IIAS/Db2 Warehouse

SQLチューニングガイド

2018年10月12日版
日本アイ･ビー･エム株式会社

注意事項

本資料掲載事項は、ある特定の環境・使用状況においての正確性がIBMによって確認されていますが、すべての環境において同様の結果が得られる保証はありません。これらの製品を本番環境に適用する際には、事前に検証確認を実施いただくようにお願いします。

本導入手順書は Db2 Warehouse v2.3をベースに検証を実施した内容を元に作成しています。必要に応じ、
下記最新マニュアルを確認の上、作業を実施してください。

IBM Db2 Warehouse Knowledge Center (日本語版) ※ IBM Integrated Analytics System 含む
https://www.ibm.com/support/knowledgecenter/ja/SS6NHC/com.ibm.swg.im.dashdb.kc.doc/welcome.html

IBM Db2 Warehouse Knowledge Center (英語版) ※ IBM Integrated Analytics System 含む
https://www.ibm.com/support/knowledgecenter/ja/SS6NHC/com.ibm.swg.im.dashdb.kc.doc/welcome.html

本資料に関しての問い合わせ、案件に関する相談についてはお客様/BP様 担当IBM営業もしくは下記までご連絡ください。

IBM Analytics Technical Sales 榎本康孝(Emoto Yasutaka) 連絡先: e33119@jp.ibm.com

目次

- **概要**
  - IIASの概要
  - Db2 Warehouseの概要
  - BLU MPPの特徴
    -カラム・オーガナイズ表
    -並列データベース・パーティション構成
  - 一般的なパフォーマンス・チューニング手法
    (様々なチューニング手法における当ガイドの対象範囲)

- **パフォーマンス・チューニング手法**
  - SQLチューニング対象の特定とアクセスプランの確認方法
  - アクセスパスのチューニング

IBM Integrated Analytics System(IIAS)

“データウェアハウスのその先へ、機械学習を加速する統合データ分析プラットフォーム"

圧倒的なパフォーマンスと簡易性

性能のあくなき追求と革新的な進化によって新たに生まれ変わった次世代アプライアンス
既存DWHアプライアンスとの比較で5倍の分析クエリ応答性能、4TB/時のロード処理性能※1
大量データに対する高速な分析処理に加えて、多数のユーザーからの同時分析処理要求にも対応

最先端の機械学習をこの1台で簡単に

DWHにデータサイエンスを加速する分析ソフトウェアData Science ExperienceとSparkを統合
チームコラボレーション機能、IBM Watson Machine Learningによる分析の自動化、API連携
機械学習の常識を変える圧倒的な処理スピードを提供

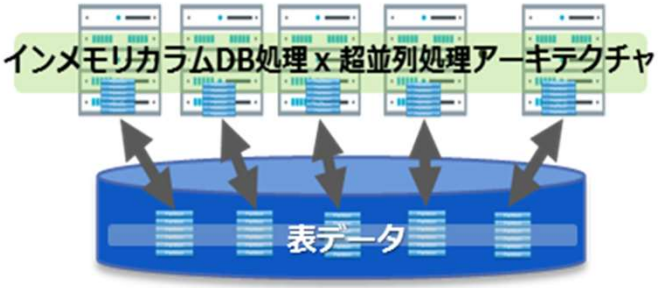
クラウド連携で広がる無限の可能性

クラウド連携機能により、外部データやオープンデータを活用しながら、素早く必要なデータマートを提供でき、ユーザーへ迅速に分析データを開放。オンプレミスはもちろん、IBM Cloud、AWS、Oracle Cloud等、各社クラウドのDBと連携可能

※1 顧客データを使用したPoCによるクエリ処理時間比較結果より

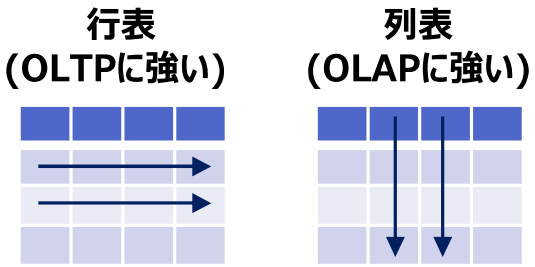
IBM Integrated Analytics System 6つの優位性ポイント+PDAからの移行容易性

