEFINES clauses, default mapping rules related to REDEFINES will be applied which will result in disabled columns in the maps. Editing of virtual maps may be required after generation to enable or disable generated columns.
- ADDI project names are limited to 13 characters due to location name restrictions in the z/OS server.

Procedure

1. On the **Server** tab, explore the ADDI metadata information by expanding the **Discovery > IBM Application Discovery** node, and then navigating down the appropriate subtree. The hierarchy begins with the project, followed by the data sets, and then the associated records.
2. Optional: Right-click a record and select **Display Data Layout** to show the copybook for the record.
3. Select one or more data sets or records to map, as follows:
 - To select individual data sets or records, hold down the Ctrl key and click each data set or record to include.
 - To select a range of data sets or records, click the first data set or record in the range, and then hold the Shift key and select the last data set or record in the range. All data sets or records within the range will be included.
 - To select all records under a data set, click the data set.
4. Right-click the selected data sets or records and select **Create Virtual Table(s)**.
The **New Virtual Tables Wizard** launches, presenting a list of proposed virtual table names and the COBOL structure names that will be used as a basis to create columns for the virtual tables.
5. On the **Create virtual tables using IBM Application Discovery** page, complete the following fields:

Field	Description
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Naming Pattern	Specify the format to use for the generated virtual table names. You can specify different patterns for the project name and records. Use the following variables to create naming patterns that are derived from the ADDI metadata: <ul style="list-style-type: none"> • {Project}: ADDI project name • {Record}: Record name
Arrays Handling	Select one of the following options: <ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime (Y): This option supports both OCCURS and OCCURS DEPENDING ON statements.

Field	Description
	<ul style="list-style-type: none"> • Return arrays into separate tables at runtime (N): This option supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables support SQL read access only. • Flatten arrays now (C): If you select this option, you cannot change array-handling after you save the virtual table.

6. In the table that lists the records, review the list of selected entries and perform the following steps:
- Optional: If duplicate target virtual table names appear, which are identified with a description in the **Errors** column, click **Remove Duplicates**.
The studio compares the definitions of the records and removes any duplicates.
 - Click **Validate** to validate each data set and determine the data set type.
The studio populates the **Type** column with the correct data set type.
 - Modify the selections to map as needed.

Tip: Use the check box in the header row of the table to control the selection of all entries.

7. Optional: Click **Advanced** to display and complete the following fields:

Field	Description
MapReduce (Server Parallelism Overrides)	When reading large volumes of data from tables, you can use the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

8. Click **Finish**.

Results

The virtual tables are created on the server and are visible under the **SQL > Data > SSID > Virtual Tables** tree node, where *SSID* is the name of your server.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries”](#) on page 114.

Creating virtual tables for VSAM and sequential access using RAA

Create virtual tables that map VSAM and sequential data for COBOL applications by using information made available through IBM Rational Asset Analyzer (RAA).

Before you begin

The Data Virtualization Manager server must be configured to access one or more RAA database schemas hosted on Db2 for z/OS. The studio recognizes RAA when RAA virtual views and target system maps are installed. Map recognition is based on target systems starting with the string TSRAA and virtual views starting with the name RAAV_. For more information on configuring the server, see the *Installation and Customization Guide*.

The preferred method to collect COBOL information is to retrieve record layouts directly from the WebSphere Application Server that hosts RAA. The WebSphere Application Server must be configured using the Metadata Discovery preferences. For more information, see [“Metadata Discovery preferences”](#) on page 134.

About this task

To create the virtual tables that are used to access VSAM and sequential data for COBOL applications, information is queried in the RAA database and from the host. Information is retrieved about the z/OS data sets and the COBOL copybooks used to access the z/OS data sets. If the WebSphere Application Server has been configured, all access to the host for record layout information will first be attempted using the WebSphere Application Server hosting RAA. If access to the RAA host fails and the record layout is stored in a PDS, layout retrieval will be attempted using the current Data Virtualization Manager server.

The following restrictions and considerations apply:

- Virtual table creation is restricted to data sets in the RAA database that are processed by COBOL programs using JCL. Data sets accessed using CICS as well as other databases (such as IMS, CA IDMS, or Adabas) are not supported.
- When retrieving data sets from the RAA database, the studio provides a list of all data sets discovered in the RAA database that correspond to copybook information. If the data set does not have a corresponding copybook, the data set will not be presented in the studio.
- When creating virtual tables in the studio, duplicate records may appear in the generated list. (Duplicate records have the same database and copybook record names but different ID values.) This is due to multiple copies of the same copybook existing in the RAA database. The studio provides a feature that compares the definitions of the records and allows you to remove any duplicates.
- When mapping COBOL copybooks containing REDEFINES clauses, default mapping rules related to REDEFINES will be applied which will result in disabled columns in the maps. Editing of virtual maps may be required after generation to enable or disable generated columns.

Procedure

1. On the **Server** tab, explore the RAA metadata information by expanding the **Discovery > IBM Rational Asset Analyzer** node, and then navigating down the appropriate subtree. The hierarchy begins with the database, followed by the data sets, and then the associated records.
2. Optional: Right-click a record and select **Display Data Layout** to show the copybook for the record.
3. Select one or more data sets or records to map, as follows:
 - To select individual data sets or records, hold down the Ctrl key and click each data set or record to include.
 - To select a range of data sets or records, click the first data set or record in the range, and then hold the Shift key and select the last data set or record in the range. All data sets or records within the range will be included.
 - To select all records under a data set, click the data set.
4. Right-click the selected data sets or records and select **Create Virtual Table(s)**.
The **New Virtual Tables Wizard** launches, presenting a list of proposed virtual table names and the COBOL structure names that will be used as a basis to create columns for the virtual tables.
5. On the **Create virtual tables using IBM Rational Asset Analyzer** page, complete the following fields:

Field	Description
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, hlq.USER.MAP). The target libraries are specified in the server's started task JCL.
Schema Name	From the list of defined Schemas, select a Schema for the virtual table.
Description	Enter an optional description.
Naming Pattern	Specify the format to use for the generated virtual table names. You can specify different patterns for the database name and records. Use the following variables to create naming patterns that are derived from the RAA metadata: <ul style="list-style-type: none">• {Database}: RAA database name

Field	Description
	<ul style="list-style-type: none"> • {Record}: Record name
Arrays Handling	Select one of the following options: <ul style="list-style-type: none"> • Flatten arrays into a single fixed table at runtime (Y): This option supports both OCCURS and OCCURS DEPENDING ON statements. • Return arrays into separate tables at runtime (N): This option supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables support SQL read access only. • Flatten arrays now (C): If you select this option, you cannot change array-handling after you save the virtual table.

6. In the table that lists the records, review the list of selected entries and perform the following steps:

a) Optional: If duplicate target virtual table names appear, which are identified with a description in the **Errors** column, click **Remove Duplicates**.

The studio compares the definitions of the records and removes any duplicates.

b) Click **Validate** to validate the data set and determine the data set type.

The studio populates the **Type** column with the correct data set type.

c) Modify the selections to map as needed.

Tip: Use the check box in the header row of the table to control the selection of all entries.

7. Optional: Click **Advanced** to display and complete the following fields:

Field	Description
MapReduce (Server Parallelism Overrides)	When reading large volumes of data from tables, you can use the MapReduce feature. The MapReduce feature enables you to divide the data into logical partitions and process those partitions in parallel using the Thread Count value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The Thread Count value you specify overrides the default value (2) and the discovered value. To disable MapReduce , select the Disable MapReduce check box.

8. Click **Finish**.

Results

The virtual tables are created on the server and are visible under the **SQL > Data > SSID > Virtual Tables** tree node, where *SSID* is the name of your server.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See [“Generating and executing SQL queries” on page 114](#).

Creating virtual views

Consider creating a virtual view if columns in your virtual table are missing or if you want to join columns from different virtual tables.

Before you begin

The virtual tables representing the data that you want to access or join must already exist.

About this task

A virtual view comprises the SELECT statement that contains the columns from the source data that are used to read data directly from the data source. For example, `SELECT * FROM HLS_JOIN_VSAM LIMIT`

1000;. In some cases, creating virtual views is more convenient than regenerating and editing SQL each time you want to access the same data.

Procedure

1. In the **Server View**, expand **SQL > Data > SSID > Virtual Tables**.
2. Right-click the virtual table that represents the data that you want to access, and select **Create Virtual View**.
3. In the **Name** field, enter a name for the virtual view.
4. From the **Target** drop-down list, select the target to use for this virtual view.
5. Optional: In the **Description** field, enter a description.
6. Click **Next**.
7. In the **Table Browser**, expand the **Virtual Tables** folder, and select an existing virtual table to use to compose the SQL statement.
8. Click **Next**.
9. Optional: Review the resulting SQL statement and make any necessary modifications.
10. Click **Validate** to validate the SQL.
11. If valid, on the **SQL Validation** message that displays, click **OK**.
12. Click **Finish**.

Results

In the **Server** view, locate the new virtual view by expanding **SQL > Data > SSID > Virtual Views**.

Viewing copybook member name in metadata

The details of the copybook member name used during Virtual Table (VT) creation is captured and saved to the DMF map XML file.

Procedure

To view the copybook member name:

1. Click **Window->Show View->Virtualization Facility**
2. In the **Virtualization Facility** pane, navigate to the VT for which you want to know the copybook member name and double click it.

The `<INPUT from=" field .."/>` contains the copybook member name that is used for VT creation.

Creating Db2 user-defined table functions

Use the **New UDTF Definitions in DB2 Wizard** to create user-defined table functions (UDTFs) in Db2 for z/OS for access to any supported data source type.

Before you begin

Db2 Virtualization (DB2V) is a feature that provides single-point access to various data source types. For additional information about configuring your system to use Db2 for z/OS as a primary access point, see [Using Db2 for z/OS to access multiple data source types](#).

To use this wizard, virtual tables should already exist for your data sources. See [“Creating virtual tables” on page 79](#).

About this task

Use the **New UDTF Definitions in DB2 Wizard** to create the necessary user-defined table functions and views in Db2 for z/OS for access to any supported data source type. For existing virtual tables, this wizard creates the necessary objects in a local Db2 subsystem so that Data Virtualization Manager data can be queried using Db2 clients.

Procedure

1. Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.
2. In the **Virtual Tables** node, right-click one or more virtual tables, and select **Create UDTF Definitions in DB2**.
3. In the **New UDTF Definitions in DB2 Wizard**, on the **Generate DDL with user-defined table functions** page, complete the following fields:

Field	Action
General DB2 Settings	<p>Specify information about the Db2 subsystem where the UDTFs and views will be created.</p> <ul style="list-style-type: none"> • Subsystem: Select the Db2 subsystem ID from the drop-down list. • Schema: Select the schema from the drop-down list, or enter a new schema name. • GRANT TO: Specify to whom privileges are granted for the generated UDTFs and views. Clear this field to not include the GRANT TO statement in the generated DDL. • UDTF Module: This value defaults to the UDTF module in use for your system. If you need to change this value, enter the name of another UDTF module. The following modules are available: <ul style="list-style-type: none"> – AVZUDT9N – AVZUDTAN – AVZUDTBN – AVZUDTCN <p>For more information, see "Using Db2 for z/OS to access multiple data source types" in the <i>Installation and Customization Guide</i>.</p> <ul style="list-style-type: none"> • WLM Environment: The address space Db2 starts to run user-defined functions. Leave this field blank to omit the WLM ENVIRONMENT clause in the generated DDL.
"Generate" Actions	<p>Specify whether to execute or save the DDL, or both.</p> <ul style="list-style-type: none"> • Execute generated DDL in DB2: Select this option to execute the generated DDL on the specified Db2 subsystem. • Save DDL to file: Optionally, enter a file name (with .sql extension) where to save the DDL. This step might be required if you do not have authorization to execute the DDL. Or, you might want to review the DDL in the SQL Editor first, before running it in Db2. <ul style="list-style-type: none"> – Append to file: Select to append the generated DDL to an existing file. – Open file in SQL Editor: Select to open the generated DDL in the SQL Editor.
Naming Patterns	<p>Specify the format to use for the generated function and view names. Use the following variables to create naming patterns:</p> <ul style="list-style-type: none"> • {Server}: Data Virtualization Manager server name • {Table}: Virtual table name <p>The Function Name and View Name columns are editable in the table, so you can also customize the names on an individual basis.</p> <p>Note: If Views is blank, views will not be generated.</p>

4. Click **Generate**.

Results

The DDL for creating UDTFs and corresponding views is generated. For more information, see [“Generated DDL for UDTFs”](#) on page 108.

After the DDL is executed, a UDTF and a corresponding view are created for each selected virtual table. In the **Server** tab, locate the new objects by expanding **SQL > Data > Other Subsystems > db2SSID > schema**, and then the appropriate nodes, as follows:

- The Db2 views are located in the **Views** node.
- The Db2 UDTFs are located in the **DB2V > User-Defined Table Functions** node.

What to do next

After the Db2 views and UDTFs have been created, you can perform the following tasks:

- Using the new Db2 views, compose and execute SQL queries to access the data. You can do this from the Data Virtualization Manager studio or from a Db2 client.
- Using the new Db2 UDTFs, compose and execute SQL queries to select data from the virtual table. Queries can be generated for a specific UDTF function or for a subset of columns within a UDTF.

See [“Generating and executing SQL queries” on page 114](#).

Generated DDL for UDTFs

This topic describes the DDL that is generated by the **New UDTF Definitions in DB2 Wizard** in the Data Virtualization Manager studio when creating Db2 UDTFs and corresponding views for virtual tables.

For each Data Virtualization Manager map created as a view in Db2, there are two DDL statements defining catalog objects to Db2. The first statement is a CREATE FUNCTION statement, which describes the user-defined function to retrieve data from Data Virtualization Manager. The second statement is a CREATE VIEW statement, which calls the user-defined table function.

Example: CREATE FUNCTION

The following example shows the Db2 DDL generated to define the user-defined table function for the STAFFVS table:

```
CREATE FUNCTION "TSUSER"."AVZS_STAFFVS"  
(CONDITION VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,  
AVZNAME VARCHAR(254) CCSID EBCDIC FOR SBCS DATA,  
COLINFO VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,  
REQUEST VARCHAR(254) CCSID EBCDIC FOR SBCS DATA)  
RETURNS TABLE  
( "STAFFVS_KEY_ID" SMALLINT,  
  "STAFFVS_DATA_NAME_L" SMALLINT,  
  "STAFFVS_DATA_NAME" CHAR(9),  
  "STAFFVS_DATA_DEPT" SMALLINT,  
  "STAFFVS_DATA_JOB" CHAR(5),  
  "STAFFVS_DATA_YRS" SMALLINT,  
  "STAFFVS_DATA_FILLER" CHAR(8))  
EXTERNAL NAME "AVZUDTCN"  
LANGUAGE ASSEMBLE  
PARAMETER STYLE DB2SQL  
DETERMINISTIC  
FENCED  
NO SQL  
SCRATCHPAD 2048  
FINAL CALL  
DISALLOW PARALLEL  
DBINFO WLM ENVIRONMENT DSN1WLM1  
STAY RESIDENT YES  
ASUTIME NO LIMIT  
SECURITY USER;
```

Note the following about the UDTF DDL:

- The following parameters are common to the generic table function:

```
(CONDITION VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,  
AVZNAME VARCHAR(254) CCSID EBCDIC FOR SBCS DATA,
```

```
COLINFO VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,  
REQUEST VARCHAR(254) CCSID EBCDIC FOR SBCS DATA)
```

These parameters are required on every UDTF definition for a Data Virtualization Manager virtual table referenced by a view in Db2. For more information about these parameters, see [“UDTF parameters” on page 109](#).

- The RETURNS TABLE clause describes the columns in the Data Virtualization Manager table as defined in the map.

Warning: These columns must match exactly the map definition to prevent errors as the UDTF does not have access to this information at invocation time and therefore does no SQL type conversion.

Note that the system parameter **SQLENGCHARTOVARCHAR** changes column definitions and must be considered when defining the UDTF. Because the RETURNS TABLE clause describes the return table data to Db2, each map requires a separate UDTF definition. While tables having identical columns can technically share the same definition, this practice is not recommended unless the table is from a common map shared by multiple Data Virtualization Manager servers.

- The following DDL elements contain information you can customize at view creation time. Considerations for this information are as follows:
 - CREATE FUNCTION "TSUSER"."AVZS_STAFFVS" – Like most DB2 catalog objects, function names include both a schema name and a function name. The studio wizard provides flexibility in both the schema and function naming, including a default pattern for function names.
 - EXTERNAL NAME "AVZUDTCN" – The external name will always be in the form AVZUDTaN, where a is the z/OS architecture level (9, A, B, C).
 - WLM ENVIRONMENT DSN1WLM1 – Identifies the name of the address space that Db2 starts for running user-defined functions. A reasonable default is the Db2 subsystem ID suffixed with WLM1, which the user can change if necessary. This element is optional, and when omitted, the default WLM environment for that Db2 subsystem will be chosen at runtime.

Example: CREATE VIEW

The following example shows the Db2 DDL generated to define the view to call the user-defined table function:

```
CREATE VIEW "TSUSER"."AVZS_VSTAFFVS" AS  
SELECT  
"STAFFVS_KEY_ID", "STAFFVS_DATA_NAME_L", "STAFFVS_DATA_NAME", "STAFFVS_DATA_DEPT",  
"STAFFVS_DATA_JOB", "STAFFVS_DATA_YRS", "STAFFVS_DATA_FILLER"  
FROM TABLE("TSUSER"."AVZS_STAFFVS"  
(  
'AVZS...STAFFVS',  
'7', STAFFVS_KEY_ID, STAFFVS_DATA_NAME_L, STAFFVS_DATA_NAME, STAFFVS_DATA_DEPT, ' ||  
'STAFFVS_DATA_JOB, STAFFVS_DATA_YRS, STAFFVS_DATA_FILLER',  
' )  
CARDINALITY 10000);
```

The CARDINALITY clause value is controlled by the **DB2 CARDINALITY in generated UDTF query** setting in [“SQL preferences” on page 133](#).

UDTF parameters

The following table describes the parameters that must be passed when calling the UDTF.

Parameter	Description
CONDITION (' ')	<p>The CONDITION parameter can be used to add a WHERE predicate to the generated SQL sent to the Data Virtualization Manager server. For example, if you want to create a Db2 view that returns all managers using the STAFFVS example, you could specify 'WHERE STAFFVS_DATA_JOB = ' 'MGR' ' ' in the CONDITION parameter. Generally, this value will be defined as ' ' ' .</p> <p>Note: By using the WHERE predicate on the UDTF call in the CONDITION parameter, the result set is filtered on the UDTF call. If you use the WHERE predicate when querying the view instead of passing it as a CONDITION parameter, all results will be returned on the UDTF call first and then filtered on the view call. This might affect performance.</p>
AVZNAME ('AVZS...STAFFVS')	<p>The AVZNAME is a four-part period-separated name in the form <i>dddd.bbbb.ssssssss.vvvvvvvv</i> where:</p> <ul style="list-style-type: none"> • <i>dddd</i> – The 4-character subsystem name or 5-8 character group name for the local Data Virtualization Manager server. If the name is greater than 4 characters, the token is assumed to be a group name. • <i>bbbb</i> – The 4-character subsystem name if the table is in another Db2 subsystem. Generally, this token will be omitted. • <i>ssssssss</i> – The schema name for the table in the Data Virtualization Manager server. This token is for future use and should be omitted. • <i>vvvvvvvv</i> – The virtual table name in the Data Virtualization Manager server.
COLINFO ('7,STAFFVS_KEY_ID,...')	<p>Describes the column count and optionally the Data Virtualization Manager virtual table column name list to the table function. Minimally, this must include the number of columns in the return table (for example, '7'). The form shown in the example includes a comma-separated list of every column in the table in COLNO order. If provided, this list will be used to generate optimized queries when the Db2 user queries a subset of columns in the view definition.</p> <p>Warning: It is critical that this count matches the number of columns included in the RETURNS TABLE clause of the CREATE FUNCTION DDL.</p>
REQUEST (' ')	<p>Specifies runtime options. See the next section, “REQUEST runtime options” on page 111</p>

REQUEST runtime options

Runtime options can be passed as comma-separated values in the REQUEST parameter. In most cases, these values are used to diagnose problems with the UDTF and should be added under the direction of contact IBM Software Support.. The following runtime options can be added to the REQUEST parameter to control execution of the table function:

Option	Description
GROUPNAME	Instructs the table function to use the first part of AVZNAME as a Data Virtualization Manager server group name instead of a Data Virtualization Manager subsystem ID. It is only needed if the first token is 4 or less characters in length, but can also be included for documentation purposes if desired (for example, 'GROUPNAME').
MRC(n)	Enables MRC or MRCC processing. When specified without MRID, the UDTF will create multiple MRC connections to the Data Virtualization Manager server and partition row data across connections based on the map/reduce partitioning algorithm for the underlying target database or data set. When specified with MRID, MRC will act as a single participant in an MRCC request (for example, 'MRC(6),MRID(1)').
MRID(n)	Used in conjunction with MRC to identify the map/reduce participant ID associated with a view in MRCC request environments. Db2 views set up with MRC and MRID will read a subset of the virtual table data based on the map/reduce partitioning algorithm for the underlying target database or data set (for example, 'MRC(6),MRID(1)')
TRACE()	Instructs the UDTF to send trace information to the Data Virtualization Manager server after a successful connection is open to the server. In the server trace, all trace information is enclosed between the XML tokens <DVUDFT_TRACE> and </DVUDFT_TRACE>. The following comma-separated sub-options can be specified within the TRACE() option. <ul style="list-style-type: none"> • BUILDINFO – Displays the build date and architecture level of the UDTF program. This trace is issued immediately after a successful connection is established to the Data Virtualization Manager server (for example, 'TRACE(BUILDINFO)'). • CLOSE – Displays a message when a UDTF query is closed (for example, 'TRACE(CLOSE)'). • SQL – Displays the generated SQL sent to the Data Virtualization Manager server in response to the UDTF call from Db2 (for example, 'TRACE(SQL)'). • STATS – Displays a summary of the statistics at query close time, including rows retrieved and producer/consumer wait times (for example, 'TRACE(STATS)').
VPIO(n)	When used with VPDNAME, VPIO sets the number of I/O threads for a VPD group (for example, 'VPNAME(VPG1),VPNO(6),VPIO(3)').
VPNAME(name)	Enables VPD processing by defining a VPD group name (for example, 'VPNAME(VPG1),VPNO(6),VPIO(3)').
VPNO(n)	Sets the number of members in a VPD group (for example, 'VPNAME(VPG1),VPNO(6),VPIO(3)').
VPTO(n)	Sets the timeout value in seconds for a VPD group (for example, 'VPNAME(VPG1),VPNO(6),VPIO(3),VPTO(10)').

UDTF generated query example

The following example shows a query of a Db2 UDTF, generated in the Data Virtualization Manager studio, where a subset of the virtual table columns have been selected:

```
-- Description: Retrieve the result set for AVZS_STAFFVS
-- Tree Location: rs99/46000/SQL/Data/Other Subsystems/DSN1/TSUSER/DB2V/
User-Defined Table Functions/AVZS_STAFFVS
-- Remarks: DB2V:AVZS...STAFFVS
SELECT "STAFFVS_DATA_NAME", "STAFFVS_DATA_DEPT", "STAFFVS_DATA_JOB", "STAFFVS_DATA_YRS"
FROM TABLE("TSUSER"."AVZS_STAFFVS"
('',
'AVZS...STAFFVS',
'7,STAFFVS_KEY_ID,STAFFVS_DATA_NAME_L,STAFFVS_DATA_NAME,STAFFVS_DATA_DEPT,' ||
'STAFFVS_DATA_JOB,STAFFVS_DATA_YRS',
''))
CARDINALITY 10000);
```

You can then modify the UDTF arguments in the generated SQL, such as the following options:

- Use the `CONDITION` parameter to add a `WHERE` predicate
- Use the `TRACE` parameter to send trace information to the server

The following example shows these modifications:

```
SELECT "STAFFVS_DATA_NAME", "STAFFVS_DATA_DEPT", "STAFFVS_DATA_JOB", "STAFFVS_DATA_YRS"
FROM TABLE("TSUSER"."AVZS_STAFFVS"
('WHERE STAFFVS_DATA_YRS > 5',
'AVZS...STAFFVS',
'7,STAFFVS_KEY_ID,STAFFVS_DATA_NAME_L,STAFFVS_DATA_NAME,STAFFVS_DATA_DEPT,' ||
'STAFFVS_DATA_JOB,STAFFVS_DATA_YRS',
'TRACE(BUILDINFO,SQL,STATS)')
CARDINALITY 10000);
```

Db2 federation nicknames for distributed environment

Data Virtualization Manager studio supports the integration and creation of Db2 federation nicknames for virtual tables in a Db2 distributed environment.

The Db2 LUW 11.5.4 installer includes the JDBC driver as a part of the installation. This simplifies the process of plugging into various IBM offerings; Db2 database family, IBM Cloud Pack for Data, IBM Cloud Pack for Data as a service, and stand-alone IBM public Cloud offerings. To know more about nicknames in the federated system, refer [working with nicknames](#).

This functionality:

- Connects to any Db2 Linux based environment that is configured with Data Virtualization Manager.
- Supports insert/update/delete operations on nicknames.
- Supports the use of predicates on the nicknames or views.
- Supports the creation of multiple nicknames using parallelism (MRC).

Creating nicknames

Create a Db2 federation nickname to enable the Db2 views created on Data Virtualization Manager virtual tables query data in a Data Virtualization Manager server using the JDBC driver.

Before you begin

Ensure that the following tasks are completed before configuring a nickname:

- Configure the Db2 Linux instance with Data Virtualization Manager by adding the required details in the IN00 file.
- Link the credentials of the Db2 Linux instance with the specific TSO ID.

- Enable the **AVZELUWG** rule to enable the Data Virtualization Manager server to connect to the Db2 Linux instance.
- In case of Db2 Linux version that is earlier than 11.5.4, ensure that the JDBC driver jar and the corresponding log jar files are added to the classpath. Use the echo \$CLASSPATH command to view the value of the **CLASSPATH** variable in a Linux environment.

Procedure

1. Select the virtual table and right-click and select **Create Db2 distributed Federation Definition**.
2. Enter the following details in **New Nickname Definitions in DB2 Wizard**.

- **DB2 System:** Select the Db2 LUW system on which the nicknames and view are created.
- **Schema:** Select the schema from the drop down list. The schema names are populated from the selected Db2 system.
- **DB2 User:** Enter the name of the Db2 user. This field allows the Db2 user to create nicknames under a different schema.

Note: The user must have appropriate permissions to perform these actions.

- **Parallelism:** Set the value for MRC parameter of the JDBC driver. This field determines the number of threads executed in parallel based on the specified value. The value entered is equal to the number of nicknames that are created in the Db2 LUW system. The default value is 4.
- **Execute generated DDL in Db2:** By default, this option is selected. If this option is selected, the generated script is immediately executed against the Db2 LUW system. If this option is not selected, the script is generated but not executed. To manually execute this script, right-click the script and select the **Execute SQL** option.
- **Save DDL to file:** Specify the file name (a suffix of .sql) for the generated script.
 - **Append to file:** If this option is selected and the file name that is mentioned by the user already exists in the local machine, the generated script code is appended to the existing file. If this option is not selected, a new file is created containing the generated script code on the local machine.
 - **Open file in SQL Editor:** By default, this option is selected. Selecting this option makes the generated script file to open in the SQL Editor of the Data Virtualization Manager Studio.
- **Default Options:** This field contains the default set of options that are needed to configure the federation server connection. This field cannot be edited.
- **Additional Options:** Additional options can be added in the format *parameter_name1 parameter_value1, parameter_name2 parameter_value2* and so on. For example:

```
DB2_MAXIMAL_PUSHDOWN 'Y', collating_sequence 'Y'
```

- **JDBC URL:** Specifies the JDBC connection string that is used to connect to the Data Virtualization Manager Server from the Federation server. This field is populated with the current active connection.
 - **Driver Package Path:** Specifies the path where the JDBC driver and log jars are located in the selected Db2 LUW machine.
 - **Nicknames:** Specifies the naming pattern for the nicknames to be created.
 - **View:** Specifies the naming pattern of the view that concatenates all the nicknames.
3. Click **Generate**.

After the script is executed, it is displayed on the **SQL Messages** section of the **SQL Results** tab. You can modify the script, if required, and execute it again.

Validating SQL statements

Data Virtualization Manager studio allows you to validate SQL queries before you execute them. You can select one or more SQL queries for validation.

About this task

You can validate the following types of SQL queries:

- SELECT queries
- INSERT, UPDATE, and DELETE queries
- Queries with parameter markers
- Queries with unnamed column

Data Virtualization Manager studio performs the following when validating the SQL queries:

- Checks syntax accuracy.
- Checks if the table name is valid. If invalid, the error message **Error: Unable to process map *table_name*** is displayed.
- Checks if the column name is valid. If invalid, the error message **Error: Invalid column reference *column_name* in SQL** is displayed.

You can select one or more SQL queries for validation. To validate the SQL queries, perform the following steps in the SQL editor of the studio.

Note: Parameterized INSERT and UPDATE to a Db2 table's Graphic column on the Z system with UTF8 (IBM1208) is not supported.

Procedure

1. Select the SQL query or SQL queries that you wish to validate.
2. Right-click the selection and select **Validate SQL** from the context menu that appears.

Results

If the validation is successful, **Validation Succeeded** message is displayed. Else, appropriate error messages are displayed.

Generating and executing SQL queries

To test SQL access to your data, generate and execute a SQL query from an existing virtual table or virtual view.

Before you begin

To avoid fetching large result sets that are memory intensive, the Data Virtualization Manager studio provides settings related to SQL generation and retrieval that can limit the amount of data that is actually retrieved for a particular query execution. For more information, see [“SQL preferences” on page 133](#).

Important: When writing SQL to access Adabas data, use caution when using the BASE_KEY in WHERE predicates, (for example, [PARENT TABLE].BASE_KEY = [CHILD TABLE].PARENT_KEY) when joining the parent table with a child subtable, since this will result in a table scan of the entire Adabas file. It is recommended instead to use the CHILD_KEY (for example, [PARENT TABLE].CHILD_KEY = [CHILD TABLE].PARENT_KEY).

Procedure

1. On the **Server** tab, right-click a virtual table and select **Generate Query**.
2. Choose from the following options:

- **Yes** – Generate the SQL query in the **Data Source Editor** and execute the query.
 - **No** – Generate the SQL query in the **SQL Editor** without executing the query. The generated SQL SELECT statement has all columns from the selected table. If the table contains a large number of columns, to avoid enumerating the various column names you can choose all columns using the **Generate Query with *** option.
3. Optional: In the **SQL Editor** view, modify the SQL to select only the data that you want to access. Any ANSI compliant SQL is acceptable.
 4. To view or test the data that the SQL statement returns, right-click the highlighted SELECT statement and click either **Execute SQL** to view results in the **SQL Results** view, or **Execute SQL and File results** to save the results in a .csv file.
 5. Optional: To create a virtual view of the SQL, highlight the SELECT statement, right-click and select **Create a virtual view**.