1 さらなるパフォーマンス向上



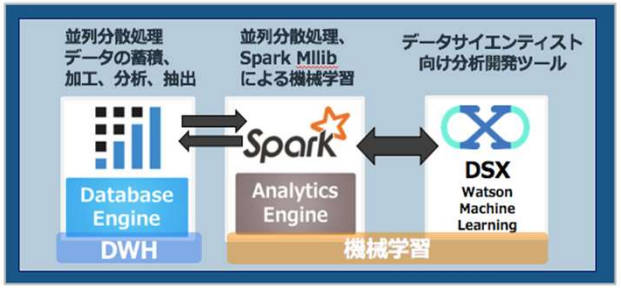
インメモリカラムナーデータベース & 超並列処理により、BI単体、多重分析処理の性能向上。標準DWHベンチマークシナリオで最新PDAと比べ平均2-5倍以上の向上を実証 ※1

2 あらゆる分析ワークロードに対応



行表・列表のハイブリッド格納が可能要件やワークロードに合あわせて適材適所で最適な構成をとることが可能。行表+索引を利用したハイトランザクションの特定レコード検索も高速対応可能

3 機械学習統合プラットフォーム



DWHとSparkエンジンを統合、PythonやRを使用した機械学習や非構造化データの分析をデータを外部に動かすことなく、単一DWHアプライアンスで処理できる。

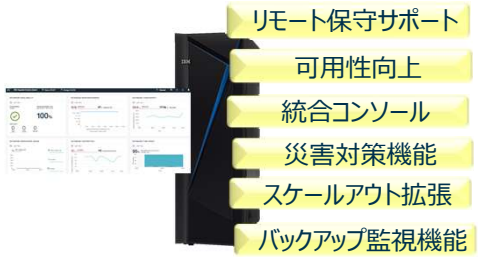
4 ハイブリッド・クラウド対応

クラウド上のDb2 Warehouse on Cloudと共通SQLエンジンとなり、アプライアンス、クラウド間でアプリケーション修正なく、クラウドの活用、移行が可能

5 多種多様なデータストアとの連携

様々なデータストア上の表をIIAS上で仮想統合可能。異種RDBに対してフェデレーション機能を利用したデータ連携ができ、GUI設定可能。

6 オペレーションビリティ向上



IBM最新ハードウェアテクノロジー統合によるシステム可用性、運用管理性向上
拡張性、災対機能、保守性の機能拡張

IBM Integrated Analytics System 物理ハードウェア構成(例:M4002-010)