Results

In the **SQL Results** view:

- Double-click a row to view additional details about that row.
- Select the **Export Result Set** view option to export the SQL results to a .csv file.
- Click **SQL Messages** to view query-related system messages.

By default, if a result set includes 25 or more columns, each set of 25 columns are displayed incrementally as groups. You can choose which group you want to view using the **Columns Group** field. You can set the number of columns that you want to include in each group, ranging from 25-200, in the **Columns per group** field.

To change how SQL results display in the **SQL Results** view, see [“Data Virtualization Manager preferences”](#) on page 130.

What to do next

After the SQL statement is generated, you can perform any of the following tasks:

- Modify the SQL to meet your needs
- Execute the SQL to test and view the resulting data
- Create virtual views to join data or include missing columns
- Generate a SQL class to get access to data from your programs or applications

Generating code from SQL

Use the **Code** wizard to generate the code that is used to get SQL access to data from your programs or applications.

Before you begin

The virtual table or virtual view that maps to the data that you want to access must already exist on the server.

Procedure

1. To launch the **Code** wizard from the **Server** tab, right-click the virtual table or virtual view and select **Generate Code From SQL**.
Alternatively, to compose and execute your SQL query directly from a selected SQL statement in the editor, right-click on the selected SQL statement and select **Generate Code From SQL**.
2. On the **SQL** page, accept or change the file name that will be used to store the generated code.
3. Select from the following query options and click **Next**:
 - **Use Selected View** – Creates a simple query. This is the default setting.

- **Compose the SQL Statement** – Selects all table columns and displays the resulting SQL statement. If necessary, you can choose to modify the resulting SQL statement before continuing to the next step.
4. On the **SQL Result Set** page, review the result set and optionally rename variables, and click **Next**.
 5. On the **Code Generation** page, choose the programming language that you want to use to generate the SQL:

- **Java JDBC Class**
- **Java Spark Application**
- **Scala Jupyter Notebook**
- **Scala Spark Application**
- **Python Jupyter Notebook**
- **VB.NET Class**
- **C# Class**
- **Other (Specify XSLT file)**

Note: The generated Scala Jupyter Notebook code provides a simple starter program showing you how to use the JDBC driver to load data into a Spark application. The wizard does not allow SQL parameter markers for this application type.

6. To finish generating the code, complete one of the following options:
 - If you did not choose the **Scala Jupyter Notebook** option, click **Finish**.
 - If you did chose the **Scala Jupyter Notebook** option, complete the following fields and click **Finish**:

Field	Action
Jupyter Kernel Name	Enter the name of the kernel to use.
Include additional Jars in the Generated Code	Select to include additional JAR files when generating the code. This setting is optional and is only available after you set the Generate Code preferences to include additional Jar files.
Leave credentials blank (use IOCTL)	Select to use the IOCTL command call.
Use the pandas API - Only for Python Notebook	If you select Python Notebook, the dialog displays this option that let's you use the pandas API.
Credentials processing in generated code	Select from the following credential processing options: <ul style="list-style-type: none"> • Omit password text – the generated code will contain *** for the password variable, and will need to be updated manually. • Use password text – the password is included in the generated code in hashed format. • Use INI file – if this option is chosen, you must also specify the INI data set name to indicate that the DSN section in the INI file contains the user and password settings. Click Sample to generate and reference a sample INI file on your local system. This file must be available on the host where Jupyter Spark is running.
Preferences	Click Preferences to review the default settings for code generation.

Results

The generated code is saved in your workspace and is accessible from the studio **Client** tab. The resulting file opens and can be modified in the editor if the **Scala Jupyter Notebook** option was not selected. Otherwise, the resulting file can be uploaded to Jupyter using the Jupyter Web UI.

Updating the IMS child segments

This section describes the conditions when you can insert, update or delete a child segment using the RECORD_ID and the PARENT_ID parameters, and contains the information on SEGMENT_RBA and PARENT_RBA columns

RECORD_ID: This field is the key feedback area from the DLI call for a target segment, and contains the sequence field information for all segments from the root segment to the target segment accessed. Note that if any segment in the path does not have a sequence field, identifying information for those segments is not included.

PARENT_ID: This field is the hex form of the RECORD_ID for a parent segment, and does not include the sequence field of the target segment.

CHILD_ID: This field contains the hex key to the child record if it has a child record.

A child segment can be inserted only if a unique sequence field is present in the child segment's direct parent and all parents in the upward hierarchical path to the root (including the root). If any segment in the hierarchical path does not have a unique sequence field, unambiguously positioning on the direct parent segment for the segment to be inserted is not possible and the segment cannot be inserted.

While inserting a child segment, either the RECORD_ID or the PARENT_ID parameter must be provided as one of the insert values. This column is used to position on the correct parent segment in the database while inserting the new child segment. The RECORD_ID parameter can only be used if all sequence fields up the hierarchical path contain character data (i.e., no integers or packed fields).

The following example shows insertion of child segments using the RECORD_ID parameter.

```
INSERT INTO PART_DI21PART_STAININFO(STANKEY, PROCUREMENT_CODE, MAKE_SPAN, RECORD_ID)
VALUES('13', '09', 100, '02AN960C1 ');
```

The following example shows insertion of child segments using the PARENT_ID parameter.

```
INSERT INTO PART_DI21PART_STAININFO(STANKEY, PROCUREMENT_CODE, MAKE_SPAN, PARENT_ID)
VALUES('13', '09', 100, 'F0F2C1D5F9F6F0C3F1F0404040404040');
```

The RECORD_ID parameter can be used to qualify the segment instance that is to be updated or deleted only if all the sequence fields (i.e., from the root segment down to and including the target segment) are of type character. If not, the PARENT_ID must be used to qualify the parent of the segment and the key of the segment to update must also be provided in the WHERE clause.

The following example shows deletion of elements using the RECORD_ID parameter.

```
DELETE FROM PART_DI21PART_STAININFO WHERE
RECORD_ID = '02AN960C10 08'
```

The following example shows deletion of elements using the PARENT_ID parameter.

```
DELETE FROM PART_DI21PART_STAININFO WHERE
PARENT_ID = 'F0F2C1D5F9F6F0C3F1F0404040404040' AND STANKEY = '08'
```

SEGMENT_RBA and **PARENT_RBA** columns:

PARENT_ID and CHILD_ID columns created for IMS virtual tables are based on the information available in the PCB key feedback area. Segments that are descendants of parents with no sequence fields or duplicate sequence fields do not contain enough information to perform SQL joins between parent and child segments. To support join operations in these situations, the columns SEGMENT_RBA and PARENT_RBA can be optionally generated as a BIGINT value containing the physical address of a segment and its parent in the database.

The SEGMENT_RBA column is added to all IMS-Direct maps and the PARENT_RBA column is added to all maps for non-root IMS segments. The value in these columns is the RBA position of the prefix for the segment (and parent segment) in the VSAM/OSAM dataset containing the segment data.

If DBCTL is selected to access the data instead of IMS-Direct, these columns return 0.

The SEGMENT_RBA and the PARENT_RBA columns are added to IMS-Direct tables after PARENT_ID and CHILD_ID columns are added during system startup or map refresh time if the following conditions are met:

1. The virtual table is defined for IMS-Direct access.
2. The system parameter **SQLGENIMSDIRBACOLS** is set to YES in the IN00 file.

Parameter name	Parameter description	Default value	Update	Output only
SQLGENIMSDIRBACOLS	SQL GEN IMS-DIRECT RBA COLUMNS. For IMS-DIRECT enabled virtual tables, this parameter generates SEGMENT-RBA or PARENT-RBA BIGINT columns to return the physical RBA positions of the source segment and its parent.		YES	NO

Note: IMS data update via a secondary index is not supported unless they index the root and the data used for the index comes from the root.

Accessing IT Operational Analytics data

To access, analyze, and report IT Operational Analytics (ITOA) data, generate the SQL from ITOA virtual tables.

When you configure the Data Virtualization Manager server, you have the option to include pre-defined data maps that administrators can use to access the following types of ITOA data:

- IBM System Management Facilities files (SMF)
- Operations Log files (OPERLOG_SYSLOG)
- System Log files (SYSLOG)

After you have configured the Data Virtualization Manager server to use ITOA pre-defined data maps, you can generate the SQL that is used to access ITOA data from the ITOA virtual tables.

For information about configuring access to operational analytics data with pre-defined data maps, see "Configuring access to SMF data for IT Operational Analytics" in the *Installation and Customization Guide*.

Accessing SMF data

Use SMF virtual tables to get SQL access to data in System Management Files (SMF).

About this task

When accessing data in SMF files, you use predefined virtual columns that are defined in the SMF virtual table map.

When using SMF log streams, you can use the following virtual columns to retrieve timestamp values:

LS_TIMESTAMP

Timestamp for log stream in GMT. When used in a WHERE predicate, the timestamp is searched in GMT.

LS_TIMESTAMP_LOCAL

Timestamp for log stream in local time zone. When used in a WHERE predicate, the timestamp is searched as local time.

To get SQL access to SMF data, complete the procedure that follows.

Procedure

1. From the Server view, expand **SQL > Data > SSID > Virtual Tables**.

2. Right-click the SMF virtual table or view from which you want to access the data.
3. Right-click **Generate Query**, and then review the resulting SQL statement. If necessary, you can modify the statement to meet your needs. The following shows a sample SQL statement:

```

----- Name           : SMF_000000
-- This statement will return all rows and all columns from the
-- following table:
-- Name           : SMF_000000 : null
-- Catalog        : null
-- Schema         : DVSQL
-- Remarks        : DATA - SMFDATA
-- Tree Location: rs28/1200/SQL/Data/VDBS/Virtual Tables/SMF_000000
-- The sql statement:
SELECT SMF_LEN, SMF_ZERO, SMF_FLAG, SMF_RTY, SMF_TIME, SMF_SID, SMF_SSI,
       SMF_STY, SMF_SEQN, SMF0JWT, SMF0BUF, SMF0VST, SMF0OPT, SMF0RST, SMF0RSV,
       SMF0OSL, SMF0SYN, SMF0SYP, SMF0TZ, SMF0MSWT, SMF0MTWT
FROM SMF_000000 LIMIT 1000;

```

4. Optional: Execute the SQL statement to view, test, or save the resulting data.

What to do next

Get the code to use in your programs and applications by creating a SQL class from the virtual table.

Viewing documentation

Use the **Browse Documentation** option to view the detailed documentation available for the SMF virtual tables, columns in the SMF virtual tables, the SMF sub-tables, and the columns in the SMF sub-tables.

Before you begin

Ensure that the server started task PROC has the following DD name:

```
//AVZSHDOC DD DISP=SHR,DSN=h1q.SAVZSHDOC
```

About this task

The following steps will explain how to view the documentation for the SMF virtual tables, columns in the SMF virtual tables, the SMF sub-tables, and columns in the SMF sub-tables.

Procedure

Expand the **SQL > Data > SSID** node, where *SSID* is the name of your server.

To view documentation for the SMF virtual table.

- Select the SMF **Virtual Table**, right-click the virtual table, and then click **Browse Documentation** to view the documentation.

To view documentation for the columns in the SMF virtual table.

- Expand the SMF virtual table, right-click the required column and then select **Browse Documentation**.

To view the documentation for the sub-tables.

- Right-click the sub-table and then click **Browse Documentation**.

To view the documentation for the columns in the SMF sub-table.

- Expand the sub-table, right-click the required field and then select **Browse Documentation**.

To view the documentation for the SMF virtual table where **Browse Documentation** feature is not available.

- a) Right-click the SMF virtual table where **Browse Documentation** feature is enabled and then click **Browse Documentation**.

The documentation page appears.

- b) Click **Documentation Index**.

- List of all the SMF virtual tables along with the links to their respective documentation appears.
- c) Select the SMF virtual table for which you want to view the documentation.

Accessing Db2 unload data

Using existing Db2 virtual table definitions, you can issue SQL queries against your Db2 sequential unload data sets.

Before you can access your Db2 unload data using your Db2 virtual tables, you must configure access to the Db2 sequential unload data set. This access is configured using a virtual table rule. VTB rule AVZMDLDU is provided to demonstrate redirecting a Db2 virtual table to a Db2 unload data set. For information about setting up access, see "Configuring access to Db2 unload data sets" in the *Installation and Customization Guide*.

After you have performed the configuration steps, you can generate the SQL that is used to access the Db2 unload data using your existing Db2 virtual tables.

As an example, consider a virtual table named DSNA_EMPLOYEES that maps the EMPLOYEES table in Db2 subsystem DSNA. With the virtual table rule that specifies the Db2 unload data set enabled, you can query an unload sequential dataset named EMPLOYEE.UNLOAD.SEQ by issuing the following query:

```
SELECT * FROM MDLDU_DSNA_EMPLOYEES__EMPLOYEE_UNLOAD_SEQ
```

The rule performs the necessary steps to access the unload data set directly.

The following restrictions and considerations apply when using this feature:

- SQL access to Db2 unload files is limited to SQL queries only.
- The columns in the Db2 virtual table definition must exactly match the table unloaded in Db2.

Creating RESTful services

Use IBM z/OS Connect Enterprise Edition and Data Virtualization Manager Web Services to fully leverage the value of your mainframe data without requiring application developers to be familiar with mainframe systems. Data integration using Web Services makes your mainframe data accessible to new digital technologies through fast, simple, and secure APIs.

You can use the Data Virtualization Manager studio to create RESTful services to access all Db2 objects such as Db2 tables, views, stored procedures, and user-defined table functions (UDTFs). When a RESTful API requests mainframe data, z/OS Connect communicates the request to the Data Virtualization Manager server using the WebSphere Optimized Local Adapter (WOLA). The Data Virtualization Manager server executes the requested Web Service to get and return the data to z/OS Connect as objects.

This environment includes the following components:

- Data Virtualization Manager server – Provides mainframe data access to RESTful APIs through z/OS Connect. The Data Virtualization Manager server executes the specified Web Service to get the requested mainframe data and converts and returns that data as objects.
- IBM Data Virtualization Manager studio– Use the **z/OS Connect Configuration Wizard** to generate the Data Virtualization configuration and connection settings that will be included in the z/OS Connect Server.xml file. Use the **Web Services** wizard to create a Web Service and to define the operations required to get the mainframe data.
- z/OS Connect – Enables RESTful API access to existing backend z/OS mainframe services and applications. Routes the data processing requests from a mapped URL of a specific application to the Data Virtualization Manager server.
- WOLA – Provides external access to the internal local communications component of the WebSphere server.
- IBM WebSphere (web server) – Interfaces with remote applications to get data requests over HTTP, REST, and TCP/IP and communicates those requests to the z/OS Connect server.

Getting started

Start the IBM Data Virtualization Manager studio and connect to the Data Virtualization Manager server that will host the Web Services to begin creating Web Services and operations, including:

Setting REST z/OS Connect Web Services preferences

Save your preferred settings to use when invoking and executing a Web Service request for z/OS Connect.

REST via z/OS Connect

To set **Web Services** preferences, complete the following:

1. From the **Window** menu, select **Preferences**.
 2. On the **Preferences** page, expand **Data Virtualization** and expand **Web Services**.
 3. Select **REST via z/OS Connect**.
 4. Complete the following fields:
 - **Service Provider URL(s)** – For each service provider, click **New** and enter the URL of the z/OS Connect server that you want to use to launch REST via z/OS Connect related actions and click **OK**. For example: `https://<host>:<port>/12346/dvs`. If multiple URLs are entered, the URL of the Data Virtualization Manager server that is currently connected is used.
 - **Max Records Parameter** – Enter the maximum number of records that you want returned. The default value is **20**.
 - **Prompt user before executing generated query** – Enable this option if you want to prompt users before executing a generated query using **REST via z/OS Connect**. By default, after generating a query the query is automatically executed and the generated REST URL is displayed in a web browser.
- Note:** Ensure that your z/OS Connect server is up and running. For more information on how to configure z/OS Connect for Data Virtualization Manager, refer [solution](#).
5. Click **Apply**.
 6. Click **OK**.

Connecting to z/OS Connect

Run the **z/OS Connect Configuration Wizard** to create the XML fragments used to connect the Data Virtualization Manager server to your z/OS Connect environment.

Before you begin

If you are using RACF, in order for the z/OS Connect server and the Data Virtualization Manager server to connect and function properly, CBIND resource classes are required. For more information, see [“RACF CBIND resource classes”](#) on page 123.

About this task

The **z/OS Connect Configuration Wizard** is used to provide the z/OS Connect systems programmer with the Data Virtualization information to include in the `z/OS Connect server.xml` file. If you do not have all of the information that the wizard prompts you to enter, you can choose to exclude that information from being generated. Completing this wizard is optional.

Procedure

1. To start the **z/OS Connect Configuration Wizard**, in the **Studio Navigator** click **z/OS Connect Configuration**.

2. The **z/OS Connect Configuration** page displays the default IBM Data Virtualization Manager for z/OS service provider settings. You can choose to complete the following Data Virtualization configuration fields (all blank fields are optional):

Section	Fields
HTTP Endpoint	<p>Enter the following Data Virtualization Manager server HTTP endpoint port numbers to use when connecting to a z/OS Connect server:</p> <ul style="list-style-type: none"> • HTTP Port • HTTPS Port <p>Exclude this fragment: If you already have the HTTP endpoint settings configured in a server.xml file or if you do not know the HTTP port numbers, you can choose to exclude this information from being generated.</p>
WOLA	<p>Enter the names to use to identify a specific z/OS Connect instance or a region in which to connect. To identify a Data Virtualization Manager server on a z/OS Connect instance, enter the following information:</p> <ul style="list-style-type: none"> • Register Name: Identifies a specific path between the Data Virtualization Manager server and a z/OS Connect instance. The name must be unique and it must match the RNAME parameter name that is defined in the Data Virtualization Manager server IN00 file. • WOLA Group Name: The name of the WOLA group to use. The name must be unique and it must match the WNAME parameter name that is defined in the Data Virtualization Manager server IN00 file.
Security	<p>SAF Group: Enter the z/OS Connect SAF group name.</p> <p>Exclude this fragment: If you already have the SAF group name configured in the z/OS Connect server.xml file or if you do not know the name, you can choose to exclude this information from being generated.</p>

3. Click **Finish**.

Review the results that display in the **XML Editor** and make any necessary modifications. You can copy XML fragments from the generated server.xml file into the z/OS Connect server.xml file. You can also choose to export and send the generated server.xml file to the appropriate z/OS Connect systems programmer. The following shows the results of a sample server.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Sample z/OS Connect Server Configuration for the DVS Service Provider.
This is not a complete server.xml file. It contains several XML sections
that you can copy and paste into your actual server.xml file on the host.
-->
<server>
<!--
HTTP Endpoint Details
-->
<httpEndpoint httpsPort="12346" httpPort="12345"
  host="*" id="defaultHttpEndpoint"/>
<!--
Enable SAF Security with Group Name (ZCONUSER)
-->
<webAppSecurity allowFailOverToBasicAuth="true"/>
<safRegistry id="saf"/>
<safAuthorization id="saf2"></safAuthorization>
<zosconnect_zosConnectManager globalInvokeGroup="ZCONUSER"
  globalOperationsGroup="<!--
Adapter Details with WOLA Group Name (RDVWOLA)
-->
<zosLocalAdapters wolaName3="NAME3" wolaName2="NAME2"
  wolaGroup="RDVWOLA"/><!--
  DVS Service Details with Register Name (DVSR1)
-->
<zosconnect_zosConnectService invokeURI="/dvs"
  serviceDescription="" serviceRef="dvsService" serviceName="dvsService"
  id="zosConnectDvsService"/>
```