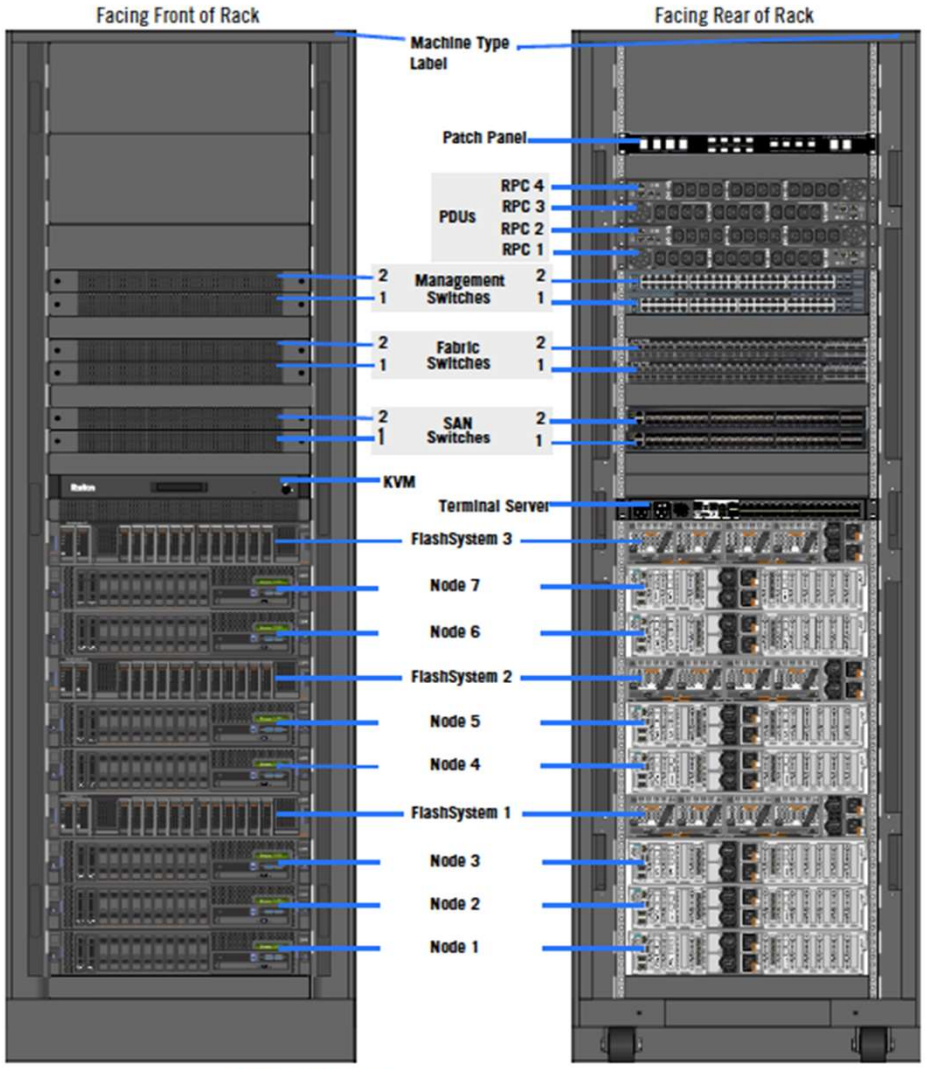


Figure 1-9: M4002-010 (Full Rack)

- コンピュートノード(003x3台,006x5台,010x7台)
  IBM Power System S822L サーバ
  24コア 512GB RAM 1.2TB SAS HDD x 2(RAID1)
  10Gbps NW x 4, 1Gbps NW x 4/台
- ストレージノード(003x1台, 006x2台, 010 x3台)
  FlashSystem 900 ストレージ
  8.5TB FlashModule x 12 枚/台
- IBM SAN64B 32/16Gb SAN スイッチ x 2台
- Mellanox SX2410 10/25Gb Fabric スイッチ x 2台
- Edge-Core AS4610 1Gb NW スイッチ x 2台

詳細はこちらを参照ください。 https://www.ibm.com/support/knowledgecenter/en/SS6NHC/com.ibm.swg.im.iias.admin.doc/doc/appl_overview_platform.html

IIAS 新モデルラインナップ情報 (2018年3月末GA 6月出荷モデル)

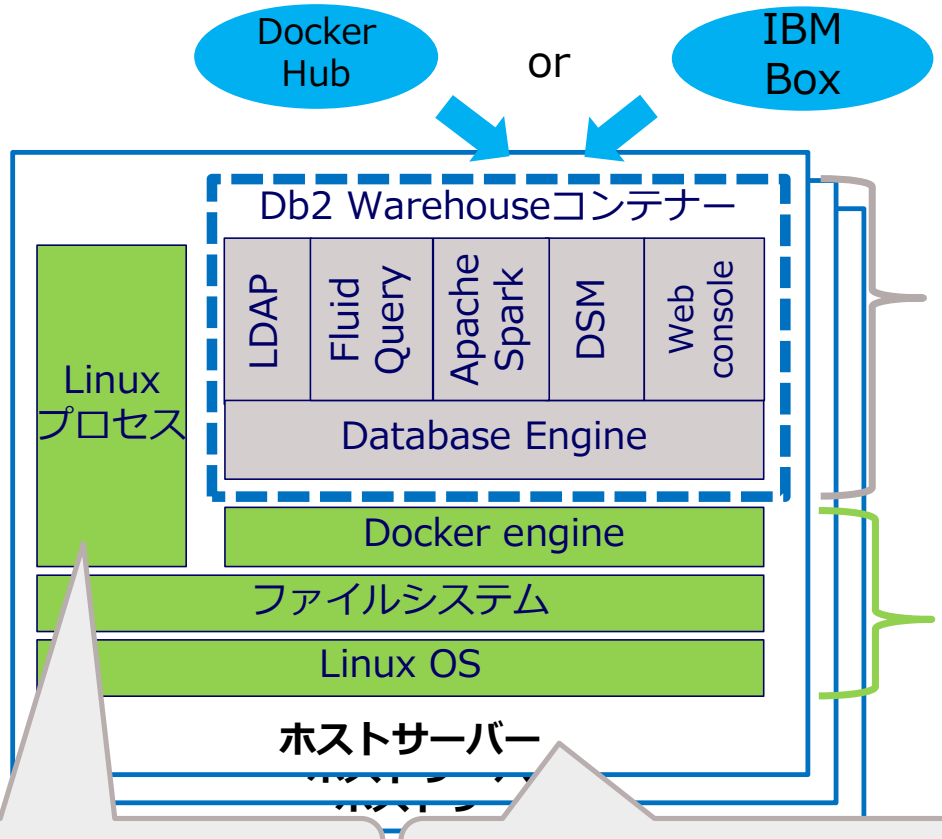
IBM Integrated Analytics System M4002ラインナップ

・マルチラックモデルリリース **(2, 4 ラックモデル)**
・全モデルフラッシュユーザーデータ領域 **約1.7倍向上**
・**Growth on Demand モデル** (M4002-003 1/3の**50%**からスタート10%ずつ拡張可能)
・その他モデルは2018年10-12月リリース予定

モデル名称	M4002-003 1/3ラック GoD 50%	M4002-003 1/3ラック	M4002-006 2/3ラック	M4002-010 1ラック	M4002-020 2ラック	M4002-040 4ラック
ノード数	3ノード	3ノード	5ノード	7ノード	14ノード	28ノード
コア数	36コア	72コア	120コア	168コア	336コア	672コア
メモリ (TB)	0.75TB	1.5TB	2.5TB	3.5TB	7TB	14TB
フラッシュ ユーザーディスク 容量	13TB (52TB)	27TB (108TB)	54TB (216TB)	81TB (324TB)	162TB (648TB)	324TB (1296TB)

※非圧縮時に格納可能なデータ量を記載しています。()内の数字は4倍圧縮時に格納可能なディスク容量の想定値となります。
上記以外にフラッシュストレージ上にはワーク領域として10TB、一時表領域としてユーザー領域と同容量が別に構成されています。

Db2 Warehouseアーキテクチャ概要



DBデータはdashDBコンテナーの外側で保管

ハードウェアやホストOSを含むDocker稼働環境を事前に準備

Db2 Warehouseの稼働環境

- パブリッククラウド
- オンプレミス (Docker Hubへの接続は必要)

Db2 Warehouseが提供する部分

- Databaseエンジン
- ユーザー管理
- データ投入
- Sparkの稼働環境
- オブジェクト管理
- 負荷モニター

事前に用意する部分

- Dockerエンジン
- ホストサーバー（H/W、OS）
- ファイルシステム　(MPPの場合は共用ファイルシステム)

ホストOS(ノード)の台数

シングルノード(SMP)またはマルチノード(MPP)

- SMPの場合：1ノード
- MPPの場合：3ノードから24ノード or 60ノード

BLU MPPの特徴 – カラム・オーガナイズ表

汎用RDBMSでの行ストア（行オーガナイズ表）

ID	商品名	価格	サイズ	発売日
1001	商品A	1000	L	2017-01-20
1002	商品B	2000	XS	2015-07-07
1003	商品C	1500	M	2016-10-31
1004	商品D	3000	S	2017-04-11

- １ブロックにすべての列のデータを格納
- クエリーの結果導出に不要な列も読み込む必要がある。

カラム・オーガナイズ表

ID
1001
1002
1003
1004

商品名
商品A
商品B
商品C
商品D

価格
1000
2000
1500
3000

サイズ
L
XS
M
S

発売日
2017-01-20
2015-07-07
2016-10-31
2017-04-11

以下の表では利用できない

パーティション表、MDC表、テンポラル表、グローバル一時表、型付き表

- 列データ毎に別ブロックに格納
- 参照処理における不要列データの読み込みが無くなり、必要なデータのみ効率良くメモリーへロード
- 同一列内には特定のデータが繰り返し現れることが多く、圧縮効率が良い

```
CREATE TABLE  MYTABLE ( ID INT, NAME CHAR(10)･･･ )
ORGANIZE BY COLU
```