```

<usr_dvsService invokeURI="/dvs" serviceName="DVSS1"
  registerName="DVSR1" connectionFactoryRef="wolaCF" id="dvsService"/>
<connectionFactory jndiName="eis/ola" id="wolaCF">
  <properties.ola/>
</connectionFactory>
<zosconnect_zosConnectService serviceRef="svc1"
  serviceAsyncRequestTimeout="600s" serviceName="dvs1" id="sdef1"/>
<zosconnect_localAdaptersConnectService connectionWaitTimeout="7200"
  connectionFactoryRef="wolaCF" serviceName="DVSS1"
  registerName="DVSR1" id="svc1"/>
<zosconnect_zosConnectManager id="dvsy1" requireSecure="false" requireAuth="true"
  preserveJsonObjectPayloadOrder="true" asyncRequestTimeout="600s"/>
</server>

```

Note:

Do not use /dvs for a base path name while creating APIs in the IBM Data Studio. This causes a conflict as /dvs is used as the invokeURI attribute for the service provider on zcEE.

What to do next

You can now begin creating the Web Services components using the studio wizards.

RACF CBIND resource classes

If you are using RACF, in order for the z/OS Connect server and the Data Virtualization Manager server to connect and function properly, CBIND resource classes are required.

Either a generic or discrete CBIND resource definition is required, as follows:

- For a generic definition, use a CBIND class of `BBG.WOLA.wolaGroup.**` with UACC(READ).
- For a discrete definition, use a CBIND class of `BBG.WOLA.wolaGroup.wolaName2.wolaName3` with UACC(READ).

The default values for `wolaName2` and `wolaName3` are `NAME2` and `NAME3`, respectively, and are defined in the `ZCONNECTPWNAMEX` parameter. If an existing zcEE server already exists with specified values for `wolaName2` and `wolaName3`, you must change the values in `ZCONNECTPWNAMEX` to those values.

Creating target systems

Create a target system on the Data Virtualization Manager server that the Web Services operations will use to map to mainframe resources.

Before you begin

Before creating the target system, you must know the name of the code page to use and the type of target system to create.

Procedure

1. On the **Server** tab, expand **Services > Target Systems** and click **Create Target System**.
2. On the **Target Systems** page, identify the target system to use by completing the following:
 - **Name** – Enter a unique name for the new target system.
 - **Connection**– Select the SQL92 connection type.
 - **CCSID** – Accept the default mainframe **1047** CCSID or use the drop-down list to select a different mainframe value.
3. Click **Finish**.
On the **Server** tab, the new target system displays under the **Target Systems** folder.

What to do next

After creating a target system, you can create Web Services directories that are used to identify the PDS on the mainframe where the metadata libraries exist.

Creating Web Services directories

You create Web Services directories to identify the Partitioned Data Set (PDS) on the mainframe where the metadata libraries exist.

Before you begin

To create a Web Service directory, you need the following:

- The URL path name that is used to access Web Services.
- The mainframe high-level qualifier to use.
- The name of the microflow library to use. This microflow library is being created for future use.
- All the necessary security authorization to perform updates to the metadata libraries.

Procedure

1. On the **Server** tab, expand **Services > Web Services** and select **Create Directory**.
2. On the **Web Services Directories** page, complete the following:
 - **Name** – Enter a unique name to identify the Web Services directory on your local machine. You cannot use this name when creating new Web Services within the directory. For example, if you name your directory "MyWebServices" you cannot name a Web Service within that directory "MyWebServices." This directory is used when generating files that reside on your local machine. No files or file structures will be created on the mainframe.
 - **High Level Qualifier**– Enter a high level qualifier to use as a data set prefix when creating, verifying, or editing the metadata library on the mainframe that is associated with this Web Services directory. By default, the data set name containing the Web Service and operations is `hlq.MMAP`. If this library does not already exist, it will be created.
 - **Supported Protocols**– Select the default **REST via z/OS Connect** or **SOAP** protocol which will be used by the Web Services in this directory.
 - **Advanced**
 - **URL Path** – To explicitly define the mainframe metadata library, enter the URL name in the text box. This is a data set where the metadata for all components contained within the library will be stored. For example, `/ZCEE/`.
 - **Metadata Library** – Accept the default metadata library that displays or enter the name of a specific metadata library. For example, `hlq.servername.MMAP`.
3. Click **Next**.
4. On the **Microflow Library** page, under the list of **Current Microflow Libraries** select a library to use or click **Create New Microflow Library** and complete the following fields:
 - **Library Name** – Enter a unique library name.
 - **Library Dataset** – Enter the data set name to use.
5. Click **Finish**.

The new Web Services directory displays on the **Server** tab under the **Web Services** folder.

What to do next

You can now begin creating Web Services and operations.

Creating Web Services and operations

You create Web Services and operations that are used to get the RESTful API requested mainframe data and to transform the data into objects.

Before you begin

Before creating a Web Service, you need to know the name of the virtual table or virtual view that will be used to generate the SQL statement. You must also identify or verify the Data Virtualization Manager server service provider URLs to use to launch REST via z/OS Connect related actions (for details, see [“Setting REST z/OS Connect Web Services preferences”](#) on page 121).

Procedure

1. On the **Server** tab, expand **Services > Web Services**.
2. Expand the **Web Services** directory where you want the Web Service to reside and double-click **Create Service**.
3. On the **Web Services** page, enter a unique name for the new Web Service and click **Next**.
4. To add a new operation to the Web Service, on the **Web Service Operation Type** page, select **Data Integration (REST via z/OS Connect or SOAP)** and click **Next**.
5. In the **Browse SQL Metadata** box, select the virtual table or virtual view to use when generating the SQL statement and click **Next**.
6. On the SQL page, accept the default **Name** and **Description** fields or enter a new name or description.
7. In the **SQL Statement** text box, accept or modify the SQL statement that displays and click **Next**.

Note: RESTful URIs use GET operation for SELECT statements and POST operation for INSERT, UPDATE, and DELETE statements.

8. In the **SQL Inputs** page, you can define the inputs to the SQL statements.
9. On the **SQL Results Set** page, verify that the results are correct and make any necessary modifications.
10. Click **Finish**.
The Web Service operation details display in the **Web Service Operation Editor**. If necessary, modify the details in the editor.
11. To validate your Web Service complete the following:
 - Right-click the operation in your new Web Service and select **z/OS Connect REST Interface > Refresh** to synchronize your new Web Service with the service provider.
 - Right-click the operation in your new Web Service and select **z/OS Connect REST Interface > Execute Query** to test the new Web Service in a web browser.
12. Optional: To test the Web Service, right-click the new operation or the new Web Service and click **Open in Tester**. Choose one of the following test options:
 - **Default:** use the default test settings.
 - **Create New:** create a new test group that you can name and configure.
 - Select an existing test group.
13. Optional: To create a .sar file, on the **Server** tab, right-click on the Web Service operation and click **Create z/OS Connect SAR File(s)**. The file is saved to your workspace and is available from the **zOSConnect** folder on the **Client** tab.

You can use this SAR file to create a service and deploy it using integration platforms such as IBM Explorer.

What to do next

To continue adding more operations under this Web Service group, click **Create New Operation** and repeat the steps starting at step 5.

Web services migration

Use the AVZSCMG1 member that is located in your *hlq.SAVZCNTL* data set as a sample JCL for migrating web services.

For information about the available parameters in the AVZSCMG1 member, see [“Sample JCL- Web Services Migration”](#) on page 17.

Server Trace

Use the **Server Trace** view to record and view Data Virtualization Manager server messages.

In the **Server Trace** view, you can perform the following tasks:

To collect and view diagnostics for the client, run the **Gather Diagnostics** wizard, which saves the information to a .zip folder.

Enabling studio calls in the Server Trace results

To include studio trace calls in your Server Trace results, enable the DV Data **Enable Server Tracing of Studio Calls** preference.

Before you begin

You must be able to connect to the Data Virtualization Manager server from which you want to collect trace information.

Procedure

1. From the **Window** menu, select **Open Preferences > DV Data**.
2. To enable tracing, select the **Enable Server Tracing of Studio Calls** check box. **Enable Server Tracing of Studio Calls** is enabled by default.
3. In the studio **HTTP Debug Option** drop-down list, select one of the following HTTP debug options:

Field	Action
Off	Do not collect HTTP messages. All trace activities are deactivated, including interactive tracing.
Normal	Commands that complete with a failure status are traced after execution, including the return codes.
All	All instructions are traced before execution.
Commands	All commands are traced before execution. Return codes are also traced for commands that complete with an error or failure status.
Error	Commands that complete with error status are traced after execution, including the return codes.
Failure	Commands that complete with a failure status are traced after execution, including the return codes.
Intermediates	All instructions are traced before execution. All terms, intermediate results, and substituted variable names are traced during expression evaluation. The final results of any expression that is evaluated also displays. Values assigned by arg , parse , or pull instructions are also traced.
Labels	Shows all labels when executed.
Results	All instructions are traced before execution. The final result of any expression that is evaluated also displays. Values assigned by arg , parse , or pull instructions are also traced.

Starting Server Trace

Start tracing Data Virtualization Manager server records in the **Server Trace** view.

Before you begin

Before running **Server Trace**, you must be able to connect to the Data Virtualization Manager server from which you want to collect the trace information.

Procedure

1. From the **Studio Navigator** view, on the **Common Tools** tab, click **Server Trace**.
2. To start tracing, click **Play** (the blue arrow).
The **Server Trace** table displays trace records.
3. Optional: To view message details, double-click the message and the details are displayed on the **Server Trace Zoom** page.
You can also choose to search for specific details within the message.

Filtering Server Trace results

Use the **Profile** option to filter the records that display in the **Server Trace** view.

Before you begin

You must be able to connect to the Data Virtualization Manager server from which you want to filter trace information. You can set filtering criteria before or after you run a Server Trace. Your most current filtering selections are automatically saved as your default filtering profile.

Procedure

1. On the **Server Trace** view, click **Profile**.
2. On the **Server Trace Profile** page, enable the fields that you want to include in the results.
3. For each enabled field, click **Add** to further filter your results. You can either select from the values that are displayed or enter the value when prompted.
4. Click **OK** to save changes to your profile and to apply the profile to the results in the **Server Trace** table.

What to do next

Use the **Display** option to select and sort columns that display in the filtered table. You can also choose to export the trace results.

Using Server Trace Zoom

Use **Server Trace Zoom** view to view Server Trace message details.

Before you begin

Server Trace must be running before you can open the **Server Trace Zoom** view.

Procedure

1. In the **Server Trace** view, double-click the message for which you want to view details.
2. In the **Server Trace Zoom** view, view message details and choose from the following options:

Field	Action
Previous	Click Previous to search for the previous occurrence of the text string entered.

Field	Action
Next	Click Next to search for the next occurrence of the text string entered.
Search	Click Search and enter a search string. To search for the next occurrence of the text string, click Search again.
Close	Click Close to close the search dialog.

Searching Server Trace messages

You can search Server Trace message results for a particular text string or message ID.

Before you begin

You must start the Server Trace before you can begin searching within the resulting Server Trace messages.

Procedure

1. On the **Server Trace** view, click the drop-down menu, and select **Search**.
2. On the **Search** page that is displayed, in the **From** section, select one of the following options to specify how to search within the results:

Field	Action
First	Search for the first occurrence of the text string.
Last	Search for the last occurrence of the text string.
ID	Search starting from the message ID you enter.

3. In the **For** field, enter the text string to use for searching within the message control blocks. Text strings cannot include spaces or special characters, and wild card searches are not supported.
4. Select **Previous** to find previous occurrences of the text string, or select **Next** to find the next occurrence of the text string.
5. Click **Search** to begin the search.

What to do next

View messages that meet the search criteria in the **Server Trace** view.

Labeling Server Trace messages

Create labels to bookmark server trace messages that you frequently access.

Before you begin

You must start the Server Trace before you can begin labeling messages.

Procedure

1. In the **Server Trace** view, right-click the message that you want to label and select **Add Label**.
2. On the **Message Label** dialog, enter text for the **Label** and click **OK**.
3. Optional: In the **Labels** view, double-click the label to locate the message in the **Server Trace** view.

Exporting Server Trace messages

Use the **Server Trace** view to export server trace messages as either ISX or CVS files.

About this task

You can limit the number of messages that you can export into a file by setting the **Server Trace export size limit** on the **Admin** preferences page.

Procedure

1. In the **Server Trace** view, from the drop-down menu, select **Export**.
2. Under **Export Type**, select one of the following message export options:

Field	Action
Summary	Exports the following minimum message information: <ul style="list-style-type: none">• Message ID• Date• Time• User ID• Message text
Full	Exports all available message information and all data about that message including: <ul style="list-style-type: none">• Message ID• Date• Time• User ID• Message text• Zoom
Comma Separated Format	Exports all table information to a CVS file. This file type cannot be imported for viewing in the Server Trace view.

3. Under **Export Content**, select one of the following message content options:

Field	Action
Message ID Range	Select a range of messages to export by entering the first message ID in From , and the last message ID to include in To .
Transaction ID	Exports only those messages with the RRS transaction ID value that you specify.
Global Transaction ID	Exports only those messages with the RRS global transaction ID that you specify.
Connection ID	Exports only those messages that are associated with a specific client that is currently connected to the server.
Message ID List	Lists message IDs. This option is only available if the Full export type option is selected.

4. Click **Next**.
5. On the **Export File** page, click **Browse** to specify a file name and export location.
6. Click **Finish**.

Importing Server Trace messages

To import and view Server Trace messages, use the **Import File Viewer** tab.

Before you begin

Server Trace must be running before you can import a file.

Procedure

1. In the **Server Trace** view, click the **Import File Viewer** tab and click **Import**.
2. Navigate to the ISX file that you want to import.
3. Double-click the ISX file. Messages and message details display on the **Import File Viewer** tab.
4. Optional: To view more details about a message, right-click on the message and select **Zoom**.
5. Optional: To change how the messages display, click **Display**.

DV Data preferences

Preferences allow you to customize several IBM Data Virtualization Manager for z/OS settings.

To view preferences, from the **Window** menu, select **Open Preferences > DV Data**.

Data Virtualization Manager preferences

Use **Data Virtualization Manager** preferences to set preferences such as general session and SQL results settings.

General **Data Virtualization Manager** preferences are identified and described in the table that follows.

Field	Description
Enable Server Tracing of Studio Calls	Includes the studio trace calls in your server trace results. This setting is enabled by default.
Studio HTTP Debug Option	The studio type of debug option to be used. The default setting is Normal .
Studio Fixed Width Font	Determines the font, font style, and font size that displays in studio. The default setting is Courier New-regular-9 .
Hex Encoding	Sets the Hex encoding to use. The default setting is UTF-8 .
File Encoding	Determines the file encoding setting to use. The default setting is windows-1252 .
CSV File Delimiter	Determines the type of file delimiter to use for CSV files. The default setting is a comma (,).
New Connection (DSN) Naming Pattern	Determines the naming pattern to use when new connections are made. The default setting is {SubSystem} .
Studio Connection Timeout (secs)	The number of seconds to wait before a server connection is determined to be unsuccessful. The default setting is 10 .
Studio Operation Timeout (secs)	The number of seconds to wait before determining that the studio operation is unsuccessful. The default setting is 30 .
Studio Remote Control Port	The port number that the studio uses for remote connections. The default setting is 31416 .

Field	Description
Use UPPER case logon credentials for both JDBC and HTTP connections	<p>Select this check box to require that logon credentials use uppercase characters for JDBC driver and HTTP connections. This setting is enabled by default.</p> <p>For systems that have mixed-case password support, you must clear this check box and add the following statement to your <i>hlq</i>.SAVZEXEC(AVZSIN00) file:</p> <pre>"MODIFY PARM NAME(PASSWORDCASE) VALUE(ASIS)"</pre>

Admin preferences

Use **Admin** preferences to set the maximum number of Server Trace messages that you want to export and to enable the tracing of Data Virtualization Manager studio calls in the **Server Trace** view.

Admin preferences are identified and described in the table that follows.

Field	Description
Server Trace export size limit	Sets the maximum number of messages to export. The default value is 5000. Specifying a value greater than 5000 can cause a MAX CPU TIME EXCEEDED error to occur.
Enable tracing of Studio Calls in Server Trace View	Includes studio trace calls in your Server Trace results. This setting is disabled by default.

Code generation preferences

Use **Code Generation** preferences to customize how your code is generated.

Code Generation preferences are identified and described in the table that follows.

Field	Description
Jupyter Kernel Name	The name of the kernel to use as defined in your Jupyter configuration.
Jupyter Kernel Name	<p>Identifies any additional JAR files that are not yet defined in the Spark configuration and that are required at runtime. The URL, <code>file://</code>, or <code>http://</code> naming format must be used. For example, you can add the following required JDBC driver JAR files:</p> <ul style="list-style-type: none"> <code>file:///opt/dv-jdbc/dv-jdbc-3.1.201608041929.jar</code> <code>file:///opt/dv-jdbc/log4j-api-2.6.2.jar</code> <code>file:///opt/dv-jdbc/log4j-core-2.6.2.jar</code>
INI Filename (as credentials store)	The INI file name that serves as your credentials store. The default setting is blank (<code>.</code>).
INI Data Set Name (DSN)	The INI file name to use for DSN. The INI file name that you choose to enter can contain multiple credentials. The DSN name is used to identify each set of credentials. For example, you could use the mainframe userid in the DSN name. The default setting is blank (<code>.</code>).

Console preferences

Use **Console** preferences to view or modify console display settings.

Console preferences are identified and described in the table that follows.

Field	Description
Fixed width console	Enable to specify a maximum number of characters to display in the console. This setting is disabled by default. Maximum character width: If Fixed width console is enabled, enter the maximum number of characters to display in the console. The default is 80 characters.
Limit console output	Enable to limit the console buffer and entry sizes by setting the maximum number of characters permitted: <ul style="list-style-type: none">• Console buffer size (characters). The default setting is 80000.• Console entry size limit (characters). The default setting is 500.

Dictionary preferences

Use **Dictionary** preferences to add or delete reserved words in dictionaries, and add or delete dictionaries based on the languages being used.

Dictionary preferences are identified and described in the table that follows.

Field	Description
Dictionary	Lists the default dictionaries. You can add new dictionaries to the list or delete existing dictionaries from the list.
Reserved word	Lists reserved words for each dictionary. You can add new words to the list or delete existing words from the list.

Driver preferences

Use **Driver** preferences to enable JDBC driver tracing and to specify the default location of the driver configuration files.

Driver preferences are identified and described in the table that follows.

Field	Description
Enable Tracing	Enables tracing for the JDBC driver. If you change this option, you must restart the IBM Data Virtualization Manager studio to complete the change. This setting is disabled by default. Note: You can also access data sources that are stored in other configuration files, by adding those configuration files from the Client view.
Default DSN Config File	Specifies the default location of the DSN file. This file is used to store the JDBC connection definitions that are generated for use in the Active Connections view.

Field	Description
Connection Overrides	To override the connection settings that the IBM Data Virtualization Manager studio uses when it creates JDBC connection definitions, specify a single name-value pair or a semicolon-delimited list to be used. The default setting is a blank field ().

SQL preferences

Use **SQL** preferences to specify settings related to SQL query generation, the SQL Results view, and SQL metadata retrieval.

SQL preferences are identified and described in the table that follows.

Field	Description
SQL Generate Query Behavior	Determines whether you are prompted to execute SQL or if SQL executes automatically. Options include: <ul style="list-style-type: none"> • Generate query and issue user prompt. This is the default setting. • Generate and execute query (no prompt) • Generate query but do not execute query (no prompt)
SQL Results Max Rows	Maximum number of rows to return in the SQL Results view. The default value is 1000.
SQL Results Max Bytes	Maximum number of data bytes to return in the SQL Results view. The default value is 1000000.
SQL Results values accessed as	Specifies how data values are returned. Options include String or Object. The default setting is String.
DB2 CARDINALITY in generated UDTF query	Specifies the cardinality value for the CARDINALITY clause that is included in generated UDTF queries. Using the CARDINALITY clause can improve the performance of queries with UDTF references. The default value is 10000. Specifying 0 omits the CARDINALITY clause from the generated query.
Use prepared statement to retrieve SQL column info for DB2 or DRDA tables	The IBM Data Virtualization Manager studio obtains column metadata information from the server for Db2 and DRDA tables and views when you expand a table or view node under the Other Subsystems tree in the Server view, or in other situations where column information needs to be retrieved. The IBM Data Virtualization Manager studio supports two different ways of retrieving this column metadata information: <ul style="list-style-type: none"> • Using a prepared statement. Typically, this server call will be faster; however, this option requires that the user have SELECT privileges to the table in the remote database. This method is the default and will be used when this preference is selected. • Using the JDBC getColumnns() API. This method is the more conventional approach; however, in some cases (for example, Oracle), the remote DRDA subsystem may take a long time to process the metadata query. This method will be used when this preference is cleared.

Field	Description
Fetch primary key and index information for virtual tables	If this preference is selected, then when you expand a virtual table or view in the Server view, any primary key or indexed column nodes will be identified. This identification process requires the IBM Data Virtualization Manager studio to make additional metadata calls to the server. To disable these calls and the associated identifications, you can clear this preference and thus speed up the time taken to populate the column nodes. This preference is selected by default.
Fetch primary key and index information for DB2 or DRDA tables	If this preference is selected, then when you expand a table or view node under the Other Subsystems tree in the Server view, any primary key or indexed column nodes will be identified. This identification process requires the IBM Data Virtualization Manager studio to make additional metadata calls to the server (and subsequently to the remote database). In some cases, these additional calls may be rather expensive (for example, Oracle). To disable these calls and the associated identifications, you can clear this preference to speed up the time taken to populate the column nodes. This preference is cleared by default.

Metadata Discovery preferences

Use **Metadata Discovery** preferences to define settings for the WebSphere Application Server that hosts IBM Rational Asset Analyzer (RAA).

When using RAA to access VSAM or sequential data sets for COBOL applications, complete COBOL layout information that is required to map data is not available in the Db2 database. The mapping wizard uses a RESTful HTTP query to collect record layouts when data is mapped. While this query can be done directly to the Data Virtualization Manager server for data in PDS files, the preferred method to collect COBOL information is to retrieve record layouts directly from the WebSphere Application Server that hosts RAA.

Metadata Discovery preferences are identified and described in the table that follows.

Field	Description
RAA REST Root URL	Location of the RAA WebSphere Application Server. For example: <code>https://<host>:<port></code>
Alternate User ID	User ID for the RAA WebSphere Application Server. You can leave this field blank if the credentials are the same as those used to connect to the current Data Virtualization Manager server (using Set Server).
Alternate Password	Password for the RAA WebSphere Application Server user ID. Specify a value in this field only if a user ID has been specified in the Alternate User ID field.

SSL preferences

Use **SSL** preferences to secure JDBC and HTTP network communications between the IBM Data Virtualization Manager studio and the Data Virtualization Manager server.

SSL preferences are identified and described in the table that follows.

Field	Description
Use SSL for Studio-Server communications (JDBC and HTTP)	<p>Enables secure JDBC and HTTP network communications between the IBM Data Virtualization Manager studio and the Data Virtualization Manager server.</p> <p>If enabled, select the Protocol version to use for communications between the IBM Data Virtualization Manager studio and the Data Virtualization Manager server.</p> <p>The default setting is TLS 1.2.</p>
Server Authentication	<p>Select the authentication strategy to use:</p> <ul style="list-style-type: none"> • Require Server Validation: Enable to require that all Data Virtualization Manager server certificates be authenticated and complete the following fields: <ul style="list-style-type: none"> – Truststore: The path name of the file on the local machine. The file must contain the Data Virtualization Manager server certificate authority (CA). – Password: The password for the truststore file. – Type: The truststore file type. For example, JKS, PKCS12, BKS, UBER. • Allow Self-Signed Certificates: Enable to allow the Data Virtualization Manager server to use self-signed certificates and complete the following fields: <ul style="list-style-type: none"> – Truststore: The path name of the file on the local machine. The file must contain the self-signed server CA (certificate authority) certificate. – Password: The password for the truststore file. – Type: The truststore file type. For example: JKS, PKCS12, BKS, UBER. • Trust All: Enable to allow all Data Virtualization Manager server certificates. If enabled, the IBM Data Virtualization Manager studio does not validate the server certificate. <p>The default setting is Require Server Validation.</p>
Client Authentication	<p>To enable client authentication by the Data Virtualization Manager server, select Enable Client Authentication and complete the following fields:</p> <ul style="list-style-type: none"> • Keystore: The path name of the file on the local machine. The file must contain a client certificate which has been signed by the server CA. • Password: The password for the keystore. • Type: The keystore file type. For example: JKS, PKCS12, BKS, UBER. • Alias: To confirm that the password is valid and that the alias (label) appears, click Refresh. <p>This setting is disabled by default.</p>

Exporting a virtual table to Software management configuration provider

Data Virtualization Manager allows you to export a Virtual Table (VT) to any Software management configuration (SCM) provider. Data Virtualization Manager Studio includes Jgit library, which is an Eclipse Distribution License based Java library, to provide APIs for implementing Git version control system. JGit supports repository access routines, version control features, and network protocols.

Before you begin

Configure an SSH key for the remote Git repository using the passphrase same as the password that is used to connect to Data Virtualization Manager. The SSH private key must be available in the default ssh keys directory **user.home/.ssh/id_rsa**. Data Virtualization Manager uses the SSH private key as credentials along with the passphrase while executing any Git APIs that need the credentials to be passed.

Procedure

1. Expand the **SQL > Data > SSID** node, where SSID is the name of your server.
2. Expand **Virtual Tables**.
3. Right-click the virtual table that you want to export and click **Export TO SCM**.
4. Provide the local GIT repository path in **Local Repository Path** and click **Next**.
5. Confirm the changes in the preview window.
6. Click **Finish**. The created XML files is exported to the GIT repository.

Exporting a virtual view to a Software management configuration provider

With Data Virtualization Manager, you can export a Virtual View (VV) to any Software management configuration (SCM) provider. Data Virtualization Manager Studio includes JGit library, which is an Eclipse Distribution License based Java library, to provide APIs for implementing Git version control system. JGit supports repository access routines, version control features, and network protocols.

Before you begin

Configure an SSH key for the remote Git repository using the passphrase same as the password that is used to connect to Data Virtualization Manager. The SSH private key must be available in the default ssh keys directory **user.home/.ssh/id_rsa**. Data Virtualization Manager uses the SSH private key as credentials along with the passphrase while executing any Git APIs that need the credentials to be passed.

Procedure

1. Expand the **SQL > Data > SSID** node, where SSID is the name of your server.
2. Expand **Virtual Views**.
3. Right-click the virtual view that you want to export and click **Export to SCM**.
4. Provide the local Git repository path in the **Local Repository Path** and click **Next**.
5. Confirm the changes in the preview window.
6. Click **Finish**.

Results

The created XML files are exported to the Git repository.

Runstats function

Use the Runstats function to analyze or troubleshoot an SQL query or find out statistics about the data that is stored in the target database table. The SQL engine uses the statistics that Runstats gathers to determine the access path to the stored data. Runstats works on all the data sources that Data Virtualization Manager supports.

Overview of the Runstats function

The Runstats function updates statistics in the system repository about the characteristics of a table and statistical views. These characteristics include the number of records, number of pages, and average record length. The optimizer uses these statistics to determine access paths to the data.

Runstats collects and stores the statistics within the global variable `NODE : GLOBAL9 , SUBNODE; STAT.` The `RUNSTATS – DISPLAY` option lets you meaningfully retrieve the statistics.

Note: With ADABAS, the Runstats function has the following limitations:

- The ADABAS S1 call on an ADABAS virtual table that has MapReduce enabled returns an error code 7 when the time exceeds the maximum search time that is set by the `TLSCMD` parameter.
- The Runstats function with the display option (`RUNSTATS – DISPLAY`) might not work.

Statistics collected by the Runstats function

- Total number of values in a particular column.
- Total number of distinct values in a column.
- Percentage distribution of unique values.
- Metadata about the table and columns.

Prerequisites to use Runstats

You require `READ` or `SELECT` access on the data sources where you want to use the Runstats function.

Using the Runstats function

Run the following command to collect the data statistics using the Runstats function:

```
SELECT RUNSTATS ('<Virtual_Table_Name>', 'RUNSTATS-OPTIONS');
```

You can use the following Runstats options:

RUNSTATS – RESET

Resets the previous statistics and restarts the collection of the data.

RUNSTATS – CLEAR

Deletes the collected Runstats data statistics.

RUNSTATS – DISPLAY

Displays the collected Runstats data statistics.

RUNSTATS – DB2STAT

Collects only the DB2 data statistics.

RUNSTATS – CATALOG

Collects the data statistics from the catalog data.

RUNSTATS – NOVDEC

Enables the `NOVDEC` flag that trims all the trailing and leading spaces from the characters and zeroes from the numbers.

RUNSTATS – NOHASH

Enables the `NOHASH` flag.

Note: The options are case-sensitive.

Example of Runstats with the RESET option on a VSAM file system:

```
select RUNSTATS('ZIPCODES', 'RESET');
```

Output:

```
RUNSTATS processed for ZIPCODES
```

Example of Runstats with the DISPLAY option on a VSAM file system:

```
select RUNSTATS('ZIPCODES', 'DISPLAY');
```

Output:

```
Statistics for table ZIPCODES, number of rows = 29467, average row length = 0
Total Unique PCT Name
29467 29467 100.00 ZIP
29467 16698 56.66 CITY
29467 51 0.17 STATE
29467 19001 64.48 POPULATION
29467 29459 99.97 LOC_X
29467 29435 99.89 LOC_Y
```

Example of Runstats with the RESET option on IMS:

```
select RUNSTATS('IMS_FLOOR_DEP', 'RESET');
```

Output:

```
RUNSTATS processed for IMS_FLOOR_DEP
```

Example of Runstats with the DISPLAY option on IMS:

```
select RUNSTATS('IMS_FLOOR_DEP', 'DISPLAY');
```

Output:

```
Statistics for table IMS_FLOOR_DEP, number of rows = 100007, average row length = 0
Total Unique PCT Name
100007 100007 100.00 DEP_KEY
100007 100006 99.99 DEP_KEY_NUMBER
100007 3 0.00 DEP_KEY_SUFFIX
100007 100007 100.00 RECORD_ID
100007 100005 99.99 PARENT_ID
100007 100007 100.00 CHILD_ID
```

Data Virtualization Manager support for NaturalONE version 9

Data Virtualization Manager supports NaturalONE version 9 (v9) by giving an option to generate a local stored procedure file that you can use with a compiler on another system.

NaturalONE v9 does not contain compilers on the mainframe servers. When you use Data Virtualization Manager to create a stored procedure using NaturalONE v9, the files are created but the compilation fails on the mainframe server.

Data Virtualization Manager now lets you create a local natural stored procedure (.NSP) file when you create a new virtual stored procedure that you can use with a compiler on a different system.

In the Data Virtualization Manager studio, when you create a virtual stored ACI procedure using the **Create Virtual Stored Procedure** dialog box, in the **ACI** panel, ensure that you select **Save Stub as .NSP**. When you run this step, a **Stub Generation Results** panel is displayed that contains the information about the location of the .NSP file. After the process completes, the .NSP file opens in the Data Virtualization Manager Studio.

If you use any other version of NaturalONE, you can generate a local .NSP file and compile it using a different compiler.

About virtualizing large Db2 columns

Easily query large, single Db2 columns by using COBOL copybook mapping layouts to map the Db2 columns to virtual tables.

With Data Virtualization Manager, you can create virtual Data Virtualization Manager tables that map a large, single Db2 column using a COBOL copybook or a PLI include file. When you query the virtual table, the single Db2 column is read and split into separate columns based on the original data mapping layout that is described in the copybook.

Data Virtualization Manager optimizes queries by matching the generated column names for the copybook structure with the column names in the Db2 table. When the column names match, the WHERE clauses of the columns referenced in the copybook virtual tables are automatically included in the WHERE clauses that are passed to retrieve data from the Db2 table.

For more information, see [“Using COBOL copybook map layouts to virtualize large Db2 columns” on page 139.](#)

Considerations

- The Db2 fixed character columns are limited to 254 bytes therefore, COBOL or PLI layouts greater than 254 bytes must be stored in Db2 as a LONG VARCHAR column. In these cases, the COBOL or PLI layout to map the VARCHAR must include a 2 byte binary length field as the first data item in the structure. If the layout does not include this data item, then the layout needs to be modified to ensure the correct alignment of the data.
- SQL UPDATE is supported, but SQL INSERT and DELETE statements are not supported unless the COBOL or PLI layout uses subtables and the insert or delete request is for a subtable instance. Insert and delete requests requiring an insert or delete of a Db2 row must use the virtual table mapped to the DB2 table instead of the virtual table mapped from the COBOL copybook or PLI include file.
- Db2 virtual tables are limited to a single link to a COBOL or PLI metadata table. If a DB2 table requires multiple links to a COBOL or PLI layout, then create separate Db2 virtual tables with unique table names for each link.

Using COBOL copybook map layouts to virtualize large Db2 columns

Virtualize your large Db2 columns using COBOL copybook mapping layouts to create virtual Data Virtualization Manager tables.

About this task

Use the following steps in the Data Virtualization Manager studio to create a Data Virtualization Manager virtual table using a COBOL copybook or a PLI include file mapping layout and link the metadata map to the Db2 column:

Procedure

1. For the Db2 table that contains the large column, right-click and select **Create Virtual Table (s)**.
2. From the **Virtualization Facility > Cobol**, select **Create Metadata** and then select and download the appropriate COBOL library.
A metadata map for the COBOL copybook or PLI include file is created.
3. Right-click the Db2 column and select **Link Db2 Column to Metadata Map**. Select the appropriate values for **Db2 Column Name** and **Metadata Map**.
The COBOL or PLI metadata map is linked the Db2 column.

Results

The Db2 column is available as a virtual table. You can query the virtual table that reads the Db2 column and splits the Db2 column data in fields using the COBOL copybook mapping layout.

Related information

[“About virtualizing large Db2 columns” on page 139](#)

Easily query large, single Db2 columns by using COBOL copybook mapping layouts to map the Db2 columns to virtual tables.

Chapter 4. Using JDBC Gateway

The *JDBC Gateway* is a Data Virtualization Manager distributed application server that allows direct connectivity to JDBC 4.0 data sources. The use of another federation server is not required.

Data sources

The JDBC Gateway solution is designed to work with any JDBC 4.0 compliant database. The following combinations of JDBC databases and drivers have been tested and verified to be supported by the JDBC Gateway:

- Hadoop 2.9.2 with the Hive 2.0 standalone JDBC driver
- Oracle 12 using the Oracle Thin Driver, version 6
- PostgreSQL version 11.1 using the JDBC driver version 42.2.5
- Greenplum versions 5.x and 6.x are supported using the JDBC driver.

Note:

1. The degree of JDBC compliance can vary across different driver vendor implementations and versions. In some cases, there may be interoperability problems when trying to use a particular JDBC driver to access a particular DBMS.
2. In PostgreSQL, it is recommended not to use double quotes for the table names or the column names in the CREATE script. Using double quotes will make the identifier as case sensitive and result in a failed query if exact match is not given.
3. Greenplum database support has the following limitations:
 - If there is a TEXT column or a BYTEA column in the Greenplum database, then Data Virtualization Manager is unable to create a virtual table.
 - Insert and update functions on boolean are not supported.
 - The **character_data** and **yes_or_no** data types are not supported.

Getting started

Use the following procedure to access your first data source using the JDBC Gateway:

1. Install the JDBC Gateway. See [Installing JDBC Gateway](#).
2. Start the JDBC Gateway server. See [“Starting the JDBC Gateway server”](#) on page 142.
3. Launch the JDBC Gateway administrative console in a supported browser using the following URL:

```
http://host:port/gateway
```

See [“Launching the JDBC Gateway administrative console”](#) on page 143.

4. In the JDBC Gateway administrative console, perform the following steps:
 - a. Determine the port that the JDBC Gateway will use for listening for incoming DRDA requests. You can review or change the port using the **Server Status** area of the JDBC Gateway administrative console. See [“Using the JDBC Gateway administrative console”](#) on page 143.
 - b. Set up access to the data source by performing the following tasks:
 - i) Locate and add JDBC driver information for the data source. See [“Adding JDBC driver information for a data source”](#) on page 145.
 - ii) Create a data source definition entry, specifying the location name, driver, URL and user information. See [“Creating a data source definition entry”](#) on page 147.
5. In the Data Virtualization Manager server, set up access to the data source by performing the following tasks:

- a. Register the connection to the JDBC Gateway by entering the location, host and the port for the data source.
- b. Enable the SEF rules and set global variables for the data source.

For information about these tasks, see [“Configuring the Data Virtualization Manager server for JDBC Gateway sources”](#) on page 148.

6. Use the Data Virtualization Manager studio to create virtual tables and views from the JDBC data source, just as you do for other supported sources, such as VSAM or IMS.

Starting the JDBC Gateway server

Start the JDBC Gateway server so that you can connect directly to JDBC data sources.

Before you begin

The JDBC Gateway must be installed. See [Installing JDBC Gateway](#).

About this task

Use the following procedure to start the JDBC Gateway server.

Information about the startup and additional activity of the JDBC Gateway is available in the Java Console and in the following log file:

```
home_dir_for_user_profile\Application Data\IBM\JDBC Gateway\log\jetty.out
```

Procedure

1. At a command prompt in the JDBC Gateway installation directory, run one of the following commands:

- For Windows:

```
startServer.cmd
```

- For Linux or Unix:

```
chmod 775 startServer.sh
```

```
./startServer.sh
```

Information about the startup process is displayed using the following format:

```
Using settings file: home_dir_for_user_profile\Application Data\IBM\JDBC Gateway\Settings\jgate.properties
Server is starting. It will be available on: http://localhost:port
Server process ID: processID
See home_dir_for_user_profile\Application Data\IBM\JDBC Gateway\log\jetty.out for server status
information.
```

2. Wait for the JDBC Gateway server startup process to complete, which is indicated by the following message in the `jetty.out` log file:

```
date time : JGATE Server started and ready to accept connections on port port_number
```

3. Optional: To stop the JDBC Gateway server, run the following command in the JDBC Gateway installation directory:

- For Windows:

```
stopServer.cmd
```

- For Linux or Unix:

```
chmod 775 stopServer.sh
```

```
./stopServer.sh
```

Results

The JDBC Gateway server has been started and is ready for use. Information about the activity of the JDBC Gateway is available in the Java Console and in the log files.

What to do next

Start the JDBC Gateway administrative console. See [“Launching the JDBC Gateway administrative console”](#) on page 143.

Launching the JDBC Gateway administrative console

Launch the JDBC Gateway administrative console so that you can configure connections to JDBC data sources.

Before you begin

The JDBC Gateway server must be installed and active. See [Installing JDBC Gateway](#) and [“Starting the JDBC Gateway server”](#) on page 142.

About this task

Use the following procedure to start the JDBC Gateway administrative console.

Only a single user (web client) can access the JDBC Gateway administrative console at a time.

Note: The JDBC Gateway does not require an external web application server. It contains its own Jetty web application server.

Procedure

1. In a web browser, launch the JDBC Gateway administrative console using the following URL:

```
http://server:port/gateway
```

where:

- *server* is the machine name or address where the JDBC Gateway server is running
 - *port* is the port specified during the installation
 - *gateway* is the gateway specified during the installation
2. Enter the **Username** and **Password** specified during installation.

The JDBC Gateway administrative console launches.

Results

The JDBC Gateway administrative console is running and ready for use. Information about the activity of the JDBC Gateway is available in the Java Console and in the log files.

What to do next

Configure access to data sources in the JDBC Gateway and the Data Virtualization Manager server. See [“Configuring access to data sources using the JDBC Gateway”](#) on page 144.

Using the JDBC Gateway administrative console

Use the JDBC Gateway administrative console to create and manage your data source definitions.

Before you begin

The JDBC Gateway must be installed, the JDBC Gateway server must be active, and the JDBC Gateway administrative console must be launched. See [Installing JDBC Gateway](#).

Procedure

Use the JDBC Gateway administrative console to create and manage your data source definitions. The following table describes the areas of the default JDBC Gateway view:

Field/Element	Action
Add New Data Source	Click this button add a new data source. For details, see “Creating a data source definition entry” on page 147 .
Location JDBC URL	Displays a list of defined data sources. Select an entry to display properties and location information. <ul style="list-style-type: none">• Location: Location name of the data source.• Note: This value corresponds to the LOCATION parameter defined for the data source in the Data Virtualization Manager server.• JDBC URL: The URL that points to the data source.
Server Status	Displays and controls the JDBC Gateway server status and the DRDA listener port. <ul style="list-style-type: none">• Status: JDBC Gateway server status. Click Start or Stop to control the server.• Port: The port on which the JDBC Gateway is listening for incoming DRDA requests. Click Edit to change the port number. This setting also allows you to control whether the server is started automatically when the JDBC Gateway startServer script is run.• Note: This port value will be used when adding a JGATE database definition statement to the Data Virtualization Manager server configuration file (AVZSIN00).
Location Information	Displays the following details for selected data source entry: <ul style="list-style-type: none">• Domain: Domain name of the JDBC Gateway.• Location: Name of the target data source.• Port: Port on which the JDBC Gateway is listening for incoming DRDA requests. Note: These values will be used when adding a JGATE database definition statement to the Data Virtualization Manager server configuration file (AVZSIN00). Click Test Connection to test the connection to the data source. If you have specified any information incorrectly you will not be able to connect.

Configuring access to data sources using the JDBC Gateway

Configure access to JDBC data sources that will be accessed using the JDBC Gateway.

To configure access for a data source, you must complete the following steps:

1. Add the compliant JDBC driver for the data source to the JDBC Gateway. See [“Adding JDBC driver information for a data source” on page 145](#).
2. Create the data source definition entry in the JDBC Gateway, specifying the location name, driver, URL, and user information. See [“Creating a data source definition entry” on page 147](#).
3. Configure the Data Virtualization Manager server for the data source. See [“Configuring the Data Virtualization Manager server for JDBC Gateway sources” on page 148](#).

Adding JDBC driver information for a data source

Add JDBC driver information to the JDBC Gateway.

Before you begin

The JDBC Gateway must be installed, the JDBC Gateway server must be active, and the JDBC Gateway administrative console must be launched. See [Installing JDBC Gateway](#).

About this task

The JDBC Gateway requires a compliant JDBC driver for each data source to be accessed. You must locate and add JDBC driver information for each data source. The driver files must be accessible to the JDBC Gateway. The JDBC Gateway retains the defined JDBC driver information, and you would only repeat this specification process to add new drivers or make changes to the properties of an existing driver.

In preparation for this task, obtain the following driver information for the data source from the data source vendor or from the driver documentation:

- Driver class name. For example: `org.postgresql.Driver`
- Driver JAR files
- URL format. Each data source type has a unique URL format that is used to access the data and is specific by vendor. For example, for Postgres: `jdbc:postgresql://{host}:{port}/{database}`

To add JDBC driver information to the JDBC Gateway, using the JDBC Gateway administrative console, you will define the driver library for the data source, and then add the driver files to the library. Use the following procedure to add JDBC driver information for a data source.

Procedure

1. In the JDBC Gateway administrative console, select **Preferences > JDBC Libraries**.

The following table describes the areas of the page:

Area	Description
JDBC driver libraries	JDBC driver libraries that are already set up. Use the search bar to quickly locate information in the table.
Driver files	JAR files associated with selected driver library.
Details	Additional information about the selected driver library

2. Add a driver library by performing the following steps:

- a) Click the **Add Driver** button.
- b) In the **Add New Driver Library** window, provide the following information:

Field	Action
Enter new library name	Enter a name for the library. The JDBC driver information for each type of database is organized by libraries. It is recommended that the name that you specify describes the JDBC information that will be included in the library. For example, if you are adding JDBC driver information for accessing Postgres databases, you might call the library <code>Postgres</code> . However, this is a descriptive field and can include any text.

Field	Action
Driver class name	Specify the actual name of the driver class that will be used. This information can be found in your JDBC driver documentation. For example: <code>org.postgresql.Driver</code>
URL templates	Optional: Specify a generic example of a correctly formatted URL that could be used to connect to the database. For example, if you are adding JDBC driver information for accessing Postgres databases, you might specify the following JDBC URL template: <code>jdbc:postgresql://<i>{host}</i>:<i>{port}</i>/<i>{database}</i></code> . The generic information as specified in the template is presented when you are adding data sources, where you will replace the generic information with the specific database information.

Note: The **Validate** and **JDBC Driver Properties** options are not applicable until the driver files have been added.

- c) Click **OK**.
3. Add JDBC driver files to the library by performing the following steps:
 - a) Click the **Add Driver Files** button.
 - b) In the **Add Files** dialog, click **Add** and specify the path to the JDBC driver files to add.
 - c) Click **OK**.
4. Optional: Update JDBC driver information as follows:
 - To edit the JDBC driver library information, validate the drivers, or add connection keywords, select an existing JDBC driver library from the list and click **Edit Driver**. The **Edit Driver Library** window opens where you can make changes to the library name, class name, and URL templates. You can also use the **Validate** option to validate the driver files, and the **JDBC Driver Properties** option to enter driver-specific connection keywords.
 - To remove a JDBC driver library, select an existing JDBC driver library from the list and click **Remove Driver**. The library, including all the JAR files that it contains, is removed.
 - To remove a JAR file from a JDBC driver library, select an existing file from the list and click **Remove Driver File**. The file is removed.
5. Click **OK**.

Results

The JDBC driver information is saved.

Note: You must repeat this process for each JDBC driver that will be used to access a data source type.

What to do next

Create the data source definition entry, specifying the location name, driver, URL, and user information. [“Creating a data source definition entry” on page 147.](#)

Creating a data source definition entry

Configure the JDBC Gateway for access to data sources.

Before you begin

The JDBC Gateway must be installed, the JDBC Gateway server must be active, and the JDBC Gateway administrative console must be launched. See [Installing JDBC Gateway](#).

Also, the compliant JDBC driver should be added to the JDBC Gateway. See [“Adding JDBC driver information for a data source”](#) on page 145.

About this task

Use the following procedure to create a data source definition entry. This data source definition entry is made in the JDBC Gateway administrative console and is used for access to the data source by the JDBC Gateway.

Procedure

1. In the JDBC Gateway administrative console, click the **Add New Data Source** button.
2. In the **JDBC Gateway** dialog, complete the following fields.

Field	Action
Location	Enter the location name. A valid value is a string 1 - 16 characters. For example: ORCL. Note: This value must match the LOCATION value that will be specified for the corresponding data source definition in the Data Virtualization Manager server configuration file.
Connection Parameters	Enter the JDBC connection information, as follows: <ul style="list-style-type: none">• JDBC Driver: Specify the library for the JDBC driver that will be used to access the data source. Select a library from the drop-down list, or click the ellipsis (...) option to the right of the field to open the Select JDBC Driver dialog where you can create additional JDBC driver libraries. (For more information, see “Adding JDBC driver information for a data source” on page 145.)• JDBC URL: Specify the URL that points to the data source to which you want to connect. The format for the URL can be displayed in the drop-down list if a JDBC URL template was supplied when the driver was configured. Note: You can also use the Build URL by URL-Template dialog box to form the correct string. Click Build URL to open the Build URL by URL-Template dialog box. From the JDBC URL drop-down list, select the template. In the table, specify the server, port, and database information and click OK. The result URL string is added to the JDBC URL list. This feature is available if a JDBC URL template was provided when the driver was configured.• Advanced: Click Advanced to specify any driver-specific connection string keywords and their values that will be used for the data source. The list of available advanced properties will change depending on both the type of driver being used, and the version of the driver. For information on any keywords that are required by a selected database driver, see the documentation for the driver.
Set User Information	Click Set User Information to provide authorization information used when accessing the data source. Provide the following information on the User Information dialog:

Field	Action
	<ul style="list-style-type: none"> • User ID and password are required: Select this option to require the use of a user ID and password when accessing the data source. If the data source allows access without a user ID and password, selecting this option will override that allowance. • Allow users to save password: Select this option to allow users to save passwords. • Allow users to change password: Select this option to allow users to change passwords. (Note: This option is for Db2 only.) • User name and Password: Specify the user ID and password that will be used to access the data source. The user ID and password that you specify when connecting to the data source are used to authorize the user.
Test Connection	Click Test Connection to test the connection to the data source. If you have specified any information incorrectly, you will not be able to connect.

3. Click **Finish**.

Results

The connection to the data source is validated. If successful, the data source location is added to the list of available data sources.

What to do next

[Configure the Data Virtualization Manager server for the JDBC Gateway source.](#)

Configuring the Data Virtualization Manager server for JDBC Gateway sources

Configure the Data Virtualization Manager server for use with the JDBC Gateway.

Before you begin

Configure access to the data source using the JDBC Gateway. See [“Creating a data source definition entry” on page 147](#).

About this task

To use the JDBC Gateway to connect to your data source, the following changes must be made to the Data Virtualization Manager server:

- The DEFINE DATABASE TYPE value must be set, as follows:

```
"DEFINE DATABASE TYPE(JGATE Datasource)"
```

JGATE

DDF endpoint is the JDBC Gateway.

Datasource

Optional

You can select HIVE, MSSQL, ORACLE, POSTGRES, or TERADATA as data sources.

- Optionally, the following utility and SEF procedure can be configured in support of TYPE(JGATE):

AVZDRATH

A utility that sets encrypted passwords in GLOBALU variables. You can also use this utility to list existing credential information.

AVZEJGAG

An ATH rule that switches credentials when connecting to a JGATE data source using DRDA. This rule uses AES encrypted passwords stored as GLOBALU system variables.

Procedure

1. In the Data Virtualization Manager server configuration file (**xVZyIN00**), register the connection to the JDBC Gateway using a definition statement, such as the following example:

```
"DEFINE DATABASE TYPE(JGATE datasource)"
    "NAME(name)"
    "LOCATION(location)"
    "DDFSTATUS(ENABLE)"
    "DOMAIN(your.domain.name)"
    "PORT(port)"
    "SECMEC(EUSRIDPWD)"
    "AUTHTYPE(AES)"
    "IPADDR(1.1.1.1)"
    "CCSID(37)"
```

The following table lists the parameters:

Parameter	Description	Valid values
AUTHTYPE	<p>Authentication type. This can be either DES for Diffie Hellman Encryption Standard or AES for Advanced Encryption Standard.</p> <p>When AUTHTYPE is not supplied, the default is DES.</p> <p>To force AES, the option must be added to the DEFINE DATABASE statement. Each server can be different in what is supported as to AES/DES.</p> <p>Each server can be different in what is supported as to AES/DES.</p> <p>For this setting to have effect, you must specify a security mechanism (SECMEC) that requests encryption.</p>	<p>DES Diffie Hellman Encryption Standard (default value)</p> <p>AES Advanced Encryption Standard.</p>
CCSID	<p>Specify the EBCDIC single-byte application CCSID (Coded Character Set Identifier) configured for this RDBMS subsystem on the RDBMS installation panel DSNTIPF, option 7. (<i>Optional</i>)</p>	<p>Refer to the RDBMS vendor documentation for a list of valid CCSIDs.</p>
DDFSTATUS	<p>The DDF activation status can be altered online by using the ISPF 4-Db2 dialog panels. (<i>Required</i>)</p>	<p>ENABLE Make this DDF definition active.</p> <p>DISABLE DDF endpoint is not used.</p>

Parameter	Description	Valid values
DOMAIN	The domain name or hostname on which the JDBC Gateway server is running. Either DOMAIN or IPADDR is required, but not both.	No default value.
IPADDR	The dot-notation IPV4 address of the host on which the JDBC Gateway server is running. Either DOMAIN or IPADDR is required, but not both.	If this parameter is not specified, the value 127.0.0.1 (local host) is the default. For group director definitions, use the DVIPA IP address of the group director.
LOCATION	For JGATE: The location name specified in the JDBC Gateway data source definition entry. See “Creating a data source definition entry” on page 147. <i>(Required)</i>	A valid value is a string 1 - 16 characters.
NAME	The database name as known to the server. <i>(Required)</i>	A valid value consists of 1 - 4 characters. Clients use this ID when they request access to a specific downstream database server.
PORT	The TCP/IP port on which the JDBC Gateway server is listening. <i>(Required)</i>	A valid 1-5 numeric string. If this keyword is not entered, the default DRDA port number 443 is used.
SECMEC	The DRDA security mechanism in force.	EUSRIDPWD Encrypt the user ID and password. USRIDPWD User ID and password are sent as is. No encryption is used. USRIDONL User ID is sent as is. No encryption is used for the user ID only (client security). USRENCPWD Encrypt password only.

Parameter	Description	Valid values
TYPE	<p>Defines the DDF endpoint type.</p> <p>JGATE DDF endpoint is the JDBC Gateway.</p> <p>datasource This is optional and you can select a valid datasource as per your requirement.</p>	<p>When using the JDBC Gateway, JGATE is the valid value.</p> <p>Datasource: Valid values for the datasource are: HIVE, MSSQL, ORACLE, POSTGRES, and TERADATA.</p>

2. Optional: To define alternate authentication information, use the sample job AVZDRATH to add a global default user definition or authentication information for specific mainframe users as follows:
 - a) Locate the AVZDRATH member in the *hlq.SAVZCNTL* data set.
 - b) Modify the JCL according to the instructions provided in the AVZDRATH member.

When adding the SYSIN statements that define the alternate credentials for logging in to your JDBC Gateway source, as instructed in the JCL, make sure to specify the correct DBTYPE. For JDBC Gateway sources, specify DBTYPE=JGATE.
 - c) Submit the job.
 - d) Optional: To verify the information stored in the GLOBALU variables and list existing authentication, use the REPORT=SUMMARY statement in the AVZDRATH member and submit the job.
 - e) Optional: To enter the encrypted password, enter ENCPWD = *encrypted password* statement in the AVZDRATH member and submit the job.
3. Optional: If using alternate authentication information, auto-enable the SEF ATH rule SAVZXATH(AVZEJGAG) to provide the logon credentials to each JDBC Gateway data source instance. Global variables are used to define alternate authentication credential mapping for the SEF ATH rule.
 - a) On the Data Virtualization Manager server - Primary Option Menu, select option **E** for Rules Mgmt.
 - b) Select option **2** for SEF Rule Management.
 - c) Enter * to display all rules, or ATH to display only authentication rules.
 - d) Enable the rule by specifying E and pressing Enter.
 - e) Set the rule to Auto-Enable by specifying A and pressing Enter.

Setting the rule to Auto-enable activates the rule automatically when the server is restarted.
4. Restart the Data Virtualization Manager server.

Results

The connection between the JDBC Gateway and the Data Virtualization Manager server for the JDBC data source has been defined.

What to do next

Use the studio to create virtual tables and views from the JDBC data source.

Encrypting Passwords in Data Virtualization Manager server using JDBC Gateway

Encrypt password in the Data Virtualization Manager server.

Before you begin

To encrypt a password, ensure the following:

- The JDBC Gateway server must be installed and active. See [Installing JDBC Gateway](#)
- The user must have login credentials to Mainframe.

- Set ENCRPASS to ENABLE in the server to use the web interface.

About this task

Encryption of the password is a part of configuring the Data Virtualization Manager server. This section outlines the steps to encrypt your password.

Procedure

1. In a web browser, launch the Encrypt Password console using the following URL:

```
http://server:port/encryptpassword
```

where:

- *server* is the machine name or address where the Mainframe server is running.
- *port* is the server port.

Note: Always use SSL connection.

2. Enter the **Username**, **Password** and then click **Sign in**. Use your mainframe credentials. The Encrypt Password console launches.
3. Enter the **Password** and then re-enter the same password. Click **Generate Encrypted Password**. The **Encrypted Password** box displays the encrypted password.

Example: Configuring access to Oracle data

Configure the JDBC Gateway for access to Oracle data.

Before you begin

The JDBC Gateway must be installed, the JDBC Gateway server must be active, and the JDBC Gateway administrative console must be launched. See [Installing JDBC Gateway](#).

About this task

Use the following procedure to configure access to Oracle data.

Procedure

1. Download the Oracle Thin Driver from the Oracle website. For example, ojdbc8.jar.
2. In the JDBC Gateway administrative console, select **Preferences > JDBC Libraries**, and then complete the following steps:
 - a) Select the row for the **Driver Library Name Oracle Thin Driver** in the table, and click **Add Driver Files**.
 - b) Use the **Add Files** dialog to add the Oracle Thin Driver file.
 - c) Click **OK** to close the **JDBC Libraries** preference page.
3. Create a JDBC Gateway data source for Oracle as follows:
 - a) Select **File > New > Other**, and then in the **New** wizard dialog, select **Data Source** and click **Next**.
 - b) Complete the following fields:

Field	Action
Location	Enter the location name. For example, Oracle.

Field	Action
Connection Parameters	Enter the connection parameters: <ul style="list-style-type: none"> • JDBC Driver: From the drop-down list, select Oracle Thin Driver. • JDBC URL: Enter the JDBC URL as follows: <code>jdbc:oracle:thin:@//oracle-host:1521/ORCL</code>
Set User Information	Click Set User Information , and enter the credentials for accessing the Oracle database, as follows: <ul style="list-style-type: none"> • User name: <code>OracleUser</code> • Password: <code>OraclePwd</code>

- c) Click **Test Connection**.
- d) Click **Finish**.
4. In the Data Virtualization Manager server configuration file, register the connection to the JDBC Gateway data source using a definition statement, such as the following example:

```
"DEFINE DATABASE TYPE(JGATE) "
      "NAME(ORCL) "
      "LOCATION(Oracle) "
      "DDFSTATUS(ENABLE) "
      "SECMEC(USRIDPWD) "
      "PORT(1527) "
      "IPADDR(10.26.4.125) "
      "CCSID(37) "
      "IDLETIME(110) "
```

For details about this statement, see [“Configuring the Data Virtualization Manager server for JDBC Gateway sources”](#) on page 148.

5. In the Data Virtualization Manager server, enable rule AVZEJGAG. For more information, see [“Configuring the Data Virtualization Manager server for JDBC Gateway sources”](#) on page 148..

Results

The following connections have been established:

- The connection from the JDBC Gateway to the Oracle data source
- The connection between the JDBC Gateway and the Data Virtualization Manager server for the Oracle data source

What to do next

Use the studio to create virtual tables and views to access the Oracle data.

Setting preferences

The **Preferences** dialog is used to set user preferences and add necessary drivers.

The **Preferences** window consists of two panes. The left pane displays the list of preferences groups and the right pane displays the page for the selected group. The following groups of preferences are displayed in the **Preferences** window:

- [JDBC Libraries](#)
- [Log](#)
- [Output](#)

Setting JDBC driver preferences

Use the **JDBC Libraries** preferences to set up and manage JDBC driver information for your data sources.

About this task

You can use the **JDBC Libraries** preferences page to review, define or update JDBC driver information for each type of database (such as Db2®, Informix®, Oracle) that will be accessed.

Use the following procedure to access the **JDBC Libraries** preferences page. For details about adding new driver definitions, see [“Adding JDBC driver information for a data source” on page 145](#).

Procedure

1. To access the **JDBC Libraries** page, select **Preferences > JDBC Libraries**.
All of the JDBC driver libraries that you have already set up are listed in the **JDBC driver libraries** area. The JAR files associated with selected driver library are listed in the **Driver files** area. Additional information about the selected driver library is displayed on the **Details** panel.
2. For information about adding or editing driver definitions, see [“Adding JDBC driver information for a data source” on page 145](#).

Setting log preferences

Use the **Log** page of the **Preferences** window to activate a log file that will track JDBC Gateway processing information.

About this task

The log file information can be useful in debugging.

It is recommended to leave the log level at the default setting of `error`. Only increase the level at the direction of IBM Software Support.

Use the following procedure to specify the log file preferences.

Procedure

1. Click **Preferences > Log**.
2. Check **Enable log** to activate the log file for debugging purposes. If this check box is selected, the log file option fields are enabled.
3. Check one or more of the log file options to indicate what information should be gathered. It is recommended that all options remain checked. The available log file options are as follows:
 - Print stack trace for log exceptions
 - Print log class and method
 - Print log user token
4. Click **Edit Log Categories** to modify the category level.
The following levels are available: none, emergency, alert, critical, error, warning, notice, info, debug, all.
5. Click **Apply** to save your preferences choices.
6. Click **Restore Defaults** to restore the default preference values.
7. Click **OK** to close the **Preferences** window.

You can collect the generated log file and save it as a zip file to be able to send it for IBM Software Support.

To collect the generated log file:

8. Click **Help > Collect Support Data....** In the **Collect Support Data** window, you can choose the option **All dates** if you want all the files generated from the beginning or specify a date range.

9. Click **Save Report**. This will save the log file as a zip file on your local system.

Setting output preferences

You can use the **Output** page of the **Preferences** window to activate the **Output** view that tracks the information about errors and connections in the JDBC Gateway.

About this task

The information from the **Output** view can be useful for debugging. It can be delivered as a report in the **Output** view and automatically added to the log file.

Use the following procedure to specify the output file preferences:

Procedure

1. Click **Preferences > Output**.
2. On the **Output** page, you can specify the following options:
 - Show errors**
This option displays all error texts in the **Output** view.
 - Show connection status**
This option displays the statuses of connections to data sources in the **Output** view.
 - Automatically activate Output view**
When an error occurs or a message appears, this option automatically opens the **Output** view.
3. Click **Apply** to save your preferences choices.
4. Click **Restore Defaults** to restore the default preference values.
5. Click **OK** to close the **Preferences** window.

Troubleshooting

Collect troubleshooting data to provide to technical support.

About this task

Use the following procedure to collect troubleshooting data.

Procedure

1. Set the log level to debug. See [“Setting log preferences” on page 154](#).
2. Reproduce the issue.
3. Set the log level to the previous value.
4. Select **Help > Collect Support Data**.
5. Complete the fields and click **Save Report**.

Chapter 5. SQL DMF supported data types

This appendix contains the language-specific data definitions that are used by the Data Mapping Facility (DMF) and shows the equivalent SQL data types that are used by IBM Data Virtualization Manager for z/OS. It also shows the SQL data types that are supported by the different interfaces in IBM Data Virtualization Manager for z/OS.

Adabas

Although it is not a programming language, Adabas has a file definition, created by the Adabas database administrator, that is used to generate a map.

Data definition	SQL type	Host format
A - Alphanumeric	SQL_Char	Character
B - Binary	SQL_Binary	Binary
F - Fixed point	Length 2 SQL_Smallint	Smallint
	Length 4 SQL_Integer	Integer
	Length 8 SQL_BigInt	BigInt
G - Floating point	Length 4 SQL_Float	Float
	Length 8 SQL_Double	Float
P - Packed decimal	SQL_Decimal	Packed Decimal
	Length 4 SQL_Date	Date
	Length 7 SQL_Timestamp	Timestamp
U - Unpacked decimal	SQL_Char	Unpacked decimal
W – Wide Alphanumeric	Not supported	Not supported

COBOL

This topic lists the data definitions of COBOL that are used by the Data Mapping Facility (DMF) and shows the equivalent SQL data types that are used by IBM Data Virtualization Manager for z/OS.

Data definition	SQL data type	Host format
PIC X(30) PIC A(30)	SQL_Char	Character
PIC S9(3)V9(3) PIC S9(3)V9(3) USAGE DISPLAY	SQL_Char	Display Numeric
PIC G(30) USAGE DISPLAY-1	SQL_Graphic (SQL_Unicode)	Graphic (DBCS)

Table 28. COBOL data definitions used by DMF (continued)

Data definition	SQL data type	Host format
PIC S9() USAGE BINARY PIC S9() USAGE COMP PIC S9() USAGE COMP-4	Length 1 to 4 SQL_Smallint Length 5 to 9 SQL_Integer Length 10 to 18 SQL_Binary	Smallint Integer Binary Note: Fields with a length of 10 to 18 become SQL_BIGINT when support for BIGINT is added.
USAGE IS COMP-1	SQL_Float	Float
USAGE IS COMP-2	SQL_Double	Float
PIC S9(03)V9(3) USAGE COMP-3 PIC S9(03)V9(3) USAGE PACKED-DECIMAL	SQL_Decimal	Packed Decimal
PIC S9(7)V99 COMP	SQL_Decimal	Signed Binary Integer
PIC 9(7)V99 COMP	SQL_Decimal	Unsigned Binary Integer
PIC S9() USAGE COMP-5 PIC 9() USAGE COMP-5	Length 1 to 4 SQL_Smallint Length 5 to 9 SQL_Integer Length 10 to 18 SQL_Binary	Smallint Integer Binary Note: Fields with a length of 10 to 18 become SQL_BIGINT when support for BIGINT is added.

Table 29. PIC S9() USAGE COMP-5

Picture	Storage representation	Numeric values
S9(1) through S9(4)	Binary half-word (2 bytes)	-32768 to +32767
S9(5) through S9(9)	Binary full-word (4 bytes)	-2,147,483,648 to +2,147,483,647
S9(10) through S9(18)	Binary double-word (8 bytes)	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807

Table 30. PIC 9() USAGE COMP-5

Picture	Storage representation	Numeric values
9(1) through 9(4)	Binary half-word (2 bytes)	0 to 65535
9(5) through 9(9)	Binary full-word (4 bytes)	0 to 4,294,967,295
9(10) through 9(18)	Binary double-word (8 bytes)	0 to 18,446,744,073,709,551,615

IMS - DBD (database description)

A database description defines an IMS database. To increase the available data type, merge a COBOL map with the database description.

Table 31. Data definitions for IMS - DBD

Data definition	SQL type	Host format
TYPE=C - Alphanumeric	SQL_Char	Character
TYPE=X - Hexadecimal	SQL_Binary	Binary
TYPE=P - Packed Decimal	SQL_Decimal	Packed Decimal
TYPE=F - Binary Fullword Note: Only valid for MSDB databases.	SQL_Integer	Integer
TYPE=H - Binary Halfword Note: Only valid for MSDB databases.	SQL_Smallint	Smallint

Natural conversions

The table describes how Natural data types are converted to ODBC data types.

Natural	ODBC
A-Alphanumeric	SQL_CHAR
B-Binary	(If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
C-Attribute Control	N/A
D-Date	*SQL_DECIMAL
F-Floating Point	(If 4 bytes) SQL_FLOAT (If 8 bytes) SQL_DOUBLE
I-Integer	(If 1 byte) SQL_BINARY (If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
L-Logical	SQL_BINARY
N-Numeric	SQL_NUMERIC
P-Packed	SQL_DECIMAL
T-Time	*SQL_DECIMAL

Note: Although the IBM Data Virtualization Manager for z/OS Interface for ADABAS supports the conversion of ODBC date and time to the Natural date and time format, the IBM Data Virtualization Manager for z/OS Interface for Natural only allows the passing of the internal format for date and time (P6 and P12, respectively).

Natural DDM (data definition module)

A DDM is a file that is used to create a view of an ADABAS file. It is used to provide long column names and to limit the view to a subset of the fields that are defined in the Adabas file.

Table 32. Data definitions for Natural DDM

Data definition	SQL type	Host format
A – Alphanumeric	SQL_Char	Character
B - Binary	SQL_Binary	Binary
F - Fixed point	Length 2 SQL_Smallint	Smallint
	Length 4 SQL_Integer	Integer
G - Floating point	Length 4 SQL_Float	Float
	Length 8 SQL_Double	Float
P - Packed decimal	SQL_Decimal	Packed Decimal
N – Unpacked decimal	SQL_Char	Unpacked decimal
D - Date	SQL_Date	
T - Time	SQL_Time	Not supported
	SQL_Graphic	Graphic (DBCS)

SQL Type Support by the IBM Data Virtualization Manager for z/OS interface

SQL Type	VSAM/ CICS VSAM	Adabas	ACI	SQL/IMS	CICS SP	IMS SP
SQL_Char	X	X	X	X	X	X
SQL_Numeric	X	X	X	X	X	X
SQL_Decimal	X	X	X	X	X	X
SQL_Bigint						
SQL_Integer	X	X	X	X	X	X
SQL_Smallint	X	X	X	X	X	X
SQL_Float						
SQL_Real						
SQL_Double						
SQL_Date		X		X		
SQL_Time	X	X				
SQL_Binary	X	X	X	X	X	X
SQL_Graphic	X	X	X	X	X	X

Chapter 6. Supported SQL functions

This chapter describes the SQL functions supported in Data Virtualization Manager for z/OS.

The JDBC and ODBC drivers has aggregation capabilities and supports parallelism for the following functions.

- Count
- Min
- Max
- SUM
- AVG
- ORDER BY

Based on the MapReduce (MRC) value set in the driver connection properties, the driver queries the Data Virtualization Manager in threads, aggregate the results, and display them.

Examples are used to explain some of the supported SQL functions. Following tables are used in the examples to illustrate the SQL function outputs:

Table 33. VIRTUAL TABLE - EMPL_COMP

EMPL_ID	EMPL_NAME	COMPANY	LOCATION	UPDATED_TS
1	AKHIL	TIMBER INC.	NY	2019-11-26 05:39:59.454707
2	ELICA	STANLEY INC.	TX	
3	SMITH	CHROME INC.	WA	
1919574970	KANE			

Table 34. VIRTUAL TABLE - EMPLOYEE

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					

Table 35. Virtual Table - EMP

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
DEV	33	ADMIN	RETAIL	25000	2019-11-27
WAYNE	36	ADMIN	RETAIL	45000	2018-11-01

Table 36. Virtual Table - JOIN1

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
999	JOHN		SALES		16.99	
10	MIC	20	MGR	7	98357	0.00
20	WILLIAM	20	SALES	8	78171	612.45
30	MIC	38	MGR	5	77506	0.00
40	OBRIEN	38	SALES	6	78006	846.55

Table 37. Virtual Table - JOIN2

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
0	SANDERS	20	MGR	7	18357	
1	WILLIAM	20	SALES	8	18171	612.45

ABS

The ABS function returns the absolute value of a number.

➡ ABS (numeric-expression) ➡

The argument must be an expression that returns a value of any built-in numeric data type.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The result of the function has the same data type and length attribute as the argument.

Notes

Syntax alternatives:

ABS should be used for conformance to the SQL standard.

Example:

The following statement returns the absolute value of the values in the *SALARY* column from the table *EMPLOYEE*.

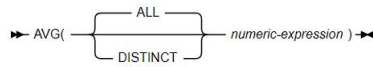
```
SELECT ABS(SALARY) FROM EMPLOYEE;
```

The above example returns the following:

```
50000
20000
5000
25000
75000
75000
```

AVG

The AVG function returns the average of a set of numbers.



The argument values can be of any built-in numeric data type, and their sum must be within the range of the data type of the result.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The data type of the result is determined as follows:

- DECFLOAT(34) if the argument is DECFLOAT(n).
- Large integer if the argument is small integer.
- Double precision floating-point if the argument is single precision floating-point.
- Otherwise, the result is the same as the data type of the argument.

If the data type of the argument values is decimal with precision p and scale s , the precision (P) and scale (S) of the result depend on p and the decimal precision option:

- If p is greater than 15 or the DEC31 option is in effect, P is 31 and S is $\max(0, 28 - p + s)$.
- Otherwise, P is 15 and S is $15 - p + s$.

The function is applied to the set of values derived from the argument values by the elimination of null values. If DISTINCT is specified, redundant duplicate values are also eliminated.

If the type of the result is integer, the fractional part of the average is lost.

Example:

The following example returns the average of values in the column SALARY in the table EMPLOYEE.

```
SELECT AVG(SALARY) FROM EMPLOYEE;
```

The above example returns 49000.00000.

BETWEEN

The BETWEEN predicate determines whether a given value lies between two other given values that are specified in ascending order.



Each of the predicate's two forms has an equivalent search condition, as shown in the following table:

Table 38. BETWEEN predicate and equivalent search conditions	
BETWEEN predicate	Equivalent search condition
$value1$ BETWEEN $value2$ AND $value3$	$value1 \geq value2$ AND $value1 \leq value3$ Note: Might not be equivalent if $value1$, $value2$, or $value3$ are columns or derived values based on columns that are not the same CCSID set because the clause is evaluated in Unicode.
$value1$ NOT BETWEEN $value2$ AND $value3$ or, equivalently: NOT($value1$ BETWEEN $value2$ AND $value3$)	$value1 < value2$ OR $value1 > value3$ Note: Might not be equivalent if $value1$, $value2$, or $value3$ are columns or derived values based on columns that are not the same CCSID set because the clause is evaluated in Unicode.

If the operands include a mixture of datetime values and valid string representations of datetime values, all values are converted to the data type of the datetime operand.

Example:

```
A BETWEEN B AND C
```

The following example returns the record where the age of an employee in the table `EMPLOYEE` is `>50`.

```
SELECT EMPL_NAME,AGE FROM EMPLOYEE
WHERE AGE >50
```

The above example returns the following.

<i>Table 39. Between</i>	
EMPL_NAME	AGE
BUMRAH	53
NEWTON	53

BIGINT

The `BIGINT` function returns a big integer representation of either a number or a character or graphic string representation of a number.

➡ `BIGINT(numeric-expression)` ➡

Numeric to Big Integer

➡ `BIGINT(string-expression)` ➡

numeric-expression

An expression that returns a value of any built-in numeric data type.

The result is the same number that would occur if the argument were assigned to a big integer column or a variable. If the whole part of the argument is not within the range of big integers, an error is returned. The fractional part of the argument is truncated.

String to Big Integer

string-expression

An expression that returns a value of a character or graphic string (except a `CLOB` and `DBCLOB`) with a length attribute that is not greater than 255 bytes. The string must contain a valid string representation of a number.

The result is the same number that would result from `CAST(string-expression AS BIGINT)`. Leading and trailing blanks are eliminated and the resulting string must conform to the rules for forming an integer constant. If the whole part of the argument is not within the range of big integers, an error is returned.

The result of the function is a big integer.

To increase the portability of applications, use the `CAST` specification.

Example 1:

The following example returns 7 from the table EMPLOYEE:

```
SELECT BIGINT(COUNT(*)) FROM EMPLOYEE;
```

CASE

The CASE statement selects an execution path based on the evaluation of one or more conditions. A CASE statement operates in the same way as a CASE expression.

Syntax



simple-when-clause:



searched-when-clause:



Description

CASE

Begins a case-expression.

simple-when-clause

Specifies the expression prior to the first WHEN keyword that is tested for equality with the value of each expression that follows the WHEN keyword, and the result to be executed when those expressions are equal. If the comparison is true, the THEN statement is executed. If the result is unknown or false, processing continues to the next expression or the ELSE statement.

The data type of the expression prior to the first WHEN keyword must be comparable to the data types of each expression that follows the WHEN keywords.

searched-when-clause

Specifies the search-condition that is applied to each row or group of table data presented for evaluation, and the result when that condition is true. search-condition cannot contain a fullselect. If the search condition is true, the THEN statement is executed. If the condition is unknown or false, processing continues to the next search condition or the ELSE statement.

SQL-procedure-statement

Specifies a statement that follows the THEN and ELSE keyword. The statement specifies the result of a searched-when-clause or a simple-when-clause that is true, or the result if no case is true.

search-condition

Specifies a condition that is true, false, or unknown about a row or group of table data.

ELSE SQL-procedure-statement

If none of the conditions specified in the simple-when-clause or searched-when-clause are true, the statements in the else-clause are executed.

If none of the conditions specified in the WHEN clause are true and an ELSE clause is not specified, an error is returned at run time, and the execution of the CASE statement is terminated.

END CASE

Ends a case-statement.

Note:

If none of the conditions specified in the WHEN clause are true and an ELSE clause is not specified, an error is returned at run time, and the execution of the CASE statement is terminated.

CASE statements that use a simple case statement WHEN clause can be nested up to three levels. CASE statements that use a searched statement WHEN clause have no limit to the number of nesting levels.

Examples:

The following example assigns the experience level of *1* to employees with age <40 and the experience level of *2* to employees with age >40 from the table `EMPLOYEE`.

```
SELECT EMPL_NAME,  
CASE  
WHEN AGE > 40 THEN 'LEVEL 2'  
WHEN AGE < 40 THEN 'LEVEL 1'  
END AS EXP_LEVEL,  
FROM EMPLOYEE;
```

The above example returns the following:

Table 40. Case	
EMPL_NAME	EXP_LEVEL
BUMRAH	LEVEL 2
CEASAR	LEVEL 2
ELICA	LEVEL 1
DEV	LEVEL 1
NEWTON	LEVEL 2
SAMUEL	LEVEL 2
ELICA	

CHAR

The CHAR function returns a fixed-length character string representation of the argument.

The syntax of the CHAR function depends on the data type of the input argument. The following types of input arguments are accepted.

The CHAR function returns a fixed-length character string representation of one of the following values:

- An integer number if the first argument is a SMALLINT, INTEGER, or BIGINT
- A decimal number if the first argument is a decimal number
- A double-precision floating-point number if the first argument is a DOUBLE or REAL
- A decimal floating-point number if the first argument is a DECFLOAT
- A character string value if the first argument is any type of character string
- A graphic string if the first argument is an EBCDIC or Unicode graphic string
- A datetime value if the first argument is a date, time, or timestamp
- A row ID value if the first argument is a row ID

The result of the function is a fixed-length character string (CHAR).

The result can be null; if the first argument is null, the result is the null value.

Integer to Character:

➔ CHAR(*integer-expression*) ➔

integer-expression

An expression that returns a value that is a built-in integer data type (SMALLINT, INTEGER, or BIGINT).

The result is the fixed-length character string representation of the argument in the form of an SQL integer constant. The result is the smallest number of characters that can be used to represent the value of the argument, padded with blanks. The result consists of n characters that are the significant digits that represent the value of the argument with a preceding minus sign if the argument is negative. A positive value starts with a digit and always includes at least one trailing blank. Leading zeroes are not included. The result is left justified:

- If the argument is a small integer, the length of the result is 6. If the number of characters in the result is less than 6, the result is padded on the right with blanks.
- If the argument is a large integer, the length of the result is 11; if the number of characters in the result is less than 11, the result is padded on the right with blanks.
- If the argument is a big integer, the length of the result is 20. If the number of characters in the result is less than 20, the result is padded on the right with blanks.

A positive value always includes one trailing blank.

The CCSID of the result is determined from the application encoding scheme.

Decimal to Character:

► CHAR(*decimal-expression* , *decimal-character*) ◄

decimal-expression

An expression that returns a value that is a built-in decimal data type. To specify a different precision and scale for the value of the expression, apply the DECIMAL function before applying the CHAR function.

decimal-character

Specifies the single-byte character constant that is used to delimit the decimal digits in the result character string. The character must not be a digit, a plus sign (+), a minus sign (-), or a blank. The default is the period (.) or comma (,).

The result is the fixed-length character string representation of the first argument. The result is the smallest number of characters that can be used to represent the value of the argument, except that trailing zeros are included.

The result includes a decimal character and up to p digits, where p is the precision of the *decimal-expression* with the preceding minus sign if the argument is negative. A positive value starts with a digit or the decimal character, and always includes at least one trailing blank. Leading zeros are not returned. If the scale of *decimal-expression* is zero, the decimal character is not returned. If the number of bytes in the result is less than the defined length of the result, the result is padded on the right with blanks.

The length of the result is 2 + p, where p is the precision of the *decimal-expression*.

The CCSID of the result is determined from the application encoding scheme.

Floating-Point to Character:

► CHAR(*floating-point-expression*) ◄

floating-point-expression

An expression that returns a value that is a built-in floating-point data type (DOUBLE or REAL).

The result is the fixed-length character string representation of the argument in the form of an SQL floating-point constant. If the argument is negative, the first character of the result is a minus sign; otherwise, the first character is a digit. If the argument is zero, the result is 0E0.

The length of the result is 24. The result includes the smallest number of characters that can represent the value of the argument such that the mantissa consists of a single digit, other than zero, followed by a period and a sequence of digits.

If the number of characters in the result is less than 24, the result is padded on the right with blanks. The CCSID of the result is determined from the application encoding scheme.

Decimal floating-point to Character:

➔ CHAR(*decimal-floating-point-expression*) ➔

decimal-floating-point-expression

An expression that returns a value that is a built-in decimal floating-point data type (DECFLOAT).

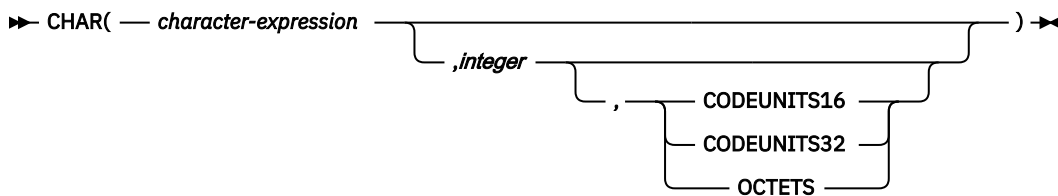
The result is the fixed-length character string representation of the argument in the form of an SQL decimal floating-point constant.

If the result value is Infinity, sNaN, or NaN, the strings 'INFINITY', 'SNAN', and 'NAN', respectively, are returned. The DECFLOAT special value sNaN does not result in an exception when converted to a string.

The length of the result is 42. If the number of characters in the result is less than 42, the result is padded on the right with blanks. Trailing zeros are significant. If the argument is negative, the first character of the result is a minus sign. Otherwise, the first character is a digit, or a letter if the result value is Infinity, sNaN, or NaN.

The CCSID of the result is determined from the application encoding scheme.

Character to Character:



character-expression

An expression that returns a value of a built-in character string.

integer

The length attribute for the resulting fixed-length character string. The value must be an integer constant between 1 and 255.

If the length is not specified, the length attribute of the result is the minimum of 255 and the length attribute of *character-expression*. If *character-expression* is an empty string constant, an error occurs.

CODEUNITS16, CODEUNITS32, or OCTETS

Specifies the unit that is used to express *integer*. If *character-expression* is a character string that is defined as bit data, **CODEUNITS16** and **CODEUNITS32** cannot be specified.

CODEUNITS16

Specifies that *integer* is expressed in terms of 16-bit UTF-16 code units.

CODEUNITS32

Specifies that *integer* is expressed in terms of 32-bit UTF-32 code units.

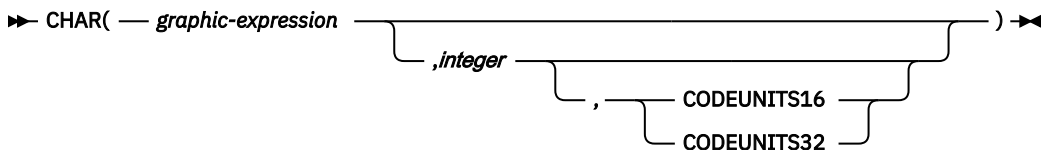
OCTETS

Specifies that *integer* is expressed in terms of bytes.

The actual length is the same as the length attribute of the result. If the length of *character-expression* is less than the length attribute of the result, the result is padded with blanks to the length of the result. If the length of *character-expression* is greater than the length attribute of the result, the result is truncated. Unless all of the truncated characters are blanks, a warning is returned.

If *character-expression* is bit data, the result is bit data. Otherwise, the CCSID of the result is the same as the CCSID of *character-expression*.

Graphic to Character:



graphic-expression

An expression that returns a value of a built-in graphic string.

integer

The length attribute for the resulting fixed-length character string. The value must be an integer constant between 1 and 255.

If the length is not specified, the length attribute of the result is the minimum of 255 and the length attribute of *graphic-expression*. The length attribute of *graphic-expression* is $(3 * \text{length}(\text{graphic-expression}))$. If *graphic-expression* is an empty string constant, an error occurs.

CODEUNITS16 or CODEUNITS32

Specifies the unit that is used to express *integer*.

CODEUNITS16

Specifies that *integer* is expressed in terms of 16-bit UTF-16 code units.

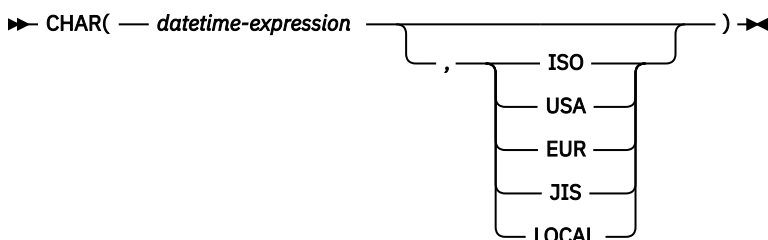
CODEUNITS32

Specifies that *integer* is expressed in terms of 32-bit UTF-32 code units.

The actual length is the same as the length attribute of the result. If the length of *graphic-expression* is less than the length attribute of the result, the result is padded with blanks to the length of the result. If the length of *graphic-expression* is greater than the length attribute of the result, the result is truncated. Unless all of the truncated characters are blanks, a warning is returned.

The CCSID of the result is the character mixed CCSID that corresponds to the graphic CCSID of *graphic-expression*.

Datetime to Character:



datetime-expression

An expression that is one of the following built-in data types:

date

The result is the character string representation of the date in the format that is specified by the second argument. If the second argument is omitted, the DATE precompiler option, if one is provided, otherwise field DATE FORMAT on installation panel DSNTIP4 specifies the format. If the format is **LOCAL**, field LOCAL DATE LENGTH on installation panel DSNTIP4 specifies the length of the result. Otherwise, the length of the result is 10.

LOCAL denotes the local format at the Db2 subsystem that executes the SQL statement. If **LOCAL** is used for the format, a date exit routine must be installed at that Db2 subsystem.

An error occurs if the second argument is specified and is not a valid value.

time

The result is the character string representation of the time in the format that is specified by the second argument. If the second argument is omitted, the TIME precompiler option, if one is provided, otherwise field TIME FORMAT on installation panel DSNTIP4 specifies the format. If the format is **LOCAL**, the field LOCAL TIME LENGTH on installation panel DSNTIP4 specifies the length of the result. Otherwise, the length of the result is 8.

LOCAL denotes the local format at the Db2 subsystem that executes the SQL statement. If **LOCAL** is used for the format, a time exit routine must be installed at that Db2 subsystem.

An error occurs if the second argument is specified and is not a valid value.

timestamp without time zone

The result is the character string representation of the timestamp. If *datetime-expression* is a TIMESTAMP(0) value, the length of the result is 19. If *datetime-expression* is a TIMESTAMP(integer) value, the length of the result is 20+integer. Otherwise, the length of the result is 26. The second argument must not be specified.

timestamp with time zone

The result is the character string representation of the timestamp with time zone, formatted as yyyy-mm-dd-hh.mm.ss.nnnnnn±th:tm with the appropriate number of 'n' characters for the precision of the timestamp. If *datetime-expression* is a TIMESTAMP(0) WITH TIME ZONE, the length of the result is 147. If *datetime-expression* is a TIMESTAMP(integer) WITH TIME ZONE, the length of the result is 148+integer. The second argument must not be specified.

The CCSID of the result is determined from the context in which the function is invoked.

ISO, EUR, USA, JIS, or LOCAL

Specifies the date or time format of the resulting character string.

Row ID to Character:

► CHAR(*row-ID-expression*) ◄

row-ID-expression

An expression that returns a value that is a built-in row ID data type.

The result is the fixed-length character string representation of the argument. The result is bit data.

The length of the result is 40. If the length of *row-ID-expression* is less than 40, the result is padded on the right with hexadecimal zeros to a length of 40.

Examples

Example 1:

HIREDATE is a DATE column in sample table DSN8D10.EMP. When it represents the date 15 December 1976 (as it does for employee 140), the following example returns the string value '12/15/1976' in character string variable DATESTRING:

```
EXEC SQL SELECT CHAR(HIREDATE, USA)
          INTO :DATESTRING
          FROM DSN8D10.EMP
          WHERE EMPNO = '000140';
```

Example 2:

Host variable HOUR has a data type of DECIMAL(6,0) and contains a value of 50000. Interpreted as a time duration, this value is 5 hours. Assume that STARTING is a TIME column in some table. Then, when STARTING represents 17 hours, 30 minutes, and 12 seconds after midnight, the following example returns the value '10:30 PM':

```
CHAR(STARTING+:HOURS, USA)
```


Example 3:

Assume that RECEIVED is defined as a TIMESTAMP column in table TABLEY. When the value of the date portion of RECEIVED represents the date 10 March 1997 and the time portion represents 6 hours and 15 seconds after midnight, the following example returns the string value '1997-03-10-06.00.15.000000':

```
SELECT CHAR(RECEIVED)
FROM TABLEY
WHERE INTCOL = 1234;
```

Example 4:

For sample table DSN8D10.EMP, the following SQL statement sets the host variable AVERAGE, which is defined as CHAR(33), to the character string representation of the average employee salary.

```
EXEC SQL SELECT CHAR(AVG(SALARY))
INTO :AVERAGE
FROM DSN8D10.EMP;
```

With DEC31, the result of AVG applied to a decimal number is a decimal number with a precision of 31 digits. The only host languages in which such a large decimal variable can be defined are Assembler and C. For host languages that do not support such large decimal numbers, use the method shown in this example.

Example 5:

For the rows in sample table DSN8D10.EMP, return the values in column LASTNAME, which is defined as VARCHAR(15), as a fixed-length character string and limit the length of the results to 10 characters.

```
SELECT CHAR(LASTNAME,10)
FROM DSN8D10.EMP;
```

For rows that have a LASTNAME with a length greater than 10 characters (excluding trailing blanks), a warning that the value is truncated is returned.

Example 6:

FIRSTNAME is a VARCHAR(12) column in a Unicode table T1. One of its values is the 6-character string 'Jürgen'. When FIRSTNAME has the values shown under 'Function', the results are shown under 'Returns':

Function ...	Returns ...
CHAR(FIRSTNAME,3,CODEUNITS32)	'Jüɀ' -- x'4AC3BC7220202020202020'
CHAR(FIRSTNAME,3,CODEUNITS16)	'Jüɀ' -- x'4AC3BC722020202020'
CHAR(FIRSTNAME,3,OCTETS)	'Jü' -- x'4AC3BC'

Example 7

For the rows in sample table DSN8D10.EMP, return the values in column EDLEVEL, which is defined as SMALLINT, as a fixed-length character string.

```
SELECT CHAR(EDLEVEL)
FROM DSN8D10.EMP;
```

An EDLEVEL of 18 is returned as CHAR(6) value '18 ' (18 followed by four blanks).

Example 8:

In sample table DSN8D10.EMP, the SALARY column is defined as DECIMAL(9,2). For those employees who have a salary of 52750.00, return the hire date and the salary, using a comma as the decimal character in the salary (52750,00).

```
SELECT HIREDATE, CHAR(SALARY, ',')
FROM DSN8D10.EMP
WHERE SALARY = 52750.00;
```

The salary is returned as the string value '52750,00'.

Example 9:

Repeat the scenario in Example 8 except subtract the SALARY column from 60000.00 and return the salary with the default decimal character.

```
SELECT HIREDATE, CHAR (60000.00 - SALARY)
FROM DSN8D10.EMP
WHERE SALARY = 52750.00;
```

The salary is returned as the string value '7250.00'.

Example 10:

Assume that host variable SEASONS_TICKETS is defined as INTEGER and has a value of 10000. Use the DECIMAL and CHAR functions to change the value into the character string '10000.00'.

```
SELECT CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
FROM SYSIBM.SYSDUMMY1;
```

Example 11:

Assume that columns COL1 and COL2 in table T1 are both defined as REAL and that T1 contains a single row with the values 7.1E+1 and 7.2E+2 for the two columns. Add the two columns and represent the result as a character string.

```
SELECT CHAR(COL1 + COL2)
FROM T1;
```

The result is the character value '1.43E2 '.

CEILING

The CEILING function returns the smallest integer value that is greater than or equal to the argument.

►► **CEILING(*numeric-expression*)** ◄◄

The argument must be an expression that returns a value of any built-in numeric data type.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The result of the function has the same data type and length attribute as the argument except that the scale is 0 if the argument is DECIMAL. For example, an argument with a data type of DECIMAL(5,5) results in DECIMAL(5,0).

The result can be null; if the argument is null, the result is the null value.

Examples

Example 1

The following statement shows the use of CEILING on positive and negative values:

```
SELECT CEILING(3.5), CEILING(3.1), CEILING(-3.1), CEILING(-3.5)
FROM SYSIBM.SYSDUMMY1;
```

This example returns: 04., 04., -03., -03.

Example 2:

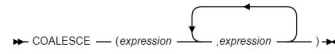
Using sample table DSN8D10.EMP, find the highest monthly salary for all the employees. Round the result up to the next integer. The SALARY column has a decimal data type.

```
SELECT CEILING(MAX(SALARY)/12)
FROM DSN8D10.EMP;
```

This example returns 04396. because the highest paid employee is Christine Haas who earns \$52750.00 per year. Her average monthly salary before applying the CEIL function is 4395.83.

COALESCE

The function COALESCE returns the value of the first non-null expression.



The arguments can be of either a built-in or user-defined data type.

The COALESCE function cannot be used as a source function when creating a user-defined function.

The arguments are evaluated in the order in which they are specified, and the result of the function is the first argument that is not null.

The selected argument is converted, if necessary, to the attributes of the result. If the COALESCE function has more than two arguments, the rules are applied to the first two arguments to determine a candidate result type. The rules are then applied to that candidate result type and the third argument to determine another candidate result type. This process continues until all arguments are analyzed and the final result type is determined.

If there are any mixed character string or graphic string and numeric arguments, the string value is implicitly cast to a DECFLOAT(34) value.

The COALESCE function can also handle a subset of the functions provided by CASE expressions. The result of using COALESCE (e1, e2) is the same as using the expression:

```
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END
```

VALUE can be specified as a synonym for COALESCE.

Example 1

The following example returns the value of the first nonnull expression between *DATEOFJOIN* and *CURRENT_DATE* from the table *EMPLOYEE*.

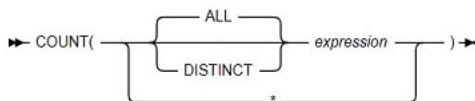
```
SELECT COALESCE (DATEOFJOIN, CURRENT_DATE) AS JOINDATE FROM EMPLOYEE;
```

The above example returns the following.

```
JOINDATE
2019-11-25
2020-01-06
2019-10-10
2019-11-27
2019-11-25
2019-11-25
2020-01-06
```

COUNT

The COUNT function returns the number of rows or values in a set of rows or values.



The argument values can be of any built-in data type other than a BLOB, CLOB, DBCLOB, or XML.

If the argument of COUNT(*) is a set of rows, the result would be the number of rows in the set. Any row that includes only null values is included in the count.

If the argument of COUNT(expression) or COUNT(ALL expression) is a set of values, the function is applied to the set of values derived from the argument values by the elimination of null values. The result is the number of nonnull values in the set, including duplicates.

If the argument of COUNT(DISTINCT expression) is a set of values, the function is applied to the set of values derived from the argument values by the elimination of null values and redundant duplicate values. The result is the number of different nonnull values in the set.

Example

The following statement counts the number of rows from the table EMPLOYEE.

```
SELECT COUNT(*) FROM EMPLOYEE
```

The above example returns 7.

CONCAT

The function CONCAT combines two compatible string arguments.

►► CONCAT(string-expression-1 , string-expression-2) ◀◀

Either argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

The result of the function is a string that consists of the first string followed by the second string.

The CONCAT function is identical to the CONCAT operator.

Example:

The following example combines strings available in the columns *EMPL_NAME* and *DEPT* from the table EMPLOYEE.

```
SELECT CONCAT(EMPL_NAME , ' ', DEPT) AS EMPL_DEPT FROM EMPLOYEE
```

The above example returns the following.

```
EMPL_DEPT
BUMRAH SALES
CEASAR MARKETING
ELICA CUST SUPR
DEV ADMIN
NEWTON DATA
SAMUEL IT
```

DATE

The DATE function returns a date that is derived from a value.

►► DATE(expression) ◀◀

The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or any numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and it must have one of the following values:
 - A valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes.

- A character or graphic string with an actual length of 7 that represents a valid date in the form `yyyynnn`, where `yyyy` are digits denoting a year and `nnn` are digits between 001 and 366 denoting a day of that year.
- If expression is a number, it must be greater than or equal to one and less than or equal to 3652059.

If expression is not a DATE value, expression is cast as follows:

- If expression is a `TIMESTAMP WITH TIME ZONE` value, expression is cast to `TIMESTAMP WITHOUT TIME ZONE`, with the same precision as expression.
- If expression is a string, expression is cast to `DATE`.

The result of the function is a date.

The other rules depend on the data type of the argument:

- **If the argument is a timestamp**, the result is the date part of the timestamp.
- **If the argument is a date**, the result is that date.
- **If the argument is a number**, the result is the date that is `n-1` days after January 1, 0001, where `n` is the integral part of the number.
- **If the argument is a string**, the result is the date that is represented by the string. If the string contains a time zone, the time zone is ignored. If the CCSID of the string is not the same as the corresponding default CCSID at the server, the string is first converted to that CCSID.

The result CCSID is the appropriate CCSID of the argument encoding scheme and the result subtype is the appropriate subtype of the CCSID.

Example 1:

The following example selects records of employees who joined later than `2019-11-10` from the table `EMPLOYEE`.

```
SELECT * FROM EMPLOYEE WHERE DATEOFJOIN > '2019-11-10'
```

The above example returns the following.

```
EMPL_NAME, AGE, DEPT, INDUSTRY, SALARY, DATEOFJOIN
BUMRAH , 53 , SALES , RETAIL , 50000 , 2019-11-25
DEV , 33 , ADMIN , RETAIL , 25000 , 2019-11-27
NEWTON , 53 , DATA , RETAIL , 75000 , 2019-11-25
SAMUEL , 43 , IT , BANKING , 75000 , 2019-11-25
```

DAY

The `DAY` function returns the day part in the given argument.

►► `DAY(expression)` ◄◄

The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or any numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes.
- If expression is a number, it must be a date duration or a timestamp duration.

If expression is a timestamp with a time zone value, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules for the function depend on the data type of the argument:

- **If the argument is a date, timestamp, or string representation of either**, the result is the day part of the value, which is an integer between 1 and 31.
- **If the argument is a date duration or timestamp duration**, the result is the day part of the value, which is an integer between -99 and 99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example:

The following example returns the day part in the given string *DATEOFJOIN* from the table *EMPLOYEE*:

```
SELECT DAY(DATEOFJOIN) FROM EMPLOYEE
```

The above example returns the following.

```
25
10
27
25
25
```

DAYOFWEEK

The DAYOFWEEK function returns an integer, in the range of 1 to 7, that represents the day of the week, where 1 is Sunday and 7 is Saturday. The DAYOFWEEK function is similar to the DAYOFWEEK_ISO function.

►► DAYOFWEEK(*expression*) ◄◄

The schema is SYSIBM.

The argument must be an expression that returns a value of one of the following built-in data types: a date, a timestamp, a character string, or a graphic string.

If *expression* is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes.

If *expression* is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

Examples for DAYOFWEEK

The result can be null; if the argument is null, the result is the null value.

Example 1

The following statement uses sample table DSN8D10.EMP, set the integer host variable DAY_OF_WEEK to the day of the week that Christine Haas (EMPNO = '000010') was hired (HIREDATE). The result is that DAY_OF_WEEK is set to 6, which represents Friday.

```
SELECT DAYOFWEEK(HIREDATE)
       INTO :DAY_OF_WEEK
       FROM DSN8D10.EMP
       WHERE EMPNO = '000010';
```

Example 2

```
SELECT DAYOFWEEK(CAST('10/11/1998' AS DATE)),
       DAYOFWEEK(TIMESTAMP('10/12/1998', '01.02')),
```

```
DAYOFWEEK(CAST(CAST('10/11/1998' AS DATE) AS CHAR(20))),
DAYOFWEEK(CAST(TIMESTAMP('10/12/1998', '01.02') AS CHAR(26)))
FROM SYSIBM.SYSDUMMY1;
```

Example 3

The following invocations of the DAYOFWEEK function all return the value 5, which represents Thursday. (1 represents Sunday for DAYOFWEEK results.) When the input argument contains a time zone, the result is determined from the UTC representation of the input value. The string representations of the example timestamp with time zone values in the SELECT statement all have the same UTC representation: 2003-01-02-20.00.00.

```
SELECT DAYOFWEEK('2003-01-02-20.00.00'),
       DAYOFWEEK('2003-01-02-12.00.00-08:00'),
       DAYOFWEEK('2003-01-03-05.00.00+09:00')
FROM SYSIBM.SYSDUMMY1;
```

DAYOFYEAR

The DAYOFYEAR function returns an integer, in the range of 1 to 366, that represents the day of the year, where 1 is January 1.

► DAYOFYEAR(*expression*) ◄

The argument must be an expression that returns a value of one of the following built-in data types: a date, a timestamp, a character string, or a graphic string.

If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes.

If expression is a timestamp with a time zone value, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

Example 1:

The following example selects the day part from the given value *DATEOFJOIN* from the table *EMPLOYEE*:

```
SELECT DAYOFYEAR(DATEOFJOIN) FROM EMPLOYEE
```

The above example returns the following.

```
329
283
331
329
329
```

DECIMAL

The DECIMAL function returns a decimal representation of either a number or a character-string or graphic-string representation of a number, an integer, or a decimal number.

Numeric to Decimal:

► DECIMAL(*numeric-expression*) ◄

String to Decimal:

Numeric to decimal

numeric-expression

An expression that returns a value of any built-in numeric data type.

precision

An integer constant with a value greater than or equal to 1 and less than or equal to 31.

The default for precision depends on the data type of the numeric-expression:

- 5 for small integer
- 11 for large integer
- 19 for big integer
- 15 for floating point or decimal
- 31 for decimal floating point

scale

An integer constant that is greater than or equal to zero and less than or equal to precision. The value specifies the scale of the result. The default value is 0.

The result of the function is the same number that would occur if the argument were assigned to a decimal column or variable with precision p and scale s , where p and s are specified by the second and third arguments. An error occurs if the number of significant digits required to represent the whole part of the number is greater than $p - s$.

String to decimal

string-expression

An expression that returns a value of a character or graphic string (except a CLOB or DBCLOB) with a length attribute that is not greater than 255 bytes. The string must contain a valid string representation of a number. Leading and trailing blanks are removed from the string, and the resulting substring must conform to the rules for forming a valid string representation of an SQL integer or decimal constant.

precision

An integer constant with a value in the range 1 to 31. The value of this second argument specifies the precision of the result. If not specified, the default is 15.

scale

An integer constant that is greater than or equal to zero and less than or equal to precision. The value specifies the scale of the result. The default value is 0.

decimal-character

A single-byte character constant used to delimit the decimal digits in string-expression from the whole part of the number. The character cannot be a digit, plus (+), minus (-), or blank. The default value is period (.) or comma (,); the default value cannot be used in string-expression if a different value for decimal-character is specified.

Digits are truncated from the end of the decimal number if the number of digits to the right of the decimal separator character is greater than the scale s . An error is returned if the number of significant digits to the left of the decimal character (the whole part of the number) in string-expression is greater than $p - s$.

The result of the function is a decimal number with precision of p and scale of s , where p and s are the second and third arguments.

Notes

Syntax alternatives:

To increase the portability of applications when the precision is specified, use the CAST specification.

Example

Represent the average salary of the employees in the table `EMPLOYEE` as an 8-digit decimal number with two of these digits to the right of the decimal point.

```
SELECT DECIMAL (AVG (SALARY) , 8, 2)
FROM EMPLOYEE;
```

The above example returns `49000.00`

DELETE

The DELETE statement deletes rows from a table or a view. The table or view can be at the current server or Data Virtualization Manager server with which the current server can establish a connection. There are two forms of this statement:

- The searched DELETE form is used to delete one or more rows, optionally determined by a search condition.
- The positioned DELETE form specifies that one or more rows corresponding to the current cursor position are to be deleted.

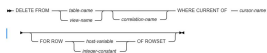
searched delete:



Note:

- If the period-clause is specified, the fetch-clause must not be specified.
- The same clause must not be specified more than one time.

positioned delete:



Example:

The following example delete a record that matches the condition `EMPL_NAME = 'DAVE'` from the table `EMPLOYEE`.