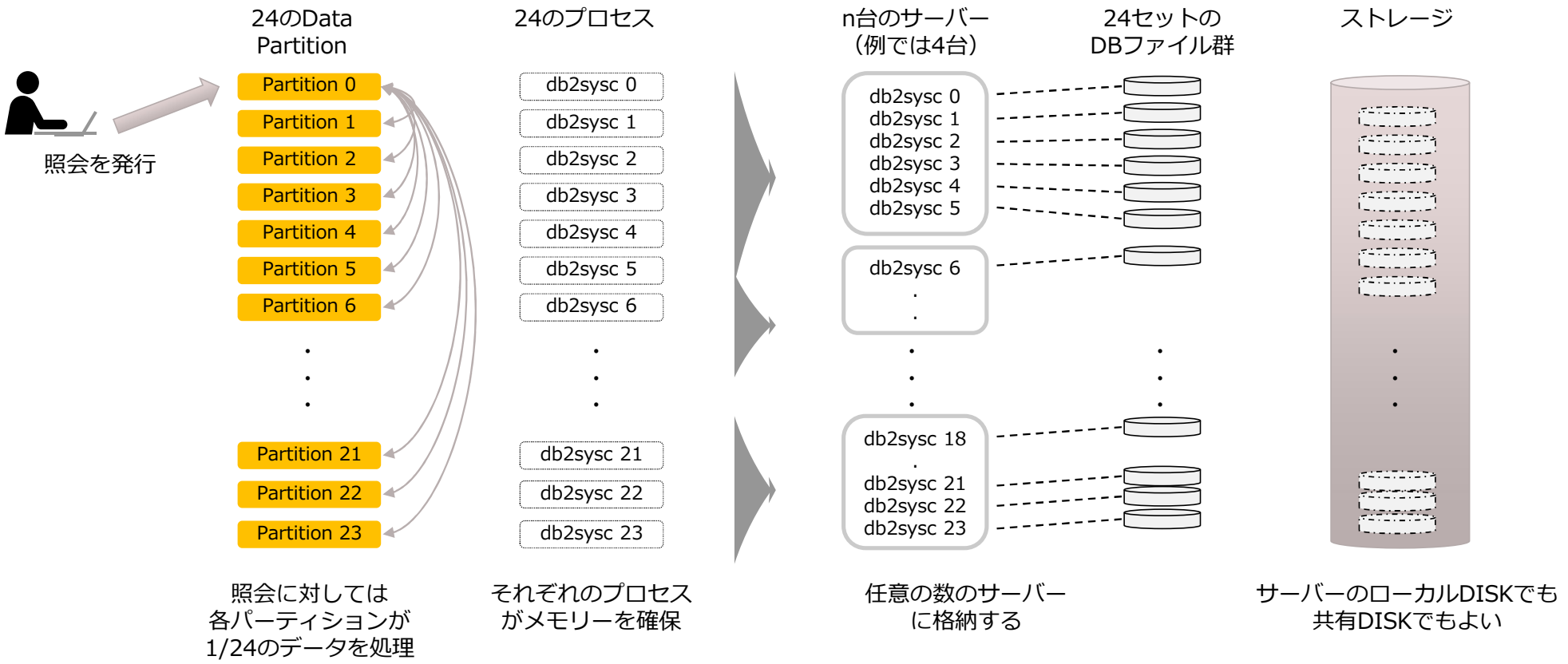
または

```
UPDATE DB CFG FOR MYDB USING
DFT_TABLE_ORG COLUMN
```

指定なしのデフォルトでもCOLUMNとなる

BLU MPPの特徴 – 並列データベース・パーティション構成

Db2 MPP環境はn個のDB partitionで構成され、それぞれのDB partitionが1/nのデータを分散して保持する



一般的なパフォーマンス・チューニング手法

- クライアント・アプリケーションの問題
  - サーバーサイドのモニターでは何も見つからないこともある
  - クライアント・サイドに問題があるケース
- 表の論理設計
  - 適切な分散キーの設定
- 表のメンテナンス
  - 統計情報の更新、圧縮辞書再作成
- SQLチューニング/アプリケーション変更
  - SQLの修正
  - 最適化クラスの変更
- オプション・チューニング
  - 構成パラメーター・チューニング
  - セカンダリー・インデックスの追加 （デザイン・アドバイザーを参考）
  - MQT（マテリアライズド・ビュー）の利用

各対応策の工数と業務への影響範囲の検討が必要

- 作業は簡単か？
- 作業にはDB2の停止が必要か？
- 影響がおよぶ期間はどれくらいか？

当ガイドの対象範囲

パフォーマンス・チューニング手法

-SQLチューニング対象の特定とアクセスプランの確認方法

SQLチューニング対象の特定とアクセスプランの確認方法

目次

- SQLチューニング対象の特定
  - dsmtopを利用したリアルタイム・クエリーモニタリング
  - Webコンソールを使用したクエリーモニタリング
- アクセスプランの取得
  - db2exfmtの使用
- アクセスプランの確認方法
  - 出力される情報の概要
  - SQL Rewriteについて(Optimized Statement)
  - ツリー(Access Plan)の見方
  - プラン詳細(Plan Details)の見方
  - 使用オブジェクト(Objects Used in Access Plan)の見方
  - 選択されるJoin方法と特徴について
  - 実行時の統計情報取得（セクションactuals）
- 参考
  - Visual Explainによる確認
  - db2explnによる確認