```
DELETE FROM EMPLOYEE
WHERE EMPL_NAME = 'DAVE'
```

DOUBLE

The DOUBLE functions returns a floating-point representation of either a number or a character-string or graphic-string representation of a number, an integer, a decimal number, or a floating-point number.

Numeric to Double

►► DOUBLE — (*numeric-expression*) ◄◄

numeric-expression

An expression that returns a value of any built-in numeric data type.

The result is the same number that would occur if the expression were assigned to a double precision floating-point column or variable.

String to Double

► DOUBLE — (*string-expression*) ◄

string-expression

An expression that returns a value of a character or graphic string (except a CLOB or DBCLOB) with a length attribute that is not greater than 255 bytes. The string must contain a valid string representation of a number.

The result is the same number that would result from `CAST(string-expression AS DOUBLE PRECISION)`. Leading and trailing blanks are removed from the string, and the resulting substring must conform to the rules for forming a valid string representation of an SQL floating-point, integer, or decimal constant.

The result of the function is a double precision floating-point number.

The result can be null; if the argument is null, the result is the null value.

Example

Using sample table DSN8D10.EMP, find the ratio of salary to commission for employees whose commission is not zero. The columns involved in the calculation, SALARY and COMM, have decimal data types. To eliminate the possibility of out-of-range results, apply the DOUBLE function to SALARY so that the division is carried out in floating-point.

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
FROM DSN8D10.EMP
WHERE COMM > 0;
```

EXISTS

The EXISTS predicate tests for the existence of certain rows. The fullselect can specify any number of columns, and can result in true or false.

► EXISTS (*fullselect*) ◄

Notes:

- The outer SELECT list of fullselect must not contain an array value.

The result of the EXISTS predicate:

- Is true only if the number of rows that is specified by the fullselect is not zero.
- Is false only if the number of rows specified by the fullselect is zero.
- Cannot be unknown.

The SELECT clause in the fullselect can specify any number of columns because the values returned by the fullselect are ignored. For convenience, use: `SELECT *`

Unlike NULL, LIKE, and IN predicates, the EXISTS predicate has no form that contains the word NOT. To negate an EXISTS predicate, precede it with the logical operator NOT, as: `NOT EXISTS (fullselect)`

The result is then false if the EXISTS predicate is true, and true if the predicate is false. Here, NOT is a logical operator and not a part of the predicate.

Example 1:

The following example compares the values in the column *EMPL_NAME* of the table *EMPLOYEE* with the values in the column *EMPL_NAME* of the table *EMPL_COMP* and returns the record that has a similar name with age >20.

```
SELECT T1.EMPL_ID, T1.EMPL_NAME FROM EMPL_COMP T1
WHERE EXISTS
SELECT T2.AGE FROM EMPLOYEE T2 WHERE T1.EMPL_NAME = T2.EMPL_NAME AND T2.AGE > 20)
```

The above example returns the following.

<i>Table 41. Exists</i>	
EMPL_ID	EMPL_NAME
2	ELICA

FULL OUTER JOIN

The FULL OUTER JOIN clause results in the inclusion of rows from two tables. If a value is missing when rows are joined, that value is null in the resultant table.

The join condition for a full outer join must be a search condition that compares two columns. The predicates of the search condition can be combined only with AND. Each predicate must have the form 'expression = expression'.

Example:

The following query performs a full outer join of the *EMPLOYEE* and *EMPL_COMP* tables:

```
SELECT A.EMPL_NAME, B.EMPL_ID, A.DEPT, A.INDUSTRY, B.COMPANY, B.LOCATION
FROM EMPLOYEE A
FULL OUTER JOIN EMPL_COMP B
ON
A.EMPL_NAME = B.EMPL_NAME
```

The result table looks like this:

<i>Table 42. Full Outer Join</i>					
EMPL_NAME	EMPL_ID	DEPT	INDUSTRY	COMPANY	LOCATION
	1			TIMBER INC.	
BUMRAH		SALES	RETAIL		
CEASAR		MARKETING	RETAIL		
DEV		ADMIN	RETAIL		
ELICA	2	CUST SUPR	RETAIL	STANLEY INC.	TX
ELICA	2			STANLEY INC.	TX
NEWTON		DATA	RETAIL		

FLOAT

The FLOAT function returns a floating-point representation of either a number or a string representation of a number.

►► FLOAT(*numeric-expression*) ◄◄

Notes

Syntax alternatives:

FLOAT is a synonym for DOUBLE_PRECISION or DOUBLE.

Example:

The following example returns the floating point representation of the values in the *AGE* from the table *EMPLOYEE*.

```
SELECT FLOAT(AGE) FROM EMPLOYEE
```

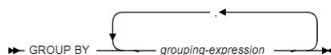
The above example returns the following.

```
53.0  
43.0  
23.0  
33.0  
53.0  
43.0
```

GROUP-BY

The GROUP BY clause specifies a result table that consists of a grouping of the rows of intermediate result table that is the result of the previous clause.

group-by-clause



In its simplest form, a GROUP BY clause contains a grouping-expression.

grouping-expression

A grouping-expression is an expression that defines the grouping of R. The following restrictions apply to grouping-expression:

- If grouping-expression is a single column, the column name must unambiguously identify a column of R.
- The result of grouping-expression cannot be a LOB data type (or a distinct type that is based on a LOB) or an XML data type.
- grouping-expression cannot include any of the following items:
 - A correlated column
 - A host variable
 - An aggregate function
 - Any function or expression that is not deterministic or that is defined to have an external action
 - A scalar fullselect
 - A CASE expression whose searched-when-clause contains a quantified predicate, an IN predicate using a fullselect, or an EXISTS predicate

The result of GROUP BY is a set of groups of rows. In each group of more than one row, all values of each grouping-expression are equal, and all rows with the same set of values of the grouping-expression are in the same group.

If a grouping-expression contains DECFLOAT values, the DECFLOAT values with the same value will be in the same group. But the number of digits returned for each group is unpredictable.

Because every row of a group contains the same value of any grouping-expression, a grouping-expression can be used in a search condition in a HAVING clause or an expression in a SELECT clause, or in a

sort-key-expression of an ORDER BY clause. In each case, the reference specifies only one value for each group. For example, if grouping-expression is col1+col2, col1+col2+3 would be an allowed expression in the select list. Associative rules for expressions do not allow the similar expression of 3+col1+col2, unless parentheses are used to ensure that the corresponding expression is evaluated in the same order. Thus, 3+(col1+col2) would also be allowed in the select list. If the concatenation operator is used, grouping-expression must be used exactly as the expression was specified in the select list.

If a grouping-expression contains varying-length strings with trailing blanks, the values in the group can differ in the number of trailing blanks and might not all have the same length. In that case, a reference to grouping-expression still specifies only one value for each group, but the value for a group is chosen arbitrarily from the available set of values. Thus, the actual length of the result value is unpredictable.

Example:

The following example groups the record by *INDUSTRY* from the table EMPLOYEE.

```
SELECT COUNT(DEPT), INDUSTRY FROM EMPLOYEE
GROUP BY INDUSTRY
```

The above example returns the following.

<i>Table 43. Group-By</i>	
	INDUSTRY
1	BANKING
5	RETAIL

GROUP_CONCAT

The GROUP_CONCAT function returns a string with concatenated non-NULL value from a group.

The GROUP_CONCAT function is used to concatenate and aggregate values from multiple rows within a specific column into a single string. It's useful for combining and displaying related data in a compact format.

➡ GROUP_CONCAT(*expression*) ➡

For the GROUP_CONCAT function we can set the maximum output length of the column using the server parameter or the SET statement else it would be defaulted to 32K. When the length is exceeded, the data is truncated.

You can use the following SET statement:

```
SET GROUP_CONCAT_MAX = value
```

This will set the maximum length for the life of the current session or until another SET statement changes it again.

Or,

You can use the following server parameter to set the maximum output length:

Parameter	Description
SQLLENGCATMAXLEN	The SQLLENGCATMAXLEN controls the maximum length of the output for a GROUP_CONCAT function. Set to zero for the global default, which is the maximum varchar length minus 32. The value can be overridden at the session level via a SET command or a VTB rule at the table level.

- The primary purpose of the GROUP_CONCAT function is to concatenate values from multiple rows into a single string.
- You can use GROUP_CONCAT to aggregate data based on a certain column or attribute.
- The GROUP_CONCAT can be used to create a list of tags for each record.
- When working with hierarchical data structures, you can use GROUP_CONCAT() to show parent-child relationships in a readable format.
- In applications, you can use GROUP_CONCAT to display a user's preferences, settings, or selected options in a user-friendly format.
- GROUP_CONCAT is useful for displaying data in applications, reports, or user interfaces where a single field needs to show multiple related values.
- The function allows you to define custom separators (other than commas) and order concatenated values, giving you flexibility in how data is presented.

Example:

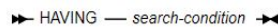
The following example concatenates values from the table EMPLOYEE:

```
SELECT GROUP_CONCAT(EMPL_NAME) FROM EMPLOYEE;
```

HAVING

The HAVING clause specifies a result table that consists of those groups of the intermediate result table for which the search-condition is true. The intermediate result table is the result of the previous clause. If this clause is not GROUP BY, the intermediate result table is considered a single group with no grouping columns of the previous clause of the subselect.

having-clause



Each column-name in search-condition must be one of the following:

- Unambiguously identify a grouping column of the intermediate result table
- Be specified within an aggregate function
- Be a correlated reference. A column-name is a correlated reference if it identifies a column of a table, view, common-table-expression, or nested-table-expression that is identified in an outer subselect

A group of the intermediate result table to which the search condition is applied supplies the argument for each function in the search condition, except for any function whose argument is a correlated reference.

If the search condition contains a subquery, the subquery can be thought of as being executed each time the search condition is applied to a group of the intermediate result table, and the results used in applying the search condition. In actuality, the subquery is executed for each group only if it contains a correlated reference.

A correlated reference to a group of the intermediate result table must either identify a grouping column or be contained within an aggregate function.

When HAVING is used without GROUP BY, any expression or column name in the select list must appear within an aggregate function.

Example:

The following example groups the records by *INDUSTRY* column from the table EMPLOYEE and returns the records with count ≥ 1 .

```
SELECT COUNT(DEPT) AS COUNT, INDUSTRY
FROM EMPLOYEE
GROUP BY INDUSTRY
HAVING COUNT(DEPT) >= 1
```

The above example returns the following.

Table 44. Having	
COUNT	INDUSTRY
1	BANKING
5	RETAIL

HEX

The HEX function returns a hexadecimal representation of a value.

► HEX(*expression*) ◄

The argument must be an expression that returns a value of any built-in data type that is not XML. A character or binary string must not have a maximum length greater than 16352. A graphic string must not have a maximum length greater than 8176.

The result of the function is a character string.

The result is a string of hexadecimal digits. The first two represent the first byte of the argument, the next two represent the second byte of the argument, and so forth. If the argument is a datetime value, the result is the hexadecimal representation of the internal form of the argument.

If the argument is a fixed-length string and the length of the result is less than 255, the result is a fixed-length string. Otherwise, the result is a varying-length string with a length attribute that depends on the following considerations:

- **If the argument is not a varying-length string**, the length attribute of the result string is the same as the length of the result.
- **If the argument is a varying-length character or binary string**, the length attribute of the result string is twice the length attribute of the argument.
- **If the argument is a varying-length graphic string**, the length attribute of the result string is four times the length attribute of the argument.

If expression returns string data, the CCSID of the result is the SBCS CCSID that corresponds to the CCSID of expression. Otherwise, the CCSID of the result is determined from the context in which the function was invoked.

If the argument is a graphic string, the length of the result is four times the maximum length of the argument. Otherwise, the length of the result is twice the (maximum) length of the argument.

Example:

The following example returns the hexadecimal value of the column *AGE* from the table [EMPLOYEE](#).

```
SELECT HEX(AGE) FROM EMPLOYEE
```

The above example returns the following:

```
00000035  
0000002B  
00000017  
00000021  
00000035  
0000002B
```

HOUR

The HOUR function returns the hour part of the given argument.

The argument must be an expression that returns a value of one of the following built-in data types: a time, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a time or timestamp duration.

If expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a time, timestamp, or string representation of either**, the result is the hour part of the value, which is an integer between 1 and 24.
- **If the argument is a time duration or timestamp duration**, the result is the hour part of the value, which is an integer between -99 and +99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example:

The following example returns the hour part in the string *UPDATED_TS* from the table *EMPL_COMP*.

```
SELECT HOUR(UPDATED_TS) FROM EMPL_COMP
```

The above example returns 5.

IFNULL

The IFNULL function returns the first nonnull expression.

IFNULL is identical to the COALESCE scalar function except that IFNULL is limited to two arguments instead of multiple arguments.

Example:

The following example returns 0 if the value in the *SALARY* column from the table *EMPLOYEE* is NULL.

```
SELECT EMPL_NAME, IFNULL(SALARY,0)
FROM EMPLOYEE;
```

The above example returns the following.

```
EMPL_NAME,
BUMRAH , 50000
CEASAR , 20000
ELICA , 0
DEV , 25000
NEWTON , 75000
SAMUEL , 75000
ELICA , 0
```

INNER JOIN

You can use an inner join in a SELECT statement to retrieve only the rows that satisfy the join conditions on every specified table.

You can request an inner join, by running a SELECT statement in which you specify the tables that you want to join the FROM clause and specify a WHERE clause or an ON clause to indicate the join condition. The join condition can be any simple or compound search condition that does not contain a subquery reference.

In the simplest type of inner join, the join condition is *column1=column2*.

Example:

The following example joins the records with similar employee name from the tables EMPLOYEE and EMPL_COMP.

```
SELECT A.EMPL_NAME, B.EMPL_ID, A.DEPT, A.INDUSTRY, B.COMPANY, B.LOCATION
FROM EMPLOYEE A
INNER JOIN EMPL_COMP B
ON
A.EMPL_NAME = B.EMPL_NAME "
```

The above example returns the following.

Table 45. Inner Join

EMPL_NAME	EMPL_ID	DEPT	INDUSTRY	COMPANY	LOCATION
ELICA	2	CUST SUPER	RETAIL	STANLEY INC.	TX
ELICA	2			STANLEY INC.	TX

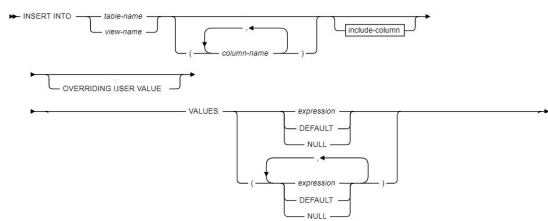
INSERT

The INSERT statement inserts rows into a table or view. The table or view can be at the current server or any Data Virtualization Manager server with which the current server can establish a connection.

The INSERT via VALUES form is used to insert a single row into the table or view using the values provided or referenced.

The owner of a view, unlike the owner of a table, might not have INSERT authority on the view (or can have INSERT authority without being able to grant it to others). The nature of the view itself can preclude its use for INSERT.

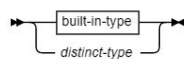
Syntax



include-column:



data-type:



Example:

The example inserts the values (0003,'SMITH','CHROME INC.','WA') to the table EMPLOYEE:

```
INSERT INTO EMPL_COMP (EMPL_ID, EMPL_NAME, COMPANY, LOCATION) VALUES(0003, 'SMITH', 'CHROME INC.', 'WA');
```

The following is the updated table after the record insertion.

Table 46. Insert

EMPL_ID	EMPL_NAME	COMPANY	LOCATION	UPDATED_TS
1	AKHIL	TIMBER INC.	NY	2019-11-26 05:39:59.454707
2	ELICA	STANLEY INC.	TX	
3	SMITH	CHROME INC.	WA	

LARGE INTEGER (INTEGER)

A *large integer* is a binary integer with a precision of 31 bits.

The range of large integers is -2147483648 to +2147483647.

Example:

The following example selects the values in the *SALARY* column of the table EMPLOYEE as an integer.

```
SELECT CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

The above example returns the following.

```
50000  
20000  
  
25000  
75000  
75000
```

LEFT

The LEFT function returns a string that consists of the specified number of leftmost bytes of the specified string units.

Character string:

► LEFT(character-expression ,length) ◄

The LEFT function returns the leftmost string of character-expression, graphic-expression, or binary-expression consisting of length of the string units that are specified implicitly or explicitly.

character-expression

An expression that specifies the string from which the result is derived. The string must be a character string. A substring of character-expression is zero or more contiguous code points of character-expression.

The string can contain mixed data. Depending on the units that are specified to evaluate the function, the result is not necessarily a properly formed mixed data character string.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

length

An expression that specifies the length of the result. The value must be an integer between 0 and n, where n is the length attribute of character-expression, expressed in the units that are either implicitly or explicitly specified.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34) which is then assigned to an INTEGER value.

The character-expression, graphic-expression, or binary-expression is effectively padded on the right with the necessary number of padding characters so that the specified substring of the expression always exists. The encoding scheme of the data determines the padding character:

- For ASCII SBCS data or ASCII mixed data, the padding character is X'20'.
- For ASCII DBCS data, the padding character depends on the CCSID; for example, for Japanese (CCSID 301) the padding character is X'8140', while for simplified Chinese it is X'A1A1'.
- For EBCDIC SBCS data or EBCDIC mixed data, the padding character is X'40'.
- For EBCDIC DBCS data, the padding character is X'4040'.
- For Unicode SBCS data or UTF-8 (Unicode mixed data), the padding character is X'20'.
- For UTF-16 (Unicode DBCS) data, the padding character is X'0020'.
- For binary data, the padding character is X'00'.

The result of the function is a varying-length string with a length attribute that is the same as the length attribute of the first expression and a data type that depends on the data type of the expression:

- VARCHAR if character-expression is CHAR or VARCHAR
- CLOB if character-expression is CLOB
- VARGRAPHIC if graphic-expression is GRAPHIC or VARGRAPHIC
- DBCLOB if graphic-expression is DBCLOB
- VARBINARY if binary-expression is BINARY or VARBINARY
- BLOB if binary-expression is BLOB

The actual length of the result is determined from length.

The CCSID of the result is the same as that of the first expression.

Example:

Assume that the column *EMPL_NAME* has a value "Richard". The following example returns "Ric" which are the three leftmost characters.

```
SELECT LEFT(EMPL_NAME, 3) FROM EMPLOYEE
```

LEFT OUTER JOIN

The LEFT OUTER JOIN clause lists rows from the left table even if there are no matching rows on right table.

As in an inner join, the join condition of a left outer join can be any simple or compound search condition that does not contain a subquery reference

Example:

Consider the following example using the records in the table EMPLOYEE:

```
SELECT A.EMPL_NAME, B.EMPL_ID, A.DEPT, A.INDUSTRY, B.COMPANY, B.LOCATION
FROM EMPLOYEE A
LEFT OUTER JOIN EMPL_COMP B
ON
A.EMPL_NAME = B.EMPL_NAME
```

The result table looks like the following example:

EMPL_NAME	EMPL_ID	DEPT	INDUSTRY	COMPANY	LOCATION
BUMRAH		SALES	RETAIL		
CEASAR		MARKETING	RETAIL		
ELICA	2	CUST SUPR	RETAIL	STANLEY INC.	TX
DEV		ADMIN	RETAIL		
NEWTON		DATA	RETAIL		
SAMUEL		IT	BANKING		
ELICA	2			STANLEY INC.	TX

LENGTH

The LENGTH function returns the length of a value.

➔ LENGTH(expression) ➔

The argument must be an expression that returns a value of any built-in data type that is not XML.

The result of the function is a large integer.

The result is the length of the argument. The length of a varying-length string is the actual length, not the maximum length.

The length of a graphic string is the number of double-byte characters. Unicode UTF-16 data is treated as graphic data; a UTF-16 supplementary character takes two DBCS characters to represent and as such is counted as two DBCS characters.

The length of all other values is the number of bytes used to represent the value:

- 2 for small integer
- 4 for large integer
- 8 for big integer
- The integer part of $(p/2)+1$ for decimal numbers with precision p
- 16 for DECFLOAT(34)
- 8 for DECFLOAT(16)
- 4 for single precision floating-point
- 8 for double precision floating-point
- The length of the string for strings
- 4 for DATE
- 3 for TIME
- 10 for TIMESTAMP
- 12 for TIMESTAMP WITH TIME ZONE
- $7+((p+1)/2)$ for TIMESTAMP(p)
- $9+((p+1)/2)$ for TIMESTAMP(p) WITH TIME ZONE
- The length of the row ID

Example 1:

The following example assigns the length of the string available in the column *EMPL_NAME* from the table *EMPLOYEE* to the variable *NAME_LENGTH*:

```
SELECT LENGTH(EMPL_NAME) AS NAME_LENGTH FROM EMPLOYEE
```

The above example returns the following.

```
NAME_LENGTH
6
6
5
3
6
6
5
```

LIKE

The LIKE predicate searches for strings that have a certain pattern.

→ match-expression — NOT — LIKE — pattern-expression — ESCAPE — escape-expression →

The match-expression is the string to be tested for conformity to the pattern specified in pattern-expression. Underscore and percent sign characters in the pattern have a special meaning instead of their literal meanings unless escape-expression is specified.

The following example returns the rows where the *INDUSTRY* column value contains the string *RET* from the table *EMPLOYEE*.

```
SELECT * FROM EMPLOYEE WHERE INDUSTRY LIKE 'RET%'
```

The above example returns the following:

Table 48. Like

EMPL_NAME	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	RETAIL	50000	11/25/2019
CEASER	43	RETAIL	20000	
ELICA	23	RETAIL	5000	10/10/2019
DEV	33	RETAIL	25000	11/27/2019
NEWTON	53	RETAIL	75000	11/25/2019

LOWER

The LOWER function returns a string in which all the characters are converted to lowercase characters.

→ LOWER(string-expression) →

string-expression

An expression that specifies the string to be converted. string-expression must return a value that is a built-in character or graphic string. A character string argument must not be a CLOB, and a graphic string argument must not be a DBCLOB. If string-expression is an EBCDIC graphic string, a blank string must not be specified for locale-name-string. If string-expression is bit data, locale-name-string must not be specified.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

Syntax alternatives:

LCASE is a synonym for LOWER. LOWER should be used for conformance to the SQL standard.

Example

The following example returns the characters in the variable *EMPL_NAME* in lowercase from the table *EMPLOYEE*.

```
SELECT LOWER(EMPL_NAME) FROM EMPLOYEE
```

The above example returns the following:

```
bumrah  
ceasar  
elica  
dev  
newton  
samuel  
elica
```

LTRIM

The LTRIM function removes spaces from the beginning of a string expression.

► LTRIM (— string-expression)

The LTRIM function removes all of the spaces that are contained in the left side of the given string-expression.

string-expression

An expression that specifies the source string. The argument must be an expression that returns a value that is a built-in string data type that is not a LOB, or a numeric data type. If the value is not a string data type, it is implicitly cast to VARCHAR before the function is evaluated. If string-expression is not FOR BIT DATA, trim-expression must not be FOR BIT DATA.

The result of the function depends on the data type of string-expression:

- VARCHAR if string-expression is a character string. If string-expression is defined as FOR BIT DATA the result is FOR BIT DATA.
- VARGRAPHIC if string-expression is a graphic string.
- VARBINARY if string-expression is a binary string.

The length attribute of the result is the same as the length attribute of string-expression.

Example:

The following example removes the spaces from the left side of the values in the *EMPL_NAME* from the table *EMPLOYEE*:

```
SELECT LTRIM(EMPL_NAME) FROM EMPLOYEE
```

The above example returns the following.

```
BUMRAH  
CEASAR  
ELICA  
DEV  
NEWTON
```

MAX

The MAX scalar function returns the maximum value in a set of values.

The diagram shows the syntax for the MAX function: `MAX(expression, expression)`. A curved arrow points from the first `expression` to the second `expression`, indicating that the function evaluates multiple arguments.

All but the first argument can be parameter markers. There must be two or more arguments.

Each argument must be an expression that returns a value of any built-in data type other than a CLOB, DBCLOB, BLOB, ROWID, or XML.

Character string arguments and binary string arguments cannot have a length attribute greater than 32704, and graphic string arguments cannot have a length attribute greater than 16352.

The arguments are evaluated in the order in which they are specified. The result of the function is the maximum argument value.

The selected argument is converted, if necessary, to the attributes of the result. If the MAX function has more than two arguments, the rules are applied to the first two arguments to determine a candidate result type. The rules are then applied to that candidate result type and the third argument to determine another candidate result type. This process continues until all arguments are analyzed and the final result type and CCSID is determined.

Notes

Syntax alternatives:

GREATEST is a synonym for MAX.

Example 1:

The following example selects the maximum value among the values in the *SALARY* column from the table *EMPLOYEE*.

```
SELECT MAX(SALARY) FROM EMPLOYEE
```

The above example returns *75000*.

MICROSECOND

The MICROSECOND function returns the microsecond part of a value.

The diagram shows the syntax for the MICROSECOND function: `MICROSECOND(expression)`. A double-headed arrow points to the `expression` parameter.

The argument must be an expression that returns a value of one of the following built-in data types: a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a timestamp duration.

If the expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a timestamp or string representation of a timestamp**, the result is the microsecond part of the value, which is an integer between 0 and 999999. If the precision of the timestamp exceeds 6, the value is truncated.

- **If the argument is a duration**, the result is the microsecond part of the value, which is an integer between -999999 and 999999. A nonzero result has the same sign as the argument.

Example 1:

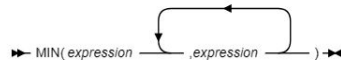
The following example returns the microsecond part of the string in the column *UPDATED_TS* in the table *EMPL_COMP*:

```
SELECT MICROSECOND(UPDATED_TS) FROM EMPL_COMP WHERE UPDATED_TS IS NOT NULL
```

The above example returns *454707*.

MIN

The MIN scalar function returns the minimum value in a set of values.



All but the first argument can be parameter markers. There must be two or more arguments.

Each argument must be an expression that returns a value of any built-in data type other than a CLOB, DBCLOB, BLOB, ROWID, or XML.

Character string arguments and binary string arguments cannot have a length attribute greater than 32704, and graphic string arguments cannot have a length attribute greater than 16352.

The arguments are evaluated in the order in which they are specified. The result of the function is the minimum argument value.

The selected argument is converted, if necessary, to the attributes of the result. If the MIN function has more than two arguments, the rules are applied to the first two arguments to determine a candidate result type. The rules are then applied to that candidate result type and the third argument to determine another candidate result type. This process continues until all arguments are analyzed and the final result type and CCSID is determined.

Notes

Syntax alternatives:

LEAST is a synonym for MIN.

Example 1:

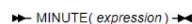
The following example selects the minimum value among the values in the *SALARY* column from the table *EMPLOYEE*

```
SELECT MIN(SALARY) FROM EMPLOYEE
```

The above example returns *20000*.

MINUTE

The MINUTE function returns the minute part of a value.



The argument must be an expression that returns a value of one of the following built-in data types: a time, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a time or timestamp duration.

If expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a time, timestamp, or string representation of either**, the result is the minute part of the value, which is an integer between 0 and 59.
- **If the argument is a time duration or timestamp duration**, the result is the minute part of the value, which is an integer between -99 and 99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example 1:

The following example selects the minute part of the string available in the *UPDATED_TS* column in the table *EMPL_COMP*.

```
SELECT MINUTE(UPDATED_TS) FROM EMPL_COMP WHERE UPDATED_TS IS NOT NULL
```

The above example returns 39.

MOD

The MOD function divides the first argument by the second argument and returns the remainder.

► MOD(*numeric-expression-1*, *numeric-expression-2*) ►

The formula used to calculate the remainder is:

$$\text{MOD}(x, y) = x - \text{FLOOR}(x/y) * y$$

Where x/y is the truncated integer result of the division. The result is negative only if the first argument is negative.

Each argument must be an expression that returns a value of any built-in numeric data type.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The attributes of the result are based on the arguments as follows:

- If both arguments are large or small integers, the data type of the result is large integer.
- If both arguments are integers and at least one argument is a big integer, the data type of the result is big integer.
- If one argument is an integer and the other is a decimal, the data type of the result is decimal with the same precision and scale as the decimal argument.
- If both arguments are decimal, the data type of the result is decimal. The precision of the result is $\min(p-s, p'-s')$ and the scale of the result is $\max(s, s')$,
 - where the symbols p and s denote the precision and scale of the first argument, and the symbols p' and s' denote the precision and scale of the second argument.
 - If one argument is a floating-point number, and the other is not a DECFLOAT, or both argument is a floating-point number, the data type of the result is double precision floating-point.

The operation is performed in floating-point. If necessary, the operands are first converted to double precision floating-point numbers. For example, an operation that involves a floating-point number and either an integer or a decimal number is performed with a temporary copy of the integer or decimal number that has been converted to double precision floating-point. The result of a floating-point operation must be within the range of floating-point numbers.

- If either argument is a DECFLOAT, the data type of the result is DECFLOAT(34).

If either argument is a special decimal floating point value, the general rules for arithmetic operations apply.

If one argument is a DECFLOAT and the second argument is zero, the result is NaN and an invalid operation condition is returned.

Example:

The following example returns the remainder of dividing the value available in the *SALARY* column from the table *EMPLOYEE* by 3.

```
SELECT MOD(SALARY, 3) FROM EMPLOYEE
```

The above example returns the following.

```
2
2
1
0
0
```

MONTH

The MONTH function returns the month part of a value.

► MONTH(*expression*) ◄

The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a date or timestamp duration.

If expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a date, timestamp, or string representation of either**, the result is the month part of the value, which is an integer between 1 and 12.
- **If the argument is a date duration or timestamp duration**, the result is the month part of the value, which is an integer between -99 and 99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example 1:

The following example selects the month part in the value available in the *UPDATED_TS* column in the table *EMPLOYEE*:

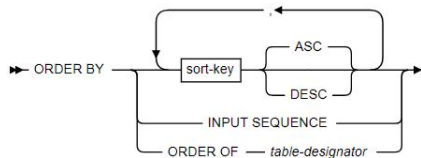
```
SELECT MONTH(UPDATED_TS) FROM EMPL_COMP WHERE UPDATED_TS IS NOT NULL
```

The above example returns *11*

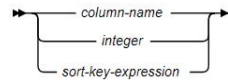
ORDER BY

The ORDER BY clause specifies an ordering of the rows of the result table.

order-by-clause



sort-key:



A subselect that contains an ORDER BY clause cannot be specified in the outermost fullselect of a view.

If the subselect is not enclosed within parentheses and is not the outermost fullselect, the ORDER BY clause cannot be specified. The ORDER BY clause cannot be used in an outermost fullselect that contains a FOR UPDATE clause.

An ORDER BY clause that is specified in a subselect only affects the order of the rows that are returned by the query if the subselect is the outermost fullselect, except when a nested subselect includes an ORDER BY clause and the outermost fullselect specifies that the ordering of the rows should be retained (by using the ORDER OF table-designator clause).

Multiple ORDER BY clauses can be specified in the same subselect if each clause is separated with parentheses.

INPUT SEQUENCE

Indicates that the result table reflects the input order of the rows specified in the VALUES clause of an INSERT statement. INPUT SEQUENCE ordering can be specified only when an INSERT statement is specified in a from-clause.

ORDER OF table-designator

Specifies that the same ordering of the rows for the result table that is designated by table-designator should be applied to the result table of the subselect (or fullselect) that contains the ORDER OF specification. There must be a table reference in the FROM clause of the subselect (or fullselect) that specifies this clause and matches table-designator.

For an ORDER BY clause in an OLAP specification, table-designator must not specify a table function, a collection-derived table, a materialized view, a nested table expression that is materialized, an alias, or a synonym.

sort-key

A column-name, integer, or sort-key-expression that specifies the value that is to be used to order the rows of the result of the subselect.

If a single sort-key is identified, the rows are ordered by the values of that sort-key. If more than one sort-key is identified, the rows are ordered by the values of the first sort-key, then by the values of the second sort-key, and so on. A sort-key cannot be a LOB or XML expression.

The result table can be ordered by a named column in the select list by specifying a sort-key that is an integer or the column name. The result table can be ordered by an unnamed column in the select list by specifying a sort-key that is an integer or, in some cases, by a sort-key-expression that matches the expression in the select list.

column-name

An identifier that usually identifies a column of the result table. In this case, column-name must be the name of a named column in the select list. If the fullselect includes a set operator, the column name cannot be qualified.

If the query is a subselect, the column-name can also identify a column name of a table, view, or nested table expression identified in the FROM clause, including a column that is defined as implicitly hidden. The subselect must not include any of the following:

- DISTINCT in the select list
- Aggregate functions in the select list
- GROUP BY clause

integer

An unsigned integer that must be greater than 0 and not greater than the number of columns in the result table. The integer *n* identifies the *n*th column of the result table.

sort-key-expression

An expression that is not simply a column-name or unsigned integer constant. The query to which ordering is applied must be a subselect to use this form of the sort-key.

The sort-key-expression cannot include an expression that is not deterministic or a function that is defined to have an external action except for the RID built-in function and the ROW CHANGE expression. If sort-key-expression includes an aggregate function, the input arguments to that function must not reference a named column in the select list that is derived from an aggregate function.

If DISTINCT is used in the select list of the subselect, sort-key-expression must match an expression in the select list of the subselect. Scalar-fullselects are never matched.

If the subselect is grouped, the sort-key-expression might or might not be in the select list of the subselect. When sort-key-expression is not in the select list the following rules apply:

- Each expression in the ORDER BY clause must either:
 - Use one or more grouping expressions
 - Use a column name that either unambiguously identifies a grouping column of R or is specified within a aggregate function.
- Each expression in the ORDER BY clause must not contain a scalar fullselect.

ASC

Uses the values of the sort-key in ascending order.

ASC is the default.

DESC

Uses the values of the sort-key in descending order.

The null value is higher than all other values. If your ordering specification does not determine a complete ordering, rows with duplicate values of the last identified sort-key have an arbitrary order. If you do not specify ORDER BY, the rows of the result table have an arbitrary order.

Column access controls do not effect the operation of the ORDER BY clause. The order is based on the original column values. However, after column masks are applied, the masked values in the final result table might not reflect the order of the original column values.

Column names in sort keys: A column name in a sort-key must conform to the following rules:

- If the column name is qualified, the query must be a subselect. The column name must unambiguously identify a column of a table, view, or nested table expression in the FROM clause of the subselect; its value is used to compute the value of the sort specification.
- If the column name is unqualified and the query is a subselect:
 - If the column name is identical to the name of more than one column of the result table, the column name must unambiguously identify a column of some table, view, or nested table expression in the FROM clause of the ordering subselect.
 - If the column name is identical is one column of the result table, its value is used to compute the value of the sort specification.
 - If the column name is not identical to a column in the result table, it must unambiguously identify a column of a table, view, or nested table expression in the FROM clause of the subselect. If the column name is identical to one column of a table, view, or nested table expression in the FROM clause of the subselect, its value is used to compute the value of the sort specification.

Example:

The following example sorts the rows from the table `EMPLOYEE` in an ascending order.

```
SELECT EMPL_NAME, AGE FROM EMPLOYEE ORDER BY AGE ASC
```

The above example returns the following:

```
EMPL_NAME , AGE
ELICA , 23
DEV , 33
CEASAR , 43
SAMUEL , 43
BUMRAH , 53
NEWTON , 53
ELICA ,
```

OUTER JOIN

An outer join is a method of combining two or more tables so that the result includes unmatched rows of one of the tables, or of both tables. The matching is based on the join condition.

Data Virtualization Manager supports three types of outer joins:

full outer join

Includes unmatched rows from both tables. If any column of the result table does not have a value, that column has the null value in the result table.

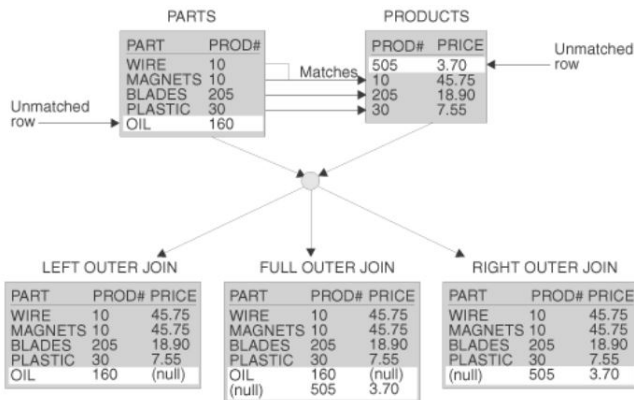
left outer join

Includes rows from the table that is specified before `LEFT OUTER JOIN` that have no matching values in the table that is specified after `LEFT OUTER JOIN`.

right outer join

Includes rows from the table that is specified after `RIGHT OUTER JOIN` that have no matching values in the table that is specified before `RIGHT OUTER JOIN`.

The following table illustrates how the sample `PARTS` and `PRODUCTS` tables can be combined using the three outer join functions.



The result table contains data that is joined from all of the tables, for rows that satisfy the search conditions.

The result columns of a join have names if the outermost `SELECT` list refers to base columns. However, if you use a function (such as `COALESCE` or `VALUE`) to build a column of the result, that column does not have a name unless you use the `AS` clause in the `SELECT` list.

RAND

The RAND function returns a random floating-point value between 0 and 1. An argument can be specified as an optional seed value.

➤ RAND(numeric-expression) ➤

numeric-expression

If numeric-expression is specified, it is used as the seed value. The argument must be an expression that returns a value of a built-in integer data type (SMALLINT or INTEGER). The value must be between 0 and 2,147,483,646.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34) and then assigned to an INTEGER value.

The result of the function is a double precision floating-point number.

A specific seed value, other than zero, will produce the same sequence of random numbers for a specific instance of a RAND function in a query each time the query is executed. The seed value is used only for the first invocation of an instance of the RAND function within a statement. RAND(0) is processed the same as RAND().

The RAND function is a non-deterministic function.

Example:

The following example assigns a random number for the column *EMPL_ID* in the table EMPLOYEE.

```
INSERT INTO EMPL_COMP (EMPL_ID, EMPL_NAME) VALUES(RANDOM(), 'KANE');
```

The above example assigns a random value for the *EMPL_ID* column for the employee *KANE* as shown below.

```
EMPL_ID, EMPL_NAME, COMPANY, LOCATION, UPDATED_TS
1 , AKHIL , TIMBER INC. , NY , 2019-11-26 05:39:59.454707
2 , ELICA , STANLEY INC. , TX ,
3 , SMITH , CHROME INC. , WA ,
1919574970 , KANE , , ,
```

REAL

The REAL function returns a single-precision floating-point representation of either a number or a string representation of a number.

Numeric to Real:

➤ REAL(*numeric-expression*) ➤

String to Real:

➤ REAL(*string-expression*) ➤

Numeric to Real

numeric-expression

An expression that returns a value of any built-in numeric data type.

The result is the same number that would occur if the argument were assigned to a single precision floating-point column or variable. If the numeric value of the argument is not within the range of single precision floating-point, an error occurs.

String to Real

string-expression

An expression that returns a value of a character or graphic string (except a CLOB or DBCLOB) with a length attribute that is not greater than 255 bytes. The string must contain a valid string representation of a number.

The result is the same number that would result from `CAST(string-expression AS REAL)`. Leading and trailing blanks are eliminated and the resulting string must conform to the rules for forming an SQL floating-point, integer, or decimal constant.

The result of the function is a single precision floating-point number.

Notes

Syntax alternatives:

To increase the portability of applications, use the `CAST` specification.

Examples

Example 1:

Using the sample table `EMPLOYEE`, find the ratio of salary to age for employees. The columns involved, `SALARY` and `AGE`, have decimal data types. To express the result in single precision floating-point, apply `REAL` to `SALARY` so that the division is carried out in floating-point (actually double precision) and then apply `REAL` to the complete expression so that the results are returned in single precision floating-point.

```
SELECT EMPL_NAME, REAL(SALARY/AGE) FROM EMPLOYEE
WHERE AGE > 0;
```

The above example returns the following.

```
EMPL_NAME,
BUMRAH , 943.396240234375
CEASAR , 465.1162109375
ELICA ,
DEV , 757.57568359375
NEWTON , 1415.09423828125
SAMUEL , 1744.18603515625
```

REPLACE

The `REPLACE` function replaces all occurrences of a search-string in a source-string with a replace-string. If the search-string is not found in the source-string, the source-string is returned unchanged.

► REPLACE ((source-string , search-string) replace-string) ►

source-string

An expression that specifies the source string. The expression must return a value that is a built-in character string, graphic string, or binary string data type that is not a LOB and it cannot be an empty string.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a `VARCHAR` data type.

search-string

An expression that specifies the string to be removed from the source string. The expression must return a value that is a built-in character string, graphic string, or binary string data type that is not a LOB; the value cannot be an empty string.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a `VARCHAR` data type.

replace-string

An expression that specifies the replacement string. The expression must return a value that is a built-in character string, graphic string, or binary string data type that is not a LOB.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

If replace-string is not specified or is an empty string, nothing replaces the string that is removed from the source string.

The actual length of each string must be 32764 bytes or less for character and binary strings or 16382 or less for graphic strings.

All three arguments must have compatible data types. If the expressions have different CCSID sets, then the expressions are converted to the CCSID set of source-string.

The data type of the result of the function depends on the data type of source-string, search-string, and replace-string:

- VARCHAR if source-string is a character string. The encoding scheme of the result is the same as source-string. The CCSID of the result depends on the arguments:
 - If source-string, search-string, or replace-string is bit data, the result is bit data.
 - If source-string, search-string, and replace-string are all SBCS Unicode data, the CCSID of the result is the CCSID for SBCS Unicode data.
 - If source-string is SBCS Unicode data, and search-string or replace-string is not SBCS Unicode data, the CCSID of the result is the mixed CCSID for Unicode data.
 - Otherwise, the CCSID of the result is the mixed CCSID that corresponds to the CCSID of source-string. However, if the input is EBCDIC or ASCII and there is no corresponding system CCSID for mixed, the CCSID of the result is the CCSID of source-string.
- VARGRAPHIC if source-string is a graphic. The encoding scheme of the result is the same as source-string. The CCSID of the result is the same as the CCSID of source-string.
- VARBINARY if source-string, search-string, and replace-string are binary strings.

The length attribute of the result depends on the arguments:

- If the length attribute of replace-string is less than or equal to the length attribute of search-string, the length attribute of the result is the length attribute of source-string.
- If the length attribute of replace-string is greater than the length attribute of search-string, the length attribute of the result is determined as follows depending on the data type of the result:
 - For VARCHAR or VARBINARY:
 - If $L1 \leq 4000$, the length attribute of the result is $\text{MIN}(4000, (L3 * (L1/L2)) + \text{MOD}(L1, L2))$
 - Otherwise, the length attribute of the result is $\text{MIN}(32764, (L3 * (L1/L2)) + \text{MOD}(L1, L2))$
 - For VARGRAPHIC:
 - If $L1 \leq 2000$, the length attribute of the result is $\text{MIN}(2000, (L3 * (L1/L2)) + \text{MOD}(L1, L2))$
 - Otherwise, the length attribute of the result is $\text{MIN}(16382, (L3 * (L1/L2)) + \text{MOD}(L1, L2))$

where:

- L1 is the length attribute of source-string
- L2 is the length attribute of search-string if the search string is a string constant. Otherwise, L2 is 1.
- L3 is the length attribute of replace-string

If the result is a character string or binary string, the length attribute of the result must not exceed 32764. If the result is a graphic string, the length attribute of the result must not exceed 16382.

The actual length of the result is the actual length of source-string plus the number of occurrences of search-string that exist in source-string multiplied by the actual length of replace-string minus the actual length of search-string. If the actual length of the result string exceeds the maximum for the return data type, an error occurs.

Example:

The following example replace all occurrences of *CEASAR* with *GEORGE* in the string available in the column *EMPL_NAME* from the table *EMPLOYEE*.

```
SELECT REPLACE(EMPL_NAME, 'CEASAR', 'GEORGE') FROM EMPLOYEE
```

The above example returns the following.

```
BUMRAH
GEORGE
ELICA
DEV
NEWTON
SAMUEL
ELICA
```

RIGHT OUTER JOIN

The RIGHT OUTER JOIN clause lists rows from the right table even if there are no matching rows on left table.

As in an inner join, the join condition of a right outer join can be any simple or compound search condition that does not contain a subquery reference

Example

The following example joins the tables [Table 36 on page 162](#) and [Table 37 on page 162](#).

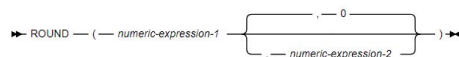
```
SELECT A.ID, A.NAME, A.DEPT, A.JOB, A.YEARS, A.SALARY, A.COMM
FROM JOIN1;
RIGHT OUTER JOIN
JOIN2 B;
ON A.ID=B.ID
```

The query returns the following.

<i>Table 49. RIGHT OUTER JOIN</i>						
ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
999	JOHN		SALES		16.99	
10	MIC	20	MGR	7	98357	0.00
20	WILLIAM	20	SALES	8	78171	612.45
30	MIC	38	MGR	5	77506	0.00
40	OBRIEN	38	SALES	6	78006	846.55

ROUND

The ROUND function returns a number that is rounded to the specified number of places to the right or left of the decimal place.



numeric-expression-1

An expression that returns a value of any built-in numeric data type.

If expression-1 is a decimal floating-point data type, the DECFLOAT ROUNDING MODE will not be used. The rounding behavior of ROUND corresponds to a value of ROUND_HALF_UP. If you want a different rounding behavior, use the QUANTIZE function.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

numeric-expression-2

An expression that returns a value that is a built-in numeric, character string or graphic string data type. If the value is not of type INTEGER, it is implicitly cast to INTEGER before evaluating the function.

The absolute value of integer specifies the number of places to the right of the decimal point for the result if numeric-expression-2 is not negative. If numeric-expression-2 is negative, numeric-expression-1 is rounded to the sum of the absolute value of numeric-expression-2+1 number of places to the left of the decimal point.

If the absolute value of numeric-expression-2 is larger than the number of digits to the left of the decimal point, the result is 0. (For example, ROUND(748.58, -4) returns 0.)

If numeric-expression-1 is positive, a digit value of 5 is rounded to the next higher positive number. If numeric-expression-1 is negative, a digit value of 5 is rounded to the next lower negative number.

The result of the function has the same data type and length attribute as the first argument except that the precision is increased by one if the argument is DECIMAL and the precision is less than 31. For example, an argument with a data type of DECIMAL(5,2) results in DECIMAL(6,2). An argument with a data type of DECIMAL(31,2) results in DECIMAL(31,2).

Example 1:

The following example rounds the number available in the SALARY column in the table EMPLOYEE.

```
SELECT ROUND(SALARY) FROM EMPLOYEE
```

The above example returns the following.

```
50000
20000

25000
75000
75000
```

RTRIM

The RTRIM function removes spaces from the right side of a string expression.

► RTRIM — (— string-expression)

The RTRIM function removes all of the spaces contained at the right side of a string-expression.

string-expression

An expression that specifies the source string. The argument must be an expression that returns a value that is a built-in string data type that is not a LOB, or a numeric data type. If the value is not a string data type, it is implicitly cast to VARCHAR before the function is evaluated. If string-expression is not FOR BIT DATA, trim-expression must not be FOR BIT DATA.

The result of the function depends on the data type of string-expression.

- VARCHAR if string-expression is a character string. If string-expression is defined as FOR BIT DATA, the result is FOR BIT DATA.
- VARCHAR if string-expression is a character string. If string-expression is defined as FOR BIT DATA, the result is FOR BIT DATA.
- VARGRAPHIC if string-expression is a graphic string.
- VARBINARY if string-expression is a binary string.

The length attribute of the result is the same as the length attribute of string-expression.

Example:

The following example removes spaces at the right side of the values in the *EMPL_NAME* from the table **EMPLOYEE**:

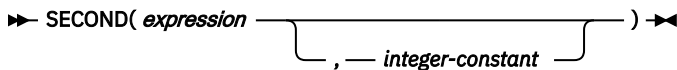
```
SELECT RTRIM(EMPL_NAME) FROM EMPLOYEE
```

The above example returns the following.

```
BUMRAH
CEASAR
ELICA
DEV
NEWTON
SAMUEL
ELICA
```

SECOND

The SECOND function returns the seconds part of a value with optional fractional seconds.



expression

expression must be an expression that returns a value of one of the following built-in data types: a time, a timestamp, a character string, a graphic string, or a numeric data type.

- If *expression* is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time or timestamp with an actual length that is not greater than 255 bytes.
- If *expression* is a number, it must be a time or timestamp duration.

integer-constant

integer-constant must be an integer constant that represents the scale for the fractional seconds portion of *expression*. The value must be in the range 0 through 12. If *integer-constant* is not specified, the result does not include fractional seconds.

If *expression* is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function with a single argument is a large integer. The result of the function with two arguments is DECIMAL(2+s,s) where s is the value of *integer-constant*.

The result can be null; if the first argument is null, the result is the null value.

The other rules depend on the data type of the argument:

- If the argument is a time, timestamp, or string representation of a time or a timestamp:
 - The result is the seconds part of the value (0 to 59) and any fractional seconds that are included in the value. If the second argument is specified, the result includes *integer-constant* digits of the fractional seconds part of the value where applicable. If there are no fractional seconds in the value, zeros are returned.
- If the argument is a time duration or timestamp duration:

The result is the seconds part of the value (-99 to 99) and any fractional seconds that are included in the value. If the second argument is specified, the result includes *integer-constant* digits of the fractional seconds part of the value where applicable. If there are no fractional seconds in the value, zeros are returned. A nonzero result has the same sign as the *expression*.

Examples

Example 1:

Assume that the variable TIME_DUR is declared in a PL/I program as DECIMAL(6,0) and can therefore be interpreted as a time duration. When TIME_DUR has the value 153045, the following function returns the value 45.

```
SECOND(:TIME_DUR)
```

Example 2:

Assume that RECEIVED is a TIMESTAMP column and that one of its values is the internal equivalent of '1988-12-25-17.12.30.000000'. The following function returns the value 30.

```
SECOND(RECEIVED)
```

Example 3:

The following invocations of the SECOND function returns the same result:

```
SELECT SECOND('2003-01-02-20.10.05.123456'),
       SECOND('2003-01-02-12.10.05.123456-08:00'),
       SECOND('2003-01-03-05.10.05.123456+09:00')
FROM SYSIBM.SYSDUMMY1;
```

For each invocation of the SECOND function in this SELECT statement, the result is 5.

When the input argument contains a time zone, the result is determined from the UTC representation of the input value. The string representations of a timestamp with a time zone in the SELECT statement all have the same UTC representation: 2003-01-02-20.10.05.123456. The second portion of the UTC representation is 5.

Example 4:

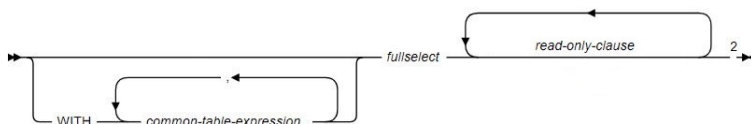
Return the seconds with fractional seconds from a current timestamp with milliseconds.

```
SELECT SECOND(CURRENT_TIMESTAMP(3),3)
FROM SYSIBM.SYSDUMMY1;
```

The SELECT statement returns a DECIMAL(5,3) value that is based on the current timestamp and could be something like 54.321.

SELECT

The SELECT statement made up a series of clauses that are defined by SQL as being executed in a logical order. SELECT statements allow users to definite and organize information that is retrieved from a specified table.



Note:

- The read-only-clause must not be specified if an update-clause is specified.
- The same clause must not be specified more than one time.

The tables and the view identified in a select statement can be at the current server or any subsystem with which the current server can establish a connection.

For local queries or remote queries, if a table is encoded as ASCII or Unicode, the retrieved data is encoded in EBCDIC.

A select statement can implicitly or explicitly invoke user-defined functions or implicitly invoke stored procedures. This technique is known as nesting of SQL statements. A function or procedure is implicitly invoked in a select statement when it is invoked at a lower level. For instance, if you invoke a user-defined function from a select statement and the user-defined function invokes a stored procedure, you are implicitly invoking the stored procedure.

- **common-table-expression**

A common table expression defines a result table with table-identifier that can be referenced in any FROM clause of the fullselect that follows.

- **read-only-clause**

The read-only clause specifies that the result table is read-only. Therefore, the cursor cannot be referred to in positioned UPDATE or DELETE statements.

Example**SELECT - All Rows**

The following example selects all rows from the table EMPLOYEE.

```
SELECT * FROM EMPLOYEE;
```

Result

<i>Table 50. SELECT ALL</i>					
EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					

SELECT - Limited Columns and Row

The following example selects only the columns EMPL_NAME, AGE, and limits the records to 3.

```
SELECT EMPL_NAME, AGE
FROM EMPLOYEE LIMIT 3;
```

Result

<i>Table 51. SELECT - Limited Columns and Row</i>	
EMPL_NAME	AGE
BUMRAH	53
CEASAR	43
ELICA	23

SELECT - Condition

The following example applies the condition of selecting only the record where the EMPL_NAME has LOCATION = 'TX' in the table EMPL_COMP.

```
SELECT EMPL_NAME, AGE, DEPT, INDUSTRY, SALARY, DATEOFJOIN
FROM EMPLOYEE
WHERE EMPL_NAME = (SELECT EMPL_NAME FROM EMPL_COMP WHERE LOCATION = 'TX');
```

Result

<i>Table 52. SELECT - Condition</i>					
EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
ELICA					

Sub Select

The following example executes the sub select function first and selects **EMPL_NAME**, **AGE**, and **INDUSTRY** columns from the table EMPLOYEE and limits the record to 10. And then the outer SELECT function selects only the **EMPL_NAME** column from the sub select result and limits to 5 records.

```
SELECT EMPL_NAME FROM (SELECT EMPL_NAME, AGE, INDUSTRY, FROM EMPLOYEE LIMIT 10) LIMIT 5
```

Result

<i>Table 53. SUB SELECT</i>
EMPL_NAME
BUMRAH
CEASAR
ELICA
DEV
NEWTON
SAMUEL

SMALLINT

The SMALLINT function returns a small integer representation either of a number or of a string representation of a number.

Numeric to Smallint:

►► SMALLINT(*numeric-expression*) ◄◄

String to Smallint:

➤ `SMALLINT(string-expression)` ➤

Numeric to Smallint

numeric-expression

An expression that returns a value of any built-in numeric data type.

The result is the same number that would occur if the argument were assigned to a small integer column or variable. If the whole part of the argument is not within the range of small integers, an error occurs. If present, the decimal part of the argument is truncated.

String to Smallint

string-expression

An expression that returns a value of character or graphic string (except a CLOB or DBCLOB) with a length attribute that is not greater than 255 bytes for a character string or 127 for a graphic string. The string must contain a valid string representation of a number.

The result is the same number that would result from `CAST(string-expression AS SMALLINT)`. Leading and trailing blanks are eliminated and the resulting string must conform to the rules for forming an SQL integer constant.

The result of the function is a small integer.

The result can be null; if the argument is null, the result is the null value.

Examples

Example 1:


Using sample table `DSN8D10.EMP`, find the average education level (`EDLEVEL`) of the employees in department 'A00'. Round the result to the nearest full education level.

```
SELECT SMALLINT(AVG(EDLEVEL)+.5)
FROM DSN8D10.EMP
WHERE DEPT = 'A00';
```

Assuming that the five employees in the department have education levels of '19', '18', '14', '18', and '14', the result is '17'.

SUBSTR

The `SUBSTR` function returns a substring of a string.

➤ `SUBSTR(string-expression ,start )` ➤

string-expression

An expression that specifies the string from which the result is derived. The string must be a character, graphic, or binary string. If `string-expression` is a character string, the result of the function is a character string. If it is a graphic string, the result of the function is a graphic string. If it is a binary string, the result of the function is a binary string.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a `VARCHAR` data type.

A substring of `string-expression` is zero or more contiguous characters of `string-expression`. If `string-expression` is a graphic string, a character is a DBCS character. If `string-expression` is a character string or a binary string, a character is a byte. The `SUBSTR` function accepts mixed data strings. However, because `SUBSTR` operates on a strict byte-count basis, the result will not necessarily be a properly formed mixed data string.

start

An expression that specifies the position within string-expression to be the first character of the result. The value of the large integer must be between 1 and the length attribute of string-expression. (The length attribute of a varying-length string is its maximum length.) A value of 1 indicates that the first character of the substring is the first character of string-expression.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34) which is then assigned to an INTEGER value.

length

An expression that specifies the length of the resulting substring. If specified, length must be an expression that returns a value that is a built-in large integer data type.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34) which is then assigned to an INTEGER value.

The value must be greater than or equal to 0 and less than or equal to n , where n is the length attribute of string-expression - start + 1. The specified length must not, however, be the large integer constant 0.

If length is explicitly specified, string-expression is effectively padded on the right with the necessary number of characters so that the specified substring of string-expression always exists. Hexadecimal zeros are used as the padding character when string-expression is binary data. Otherwise, a blank is used as the padding character.

If string-expression is a fixed-length string, omission of length is an implicit specification of $\text{LENGTH}(\text{string-expression}) - \text{start} + 1$, which is the number of characters (or bytes) from the character (or byte) specified by start to the last character (or byte) of string-expression. If string-expression is a varying-length string, omission of length is an implicit specification of the greater of zero or $\text{LENGTH}(\text{string-expression}) - \text{start} + 1$. If the resulting length is zero, the result is an empty string.

If length is explicitly specified by a large integer constant that is 255 or less, and string-expression is not a LOB, the result is a fixed-length string with a length attribute of length. If length is not explicitly specified, but string-expression is a fixed-length string and start is an integer constant, the result is a fixed-length string with a length attribute equal to $\text{LENGTH}(\text{string-expression}) - \text{start} + 1$. In all other cases, the result is a varying-length string. If length is explicitly specified by a large integer constant, the length attribute of the result is length; otherwise, the length attribute of the result is the same as the length attribute of string-expression.

The CCSID of the result is the CCSID of string-expression.

Example 1:

The following example returns the substring from the value in the *INDUSTRY* column from the table *EMPLOYEE*.

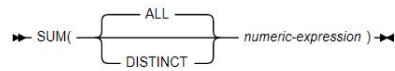
```
SELECT SUBSTR(INDUSTRY,1,3) FROM EMPLOYEE
```

The above example returns the following.

```
RET  
RET  
RET  
RET  
RET  
BAN
```

SUM

The SUM function returns the sum of a set of numbers.



The argument values can be of any built-in numeric data type, and their sum must be within the range of the data type of the result.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The data type of the result is determined as follows:

- DECFLOAT(34) if the argument is DECFLOAT(n).
- Large integer if the argument is small integer.
- Double precision floating-point if the argument is single precision floating-point.
- Otherwise, the result is the same as the data type of the argument.

If the data type of the argument values is decimal, the scale of the result is the same as the scale of the argument values, and the precision of the result depends on the precision of the argument values and the decimal precision option:

- If the precision of the argument values is greater than 15 or the DEC31 option is in effect, the precision of the result is $\min(31, P+10)$, where P is the precision of the argument values.
- Otherwise, the precision of the result is 15.

Example:

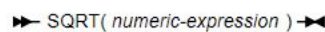
The following example selects the sum of the values in the numerical column *SALARY* from the table *EMPLOYEE*.

```
SELECT SUM(SALARY) FROM EMPLOYEE
```

The above examples returns *250000*

SQRT

The SQRT function returns the square root of the argument.



The argument must be an expression that returns the value of any built-in numeric data type. If the argument is DECFLOAT, the operation is performed in DECFLOAT. Otherwise, the argument is converted to a double precision floating-point number for processing by the functions.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

If the argument is DECFLOAT(n), the result is DECFLOAT(n). Otherwise, the result of the function is a double precision floating-point number. If the argument is a special decimal floating point value, the general rules for arithmetic operations apply.

Example:

The following example returns the square root value of the numerical column *SALARY* from the table *EMPLOYEE*.

```
SELECT SQRT(SALARY) FROM EMPLOYEE
```

The above example returns the following:

```
223.60679774997897
141.42135623730948
70.71067811865474
158.11388300841895
273.8612787525831
273.8612787525831
```

TIME

The TIME function returns a time that is derived from a value.

►► TIME(*expression*) ◄◄

The argument must be an expression that returns a value of one of the following built-in data types: a time, a timestamp, a character string, or a graphic string. If *expression* is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time or timestamp with an actual length of not greater than 255 bytes. A time zone in a string representation of a timestamp is ignored.

If *expression* is a TIMESTAMP WITH TIME ZONE value, *expression* is first cast to TIMESTAMP WITHOUT TIME ZONE, with the same precision as *expression*.

If *expression* is not a TIME value, *expression* is cast as follows:

- If *expression* is a TIMESTAMP WITH TIME ZONE value, *expression* is cast to TIMESTAMP WITHOUT TIME ZONE, with the same precision as *expression*.
- If *expression* is a string, *expression* is cast to TIME.

The result of the function is a time.

The result can be null; if the argument is null, the result is the null value.

The other rules depend on the data type of the argument:

If the argument is a time

the result is that time.

If the argument is a timestamp

the result is the time part of the timestamp.

If the argument is a string

the result is the time or time part of the timestamp represented by the string. If the CCSID of the string is not the same as the corresponding default CCSID at the server, the string is first converted to that CCSID.

The result CCSID is the appropriate CCSID of the argument encoding scheme and the result subtype is the appropriate subtype of the CCSID.

Example: Assume that a table named CLASSES contains one row for each scheduled class. Assume also that the class starting times are in the TIME column named STARTTM. Using these assumptions, select those rows in CLASSES that represent classes that start at 1:30 P.M.