SQLチューニング対象の特定

dsmtopを利用したリアルタイム・クエリーモニタリング

- dsmtopとは？
  - テキスト・ベースで対話形式のGUIモニタリング
  - MON_GET 表関数の情報を元に出力
  - 常に最新情報を表示 (リアルタイム・ビュー・モニタリング)
  - CSVファイルへの出力も可能
    - Export　現在モニターしている情報を出力
    - Trace　画面更新の refresh rate (sec) 毎に追加で情報出力

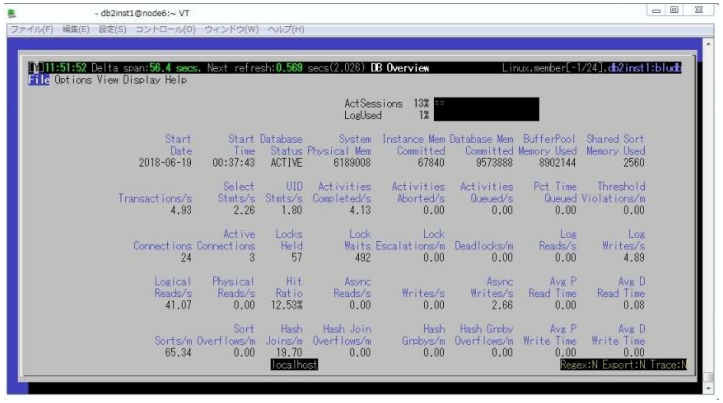
- dsmtop 起動方法
  - コマンドライン*から実行

```
[root@node6 ~]# docker exec -it Db2wh bash
[root@node6 - Db2wh /]# su - bluadmin
Last login: 金  7月 27 10:18:10 JST 2018 on pts/4
[bluadmin@node6 - Db2wh ~]$ dsmtop -d bludb
```

①Db2 Warehouseのコンテナーに接続する

②Db2ユーザーにスイッチする

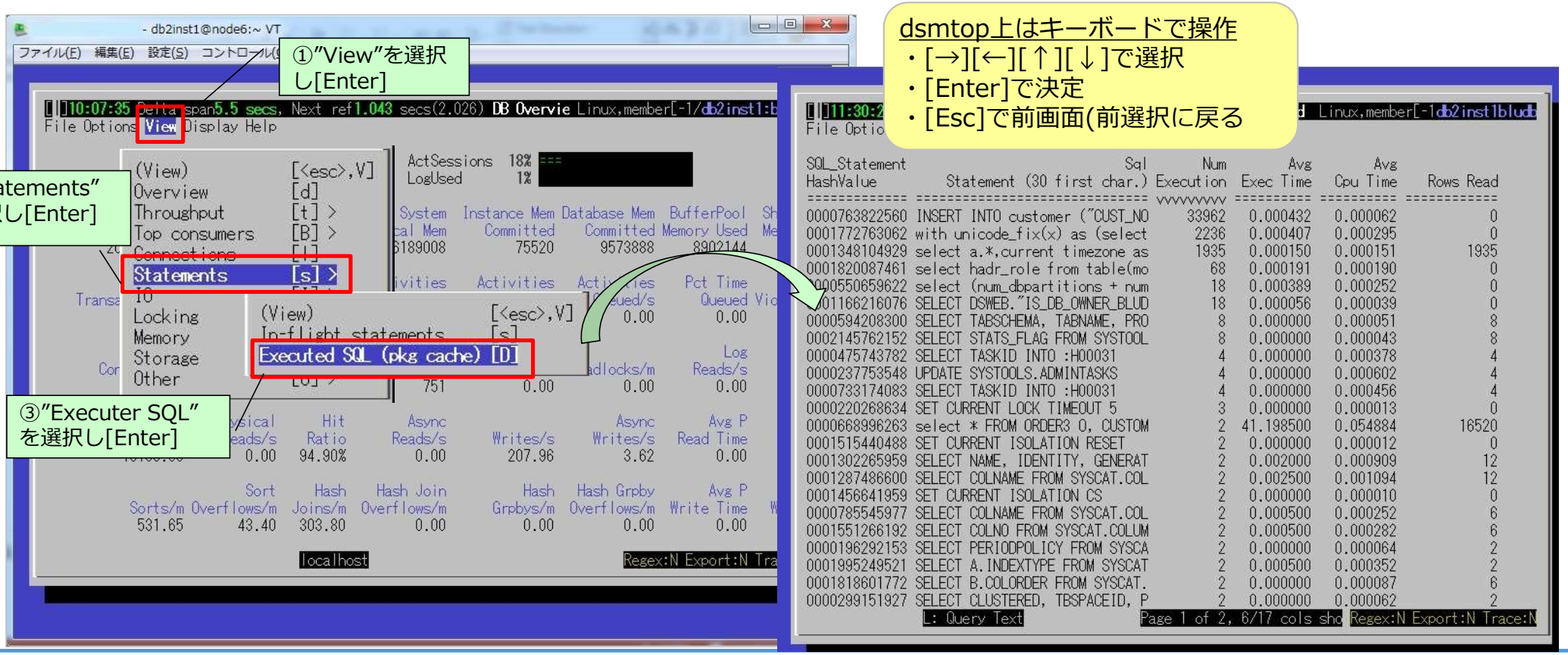
③データベース名を指定して”dsmtop”を実行すると、dsmtopの画面が立ち上がる