```
SELECT *
  FROM CLASSES
 WHERE TIME(STARTTM) = '13:30:00';
```

TIMESTAMP

The TIMESTAMP function returns a TIMESTAMP WITHOUT TIME ZONE value from its argument or arguments.

► TIMESTAMP(*expression-1* [, *expression-2*]) ◄

The rules for the arguments depend on whether the second argument is specified.

- **If only one argument is specified:** The argument must be an expression that returns a value of one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or a binary string. If *expression-1* is a character or graphic string, it must not be a CLOB or DBCLOB and it must have one of the following values:
 - A valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes. A time zone in a string representation of a timestamp is ignored.
 - A character string or graphic string with an actual length of 8 that is assumed to be a IBM® Z Store Clock value.
 - A character string with an actual length of 13 that is assumed to be a result from the GENERATE_UNIQUE function.
 - A character string or graphic string with an actual length of 14 that represents a valid date and time in the form *yyyymmddhhmmss*, where *yyyy* is the year, *mm* is the month, *dd* is the day, *hh* is the hour, *mm* is the minute, and *ss* is the seconds.

If *expression-1* is a binary string, it must not be a BLOB and its value must be one of the following:

- A binary string with an actual length of 8 bytes that is assumed to be a IBM Z Store Clock value.
- A binary string with an actual length of 16 bytes that is assumed to be a IBM Z Store Clock extended value.

- **If both arguments are specified;**

- If the data type of the second argument is not an integer:

The first argument must be an expression that returns a value of one of the following built-in data types: a date, a character string, or a graphic string. The second argument must be an expression that returns a value of one of the following built-in data types: a time, a character string, or a graphic string. A character string or graphic string must be a valid string representation of a time.

If *expression-1* is a character string or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date with an actual length that is not greater than 255 bytes. If *expression-2* is a character string or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time with an actual length that is not greater than 255 bytes.

- If the data type of the second argument is integer:

The first argument must be an expression that returns a value of one of the following built-in data types: a timestamp, a date, a character string, or a graphic string. The second argument must be an integer constant in the range 0 to 12 that represents the timestamp precision.

If *expression-1* is a character string or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a timestamp or a date with an actual length that is not greater than 255 bytes.

If *expression-1* is a binary string, it must not be a BLOB, and its value must conform to the rules for when only one argument is specified. The second argument must be an integer constant in the range 0 to 12 that represents the timestamp precision.

The result of the function is a `TIMESTAMP WITHOUT TIME ZONE` value.

The timestamp precision and other rules depend on whether the second argument is specified:

- **If both arguments are specified and the second argument is not an integer:**

The result is a `TIMESTAMP(6) WITHOUT TIME ZONE` value with the date that is specified by the first argument and the time that is specified by the second argument. The fractional seconds part of the timestamp is zero.

If both arguments are specified and the second argument is an integer:

The result is a `TIMESTAMP WITHOUT TIME ZONE` value with the precision that is specified in the second argument.

If only one argument is specified and it is a `TIMESTAMP (p) WITHOUT TIME ZONE`:

The result is that `TIMESTAMP (p) WITHOUT TIME ZONE` value.

If only one argument is specified and it is a `TIMESTAMP(p) WITH TIME ZONE`:

The result is the argument value, cast to `TIMESTAMP(p) WITHOUT TIME ZONE`. The value is the local timestamp, not UTC.

If only one argument is specified and it is a date:

The result is that date with an assumed time of midnight that is cast to `TIMESTAMP(0) WITHOUT TIME ZONE`.

If only one argument is specified and it is a character or graphic string:

The result is the `TIMESTAMP(6) WITHOUT TIME ZONE` value that is represented by that string extended with any missing time information. If the argument is a string of length 14, the `TIMESTAMP` has a fractional seconds part of zero. The string value must not contain a specification of time zone.

If only one argument is specified and it is a binary string:

The result is the `TIMESTAMP(6) WITHOUT TIME ZONE` value that is represented by that string. If the year value in the resulting timestamp is greater than 9999 an error is returned (`SQLSTATE 22007`, `SQLCODE -180`).

If the arguments include only date information, the time information in the result value is all zeros.

If an argument is a string with a `CCSID` that is not the same as the corresponding default `CCSID` at the server, the string is first converted to that `CCSID`.

The result `CCSID` is the appropriate `CCSID` of the argument encoding scheme and the result subtype is the appropriate subtype of the `CCSID`. If both arguments are specified and their encoding schemes are different, the result `CCSID` is the appropriate `CCSID` of the application encoding scheme.

Notes**Specifying an LRSN as an argument:**

When a 6-byte LRSN is used as the argument to the `TIMESTAMP` function, it must be left justified and padded on the right to a total length of 8 bytes. When a 10-byte LRSN is used, it must be left justified and padded on the right to a total length of 16 bytes.

Syntax alternatives:

If only one argument is specified, the `CAST` specification should be used for maximal portability.

`TIMESTAMP_TZ` is a similar function.

Examples**Example:**

The following example selects the timestamp from the column `UPDATED_TS` in the table `EMPLOYEE`:

```
SELECT TIMESTAMP(UPDATED_TS) FROM EMPL_COMP
```

The above example returns `2019-11-26 05:39:59.454707`.

TO_CHAR

The `TO_CHAR` function returns a character string representation of a timestamp value that has been formatted using a specified character template.

Character to VARCHAR

►► `TO_CHAR` — (— *character-expression* —) ►►

Timestamp to VARCHAR

►► TO_CHAR — (— *timestamp-expression* — , — *format-string* —) ►►

Decimal floating-point to VARCHAR

►► TO_CHAR — (— *decimal-floating-point-expression* — , — *format-string* —) ►►

TRIM

The TRIM function removes bytes from the beginning, from the end, or from both the beginning and end of a string expression.

►► TRIM — (— *trim-constant* — FROM — *string-expression* —) ►►

The first argument, if specified, indicates whether characters are removed from the end or the beginning of the string. If the first argument is not specified, the characters are removed from both the end and the beginning of the string.

trim-constant

Specifies a constant that indicates the binary, SBCS, or DBCS character that is to be removed. If *string-expression* is a character string, *trim-constant* must be an SBCS or DBCS single-character (2 bytes) constant. If *string-expression* is a binary string, *trim-constant* must be a single-byte binary string constant. If *string-expression* is a DBCS graphic or DBCS-only string, *trim-constant* must be a graphic constant that consists of a single DBCS character.

The default for *trim-constant* depends on the data type of *string-expression*:

- A DBCS blank if *string-expression* is a DBCS graphic string. For ASCII, the CCSID determines the hex value that represents a DBCS blank. For example, for Japanese (CCSID 301), X'8140' represents a DBCS blank, while for Simplified Chinese, X'A1A1' represents a DBCS blank. For EBCDIC, X'4040' represents a DBCS blank.
- A UTF-16 or UCS-2 blank (X'0020') if *string-expression* is a Unicode graphic string.
- A value of X'00' if *string-expression* is a binary string.
- Otherwise, a single byte blank. For EBCDIC, X'40' represents a blank. When not EBCDIC, X'20' represents a blank.

string-expression

An expression that returns a value that is a built-in character string data type, graphic data type, binary string data type, or numeric data type. *string-expression* must not be a LOB. If *string-expression* is numeric, it is cast to a character string before the function is evaluated.

string-expression and *trim-expression* must have compatible data types.

The data type of the result depends on the data type of *string-expression*:

- If *string-expression* is a character string data type, the result is VARCHAR. If *string-expression* is defined as FOR BIT DATA the result is FOR BIT DATA.

- If *string-expression* is a graphic string data type, the result is VARGRAPHIC.
- If *string-expression* is a binary string data type, the result is VARBINARY.

The length attribute of the result is the same as the length attribute of *string-expression*. The actual length of the result is the length of *string-expression* minus the number of characters removed. If all of the characters are removed, the result is an empty string.

If *string-expression* can be null, the result can be null; if *string-expression* is null, the result is the null value.

The CCSID of the result is the same as that of *string-expression*.

Note:

Valid content for EBCDIC mixed string input:

If *string-expression* is an EBCDIC mixed string, the string must contain valid EBCDIC mixed data.

Examples

Example:

Assume the host variable HELLO of type CHAR(9) has a value of ' Hello '.

```
SELECT TRIM(:HELLO), TRIM(TRAILING FROM :HELLO)
FROM SYSIBM.SYSDUMMY1
```

Results in 'Hello' and ' Hello' respectively.

Example:

Assume the host variable BALANCE of type CHAR(9) has a value of '000345.50'.

```
SELECT TRIM(L '0' FROM :BALANCE)
FROM SYSIBM.SYSDUMMY1
```

Results in '345.50'

UNION

Using the UNION keyword, you can combine two or more subselects to form a fullselect.

When SQL encounters the UNION keyword, it processes each subselect to form an interim result table, it combines the interim result table of each subselect and deletes duplicate rows to form a combined result table. You can use different clauses and techniques when coding select-statements.

The combined list is derived from two tables and contains no duplicates.

To better understand the results from these SQL statements, imagine that SQL goes through the following process:

When you use UNION:

- Any ORDER BY clause must appear after the last subselect that is part of the union. The ORDER BY clause specifies that the combined result table is to be in collated sequence. ORDER BY is not allowed in a view.
- A name may be specified on the ORDER BY clause if the result columns are named. A result column is named if the corresponding columns in each of the unioned select-statements have the same name. An AS clause can be used to assign a name to columns in the select list.

To identify which subselect each row is from, you can include a constant at the end of the select list of each subselect in the union. When SQL returns your results, the last column contains the constant for the subselect that is the source of that row. For example, you can specify:

```
SELECT A, B, 'A1' ...
UNION
SELECT X, Y, 'B2' ...
```

When a row is returned, it includes a value (either A1 or B2) to indicate the table that is the source of the row's values. If the column names in the union are different, SQL uses the set of column names specified in the first subselect when interactive SQL displays or prints the results, or in the SQLDA resulting from processing an SQL DESCRIBE statement.

Note: Sort sequence is applied after the fields across the UNION pieces are made compatible. The sort sequence is used for the distinct processing that implicitly occurs during UNION processing.

The following example combines the records from the tables [Table 34 on page 161](#) and [Table 35 on page 161](#).

```
SELECT A.EMPL_NAME, A.AGE, A.DEPT, A.INDUSTRY, A.SALARY, A.DATEOFJOIN
FROM EMPLOYEE A
UNION
SELECT B.EMPL_NAME, B.AGE, B.DEPT, B.INDUSTRY, B.SALARY, B.DATEOFJOIN
FROM EMP B;
```

The query combines all the records and leaves out the duplicate records.

<i>Table 54. VIRTUAL TABLE - EMPLOYEE</i>					
EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					
WAYNE	36	ADMIN	RETAIL	45000	2018-11-01

UNION ALL

If you want to keep duplicates in the result of a UNION operation, specify the UNION ALL keyword instead of just UNION.

This topic uses the same steps and example as [“UNION” on page 216](#).

The following example combines all the records from the tables [Table 34 on page 161](#) and [Table 35 on page 161](#).

```
SELECT A.EMPL_NAME, A.AGE, A.DEPT, A.INDUSTRY, A.SALARY, A.DATEOFJOIN
FROM EMPLOYEE A
UNION ALL
SELECT B.EMPL_NAME, B.AGE, B.DEPT, B.INDUSTRY, B.SALARY, B.DATEOFJOIN
FROM EMP B;
```

The query includes all the records including the duplicate records.

<i>Table 55. VIRTUAL TABLE - EMPLOYEE</i>					
EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10

Table 55. VIRTUAL TABLE - EMPLOYEE (continued)

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					
DEV	33	ADMIN	RETAIL	25000	2019-11-27
WAYNE	36	ADMIN	RETAIL	45000	2018-11-01

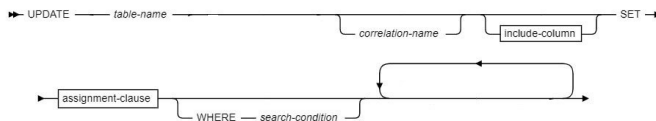
UPDATE

The UPDATE statement updates the values of specified columns in rows of a table.

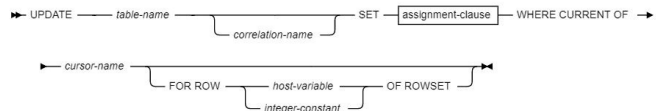
There are two forms of this statement:

- The searched UPDATE form is used to update one or more rows optionally determined by a search condition.
- The positioned UPDATE form specifies that one or more rows corresponding to the current cursor position are to be updated.

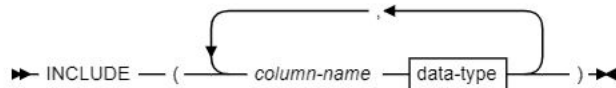
searched update:



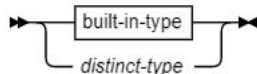
positioned update:



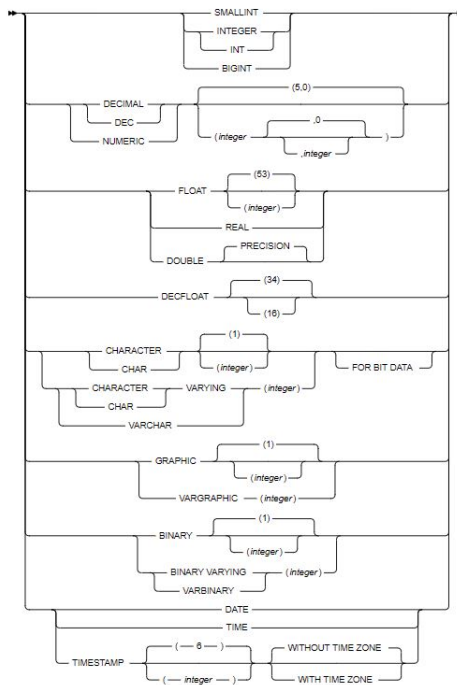
include-column:



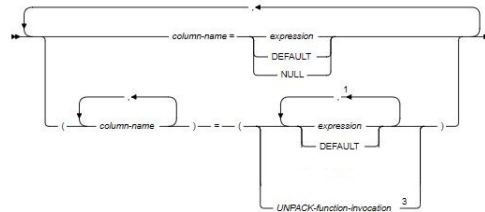
data-type:



built-in-type:



assignment clause:



Notes:

- 1 The number of expressions, DEFAULT, and NULL keywords must match the number of column-names. Expressions must not refer to UNPACK-function-invocation..
- 3 The number of items returned from UNPACK-function-invocation must match the number of column names.

Description

table-name

Identifies the object of the UPDATE statement. The name must identify a table that exists at the Data Virtualization Manager server that is identified by the implicitly or explicitly specified location name.

correlation-name

Can be used within search-condition or assignment-clause to designate the table.

include-column

Specifies a set of columns that are included, along with the columns of table-name, in the result table of the UPDATE statement when it is nested in the FROM clause of the outer fullselect that is used in a subselect, SELECT statement, or in a SELECT INTO statement. The included columns are appended to the end of the list of columns that is identified by table-name. If no value is assigned to a column that is specified by an include-column, a NULL value is returned for that column.

INCLUDE

Introduces a list of columns that are to be included in the result table of the UPDATE statement. The included columns are only available if the UPDATE statement is nested in the FROM clause of a SELECT statement or a SELECT INTO statement.

column-name

Specifies the name for a column of the result table of the UPDATE statement that is not the same name as another included column nor a column in the table that is specified in table-name.

data-type

Specifies the data type of the included column. The included columns are nullable.

built-in-type

Specifies a built-in data type.

The CCSID 1208 and CCSID 1200 clauses must not be specified for an INCLUDE column.

distinct-type

Specifies a distinct type. Any length, precision, or scale attributes for the column are those of the source type of the distinct type as specified by using the CREATE TYPE statement.

SET

Introduces the assignment of values to column names.

assignment-clause

If row-fullselect is specified, the number of columns in the result of row-fullselect must match the number of column-names that are specified. If row-fullselect is not specified, the number of expressions must match the number of column-names that are specified.

column-name

Identifies a column that is to be updated. column-name must identify a column of the specified table, and that column must be updatable if extended indicator variables are not enabled. The column must not identify a generated column where the column is derived from a scalar function, constant, or expression. column-name can also identify an INCLUDE column that must not be qualified. The same column must not be specified more than one time.

expression

Indicates the new value of the column.

DEFAULT

Specifies that the default value is used based on how the corresponding column is defined in the table.

UNPACK-function-invocation

Specifies an invocation of the UNPACK built-in function. The number of fields that are returned by the UNPACK function invocation must be the same as the number of column-names.

WHERE

Specifies the rows to be updated. You can omit the clause, give a search condition, or specify a cursor. If you omit the clause, all rows of the table are updated.

search-condition

Specifies any search condition described in [Language elements](#). Each column-name in the search condition, other than in a subquery, must identify a column of the table.

The search-condition is applied to each row of the table and the updated rows are those for which the result of the search-condition is true. If the unique key or primary key is a parent key, the constraints are effectively checked at the end of the operation.

If the search condition contains a subquery, the subquery can be thought of as being executed each time the search condition is applied to a row, and the results used in applying the search condition. In actuality, a subquery with no correlated references is executed just once, whereas it is possible that a subquery with a correlated reference must be executed once for each row.

WHERE CURRENT OF cursor-name

Identifies the cursor to be used in the update operation. cursor-name must identify a declared cursor as explained in the description of the DECLARE CURSOR statement in [DECLARE CURSOR](#). If the UPDATE statement is embedded in a program, the DECLARE CURSOR statement must include select-statement rather than statement-name.

The object of the UPDATE statement must also be identified in the FROM clause of the SELECT statement of the cursor. The columns to be updated can be identified in the FOR UPDATE clause of

that SELECT statement though they do not have to be identified. If the columns are not specified, the columns that can be updated include all the updatable columns of the table that is identified in the first FROM clause of the fullselect.

The result table of the cursor must not be read-only. For an explanation of read-only result tables, see [Read-only cursors](#). Note that the object of the UPDATE statement must not be identified as the object of the subquery in the WHERE clause of the SELECT statement of the cursor.

When the UPDATE statement is executed, the cursor must be open and positioned on a row or rowset of the result table.

- If the cursor is positioned on a single row, that row is the one updated.
- If the cursor is positioned on a rowset, all rows corresponding to the rows of the current rowset are updated.

FOR ROW n OF ROWSET

Specifies which row of the current rowset is to be updated. The corresponding row of the rowset is updated, and the cursor remains positioned on the current rowset.

host-variable or integer-constant is assigned to an integral value k. If host-variable is specified, it must be an exact numeric type with scale zero, must not include an indicator variable, and k must be in the range 1 - 32767.

The cursor must be positioned on a rowset, and the specified value must be a valid value for the set of rows most recently retrieved for the cursor. If the specified row cannot be updated, an error is returned. It is possible that the specified row is within the bounds of the rowset most recently requested, but the current rowset contains less than the number of rows that were implicitly or explicitly requested when that rowset was established.

If this clause is not specified, the cursor position determines the rows that will be affected. If the cursor is positioned on a single row, that row is the one updated. In the case where the most recent FETCH statement returned multiple rows of data (but not as a rowset), this position would be on the last row of data that was returned. If the cursor is positioned on a rowset, all rows corresponding to the current rowset are updated. The cursor position remains unchanged.

It is possible for another application process to update a row in the base table of the SELECT statement so that the specified row of the cursor no longer has a corresponding row in the base table. An attempt to update such a row results in an error.

Datetime representation when using datetime registers:

As explained under [Datetime special registers](#), when two or more datetime registers are implicitly or explicitly specified in a single SQL statement, they represent the same point in time. This is also true when multiple rows are updated.

Other SQL statements in the same unit of work:

The following statements cannot follow an UPDATE statement in the same unit of work:

- An INSERT, UPDATE, or DELETE statement that updates accelerator-only tables from a different accelerator.

Examples

Example 1

Change employee 000190's telephone number to 3565 in a sample table EMP.

```
UPDATE EMP
SET PHONENO='3565'
WHERE EMPNO='000190';
```

Example 2

Give each member of department D11 a 100-dollar raise.

```
UPDATE EMP
```

```
SET SALARY = SALARY + 100
WHERE WORKDEPT = 'D11';
```

Example 3

Employee 000250 is going on a leave of absence. Set the employee's pay values (SALARY, BONUS, and COMMISSION) to null.

```
UPDATE EMP
SET SALARY = NULL, BONUS = NULL, COMM = NULL
WHERE EMPNO='000250';
```

UPPER

The UPPER function returns a string in which all the characters have been converted to uppercase characters.

►► UPPER(*string-expression*) ◄◄

string-expression

An expression that specifies the string to be converted. The string-expression must return a value that is a built-in character or a graphic string. A character string argument must not be a CLOB, and a graphic string argument must not be a DBCLOB. If string-expression is an EBCDIC graphic string, a blank string must not be specified for locale-name-string. If string-expression is bit data, locale-name-string must not be specified.

Syntax alternatives:

UCASE is a synonym for UPPER. UPPER should be used for conformance to the SQL standard.

Examples

The following example converts the characters available in the *EMPL_NAME* from the table EMPLOYEE to uppercase characters.

```
SELECT UPPER(EMPL_NAME) FROM EMPLOYEE
```

The above example returns the following.

```
BUMRAH
CEASAR
ELICA
DEV
NEWTON
SAMUEL
ELICA
```

WHERE

The WHERE clause specifies a result table that consists of those rows of R for which the search condition is true. R is the result of the FROM clause of the subselect.

►► WHERE — *search-condition* ◄◄

The search condition must conform to the following rules:

- Each column name must unambiguously identify a column of R or be a correlated reference. A column name is a correlated reference if it identifies a column of a table, view, common-table-expression, or nested-table-expression that is identified in an outer subselect.
- An aggregate function must not be specified unless the WHERE clause is specified in a subquery of a HAVING clause and the argument of the function is a correlated reference to a group.

Any subquery in the search-condition is effectively executed for each row of R and the results are used in the application of the search-condition to the given row of R. A subquery is actually executed for each row of R only if it includes a correlated reference. In fact, a subquery with no correlated references is executed just one time, whereas a subquery with a correlated reference might have to be executed one time for each row.

The column access control does not affect the operation of the WHERE clause.

Example:

The following example returns the rows from the table `EMPLOYEE` with columns `EMPL_NAME` and `AGE` where `AGE` is greater than 50.

```
SELECT EMPL_NAME,AGE FROM EMPLOYEE
WHERE AGE >50
```

The above example returns the following:

```
BUMRAH , 53
NEWTON , 53
```

YEAR

The YEAR function returns the year part of a value that is a character or graphic string. The value must be a valid string representation of a date or timestamp.

► YEAR(expression) ◄

The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a date or timestamp duration.

If the expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument specified:

- **If the argument is a date, a timestamp, or a string representation of either**, the result is the year part of the value, which is an integer between 1 and 9999.
- **If the argument is a date duration or a timestamp duration**, the result is the year part of the value, which is an integer between -9999 and 9999. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example 1:

The following example selects the year part from the column `UPDATED_TS` from the table `EMPL_COMP`.

```
SELECT YEAR(UPDATED_TS) FROM EMPL_COMP
```

The above example returns 2019.

Accessibility features

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use a software product successfully.

The major accessibility features in this product enable users to perform the following activities:

- Use assistive technologies such as screen readers and screen magnifier software. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.
- Customize display attributes such as color, contrast, and font size.
- Operate specific or equivalent features by using only the keyboard. Refer to the following publications for information about accessing ISPF interfaces:
 - *z/OS ISPF User's Guide, Volume 1*
 - *z/OS TSO/E Primer*
 - *z/OS TSO/E User's Guide*

These guides describe how to use the ISPF interface, including the use of keyboard shortcuts or function keys (PF keys), include the default settings for the PF keys, and explain how to modify their functions.

Product legal notices

This information was developed for products and services offered in the U.S.A.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive

Armonk, NY 10504-1785
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.html>.

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java[™] and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux[®] is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions:

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and the section titled "Cookies, Web Beacons, and Other Technologies" in IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details>. Also, see the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

Trademarks

About this task

Procedure

Privacy Policy Considerations

About this task

Procedure

Terms and conditions

About this task

Procedure

Index

A

- accessibility
 - overview [225](#)
- accessing data [18](#)
- accessing schema [78](#)
- ACI server map
 - defining [19](#)
 - displaying information [27](#)
 - extracting information [25](#)
- Adabas
 - accessing Adabas data [80](#)
- ADDI [101](#)
- AVZMFPAR member [5](#)

B

- batch
 - copying maps [5](#)
 - creating maps [5](#)
 - extracting maps [5](#)
- batchmember
 - AVZMFPAR [5](#)
- bind and grant [73](#)

C

- CA IDMS data
 - accessing [100](#)
 - virtual tables [100](#)
- client authentication [134](#)
- code [115](#)
- configuring
 - access to data sources [147](#)
 - JDBC Gateway [148](#)
 - Oracle access [152](#)
- connecting
 - Data Virtualization Manager server [73](#)
- Console
 - display settings [132](#)
- cookie policy [227](#), [229](#)
- Copybook [139](#)
- copying maps [5](#)
- creating maps [5](#)

D

- data
 - accessing [1](#)
 - virtualization [1](#)
- Data Mapping Facility
 - Adabas data definitions [157](#)
 - COBOL data types [157](#)
 - data definitions [157](#)
 - displaying data maps [61](#)
 - IMS DBD [159](#)

- Data Mapping Facility (*continued*)
 - Natural data definition module (DDM) [160](#)
 - Natural data type conversion to ODBC [159](#)
 - SQL data types [157](#)
 - SQL type support [160](#)
- Data Service [130](#)
- data source connections [74](#)
- data sources [18](#)
- Db2 columns [139](#)
- DB2 subsystems
 - accessing [73](#)
- Db2 unload data
 - virtual tables [120](#)
- Db2 Virtualization (DB2V) [106](#)
- DBMS data
 - accessing [82](#)
 - virtual tables [82](#)
- DDM [160](#)
- DNS default file [132](#)
- driver class name [154](#)

H

- HTTP
 - debug [126](#)
 - messages [126](#)

I

- IBM Application Discovery and Delivery Intelligence [101](#)
- IBM MQ
 - accessing [91](#)
 - virtual tables [91](#)
- IBM Rational Asset Analyzer [103](#)
- IMS
 - accessing [84](#)
 - database description (DBD) [159](#)
 - DBD metadata [85](#)
 - PSB metadata [86](#)
 - SQL access to database [43](#)
 - virtual tables [84](#), [87](#)
- interface
 - ACI [18](#)
 - Adabas [40](#)
 - DB2 [41](#)
 - Sequential [55](#)
 - VSAM [55](#)
- Interface for ACI [18](#)
- Interface for Adabas [40](#)
- Interface for DB2 [41](#)
- Interface for VSAM and Sequential [55](#)
- internationalization [74](#)

J

- JARS [154](#)

JavaScript [115](#)
JDBC driver settings [132](#)
JDBC Gateway
 configuring [148](#)
 preferences [153](#)
 starting the administrative console [143](#)
 starting the server [142](#)
JDBC libraries [154](#)
JDBC preferences [154](#)

L

legal notices
 cookie policy [227](#), [229](#)
 notices [227](#)
 programming interface information [227](#)
 trademarks [227](#), [228](#)
locale considerations [74](#)
Log preferences [154](#)
Logstream
 virtual tables [89](#)

M

Map Migration utility [67](#)
mapping
 batch [1](#)
 IBM Data Virtualization Manager studio [1](#)
 ISPF [1](#)
maps
 copying [64](#)
 creating source library maps [65](#)
 displaying source library maps [66](#)
 refreshing [65](#)
 setting default [61](#)
 viewing individual data elements [62](#)
migrating maps [67](#)
Multiple schema support [77](#), [78](#)

N

New IMS DBD Metadata Wizard [85](#)
New IMS PSB Metadata Wizard [86](#)
New UDTF Definitions in DB2 Wizard [106](#)
New Virtual Table Wizard [87](#), [89](#)
notices [227](#)

O

Oracle
 configuring access [152](#)
Output preferences [155](#)
overview
 perspectives [70](#)
 Studio [69](#)

P

perspectives
 DV Data [70](#)
 Services [71](#)
 Web Services [71](#)
preferences

preferences (*continued*)
 Admin [131](#)
 Code Generation [131](#)
 Console [132](#)
 Data Service [130](#)
 Dictionary [132](#)
 Drivers [132](#)
 JDBC [154](#)
 Log [154](#)
 Metadata [134](#)
 Output [155](#)
 SQL [133](#)
 Web Services [71](#), [121](#), [123–125](#)
programming interface information [227](#)

R

RAA [103](#)
RESTful APIs
 Web Services [125](#)
RESTful services [120](#)

S

Scala [115](#)
screen readers and magnifiers [225](#)
searching Server Trace
 messages [128](#)
sending comments to IBM [xiii](#)
sequential data
 accessing [95](#), [101](#), [103](#)
 virtual tables [95](#), [101](#), [103](#)
server authentication [134](#)
Server Trace
 enabling [126](#)
 exporting messages [129](#)
 filtering results [127](#)
 importing messages [130](#)
 labeling [128](#)
 messages [128](#)
 starting [127](#)
 view [126](#)
 zoom [127](#)
SMF
 virtual tables [118](#)
SQL
 executing queries [114](#)
 generating queries [114](#)
 validating queries [114](#)
SQL class [115](#)
SQL data access
 virtual table [79](#)
SSL [134](#)
Studio
 overview [69](#)

T

target systems
 create [123](#)
trademarks [227](#), [228](#)
troubleshooting
 Server Trace [126](#)

U

unload data
 virtual tables [120](#)
URL template [154](#)
User-defined table functions [106](#), [108](#)

V

validating
 SQL queries [114](#)
virtual collections
 SMF [118](#)
virtual source libraries
 creating [75](#)
virtual tables
 Adabas [80](#)
 CA IDMS data [100](#)
 Db2 unload data [120](#)
 HFS data [98](#)
 IBM Application Discovery and Delivery Intelligence [101](#)
 IBM MQ [91](#)
 IBM Rational Asset Analyzer [103](#)
 IMS
 DBD [85](#)
 PSB [86](#)
 IMS data [84](#), [87](#)
 Logstream [89](#)
 sequential data [95](#), [101](#), [103](#)
 SMF [118](#)
 VSAM [93](#)
 VSAM data [101](#), [103](#)
 zFS data [98](#)
virtual views
 creating [105](#)
Virtualization [139](#)
VSAM
 accessing data [93](#)
 virtual tables [93](#)
VSAM data
 accessing [101](#), [103](#)
 virtual tables [101](#), [103](#)

W

Web Services
 creating [125](#)
 creating Web Services directories [124](#)
 directories [124](#)
 RESTful APIs [125](#)
 target systems [123](#)
 z/OS Connect Configuration Wizard
 [121](#)

Z

z/OS Connect
 Connect Configuration Wizard [121](#)
z/OS Connect Enterprise
 Edition
 Web Services [120](#)
zFS data
 accessing [98](#)

zFS data (*continued*)
 virtual tables [98](#)



SC27-9301-02