*ターミナル画面サイズの最小要件 “100×30”より小さい場合、起動不可

dsmtopを利用したリアルタイムクエリーモニタリング

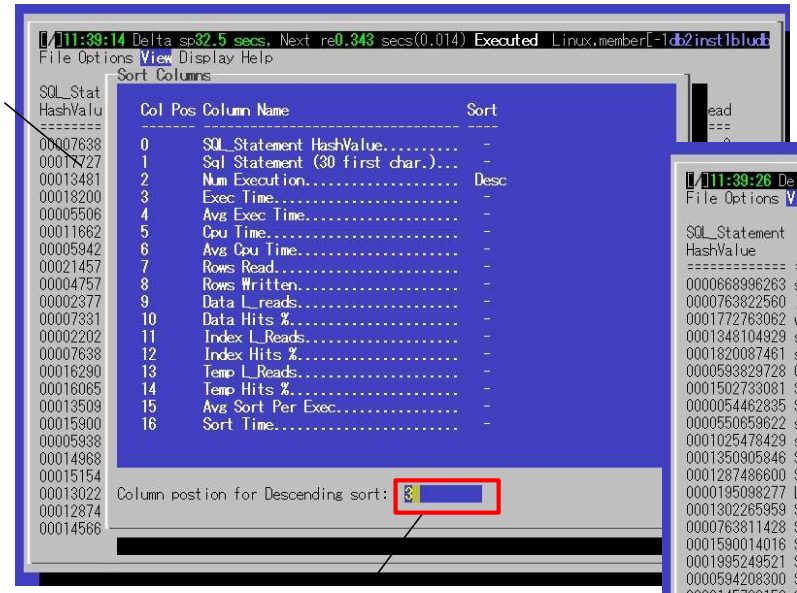
- Overview画面からSQLモニターの起動方法
  - “View” → “Statements [s]” → “Executed SQL (pkg cache) [D]”



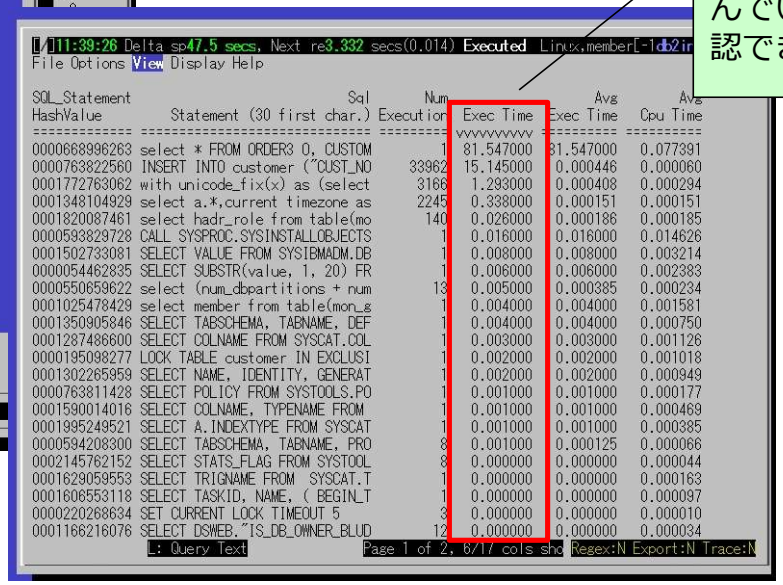
dsmtop 使用例 ：SQLステートメントのソート

- 小文字”z”を入力すると、ソート設定画面が出て、降順で並び替えたい項目の列番号を指定できる（昇順の場合は、大文字”Z”を指定する）
- 実行時間(Exec Time)でソートする場合、デフォルトの画面では3列目のため、”3”を指定する

① “z”を入力し、ソート設定画面を表示



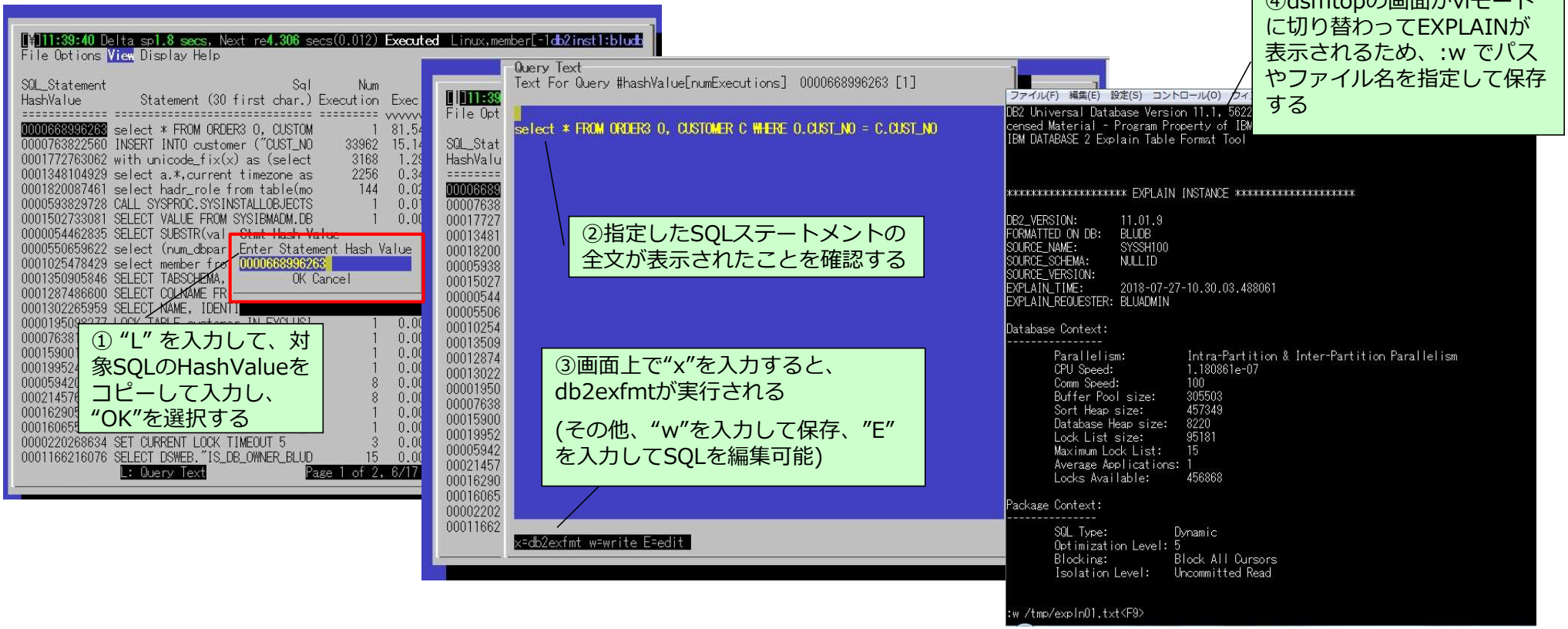
② “3”を入力して[Enter]



Exec Time順に並んでいることが確認できる

dsmtop 使用例 ： Explainの取得

- 大文字”L”を入力し、対象SQLのHashValueを指定すると、ステートメントを確認できる
- “x”を入力するとSQLのExplainがviモードで表示される



Webコンソールを使用したクエリーモニタリング

- パッケージ・キャッシュ上にあるSQLの確認

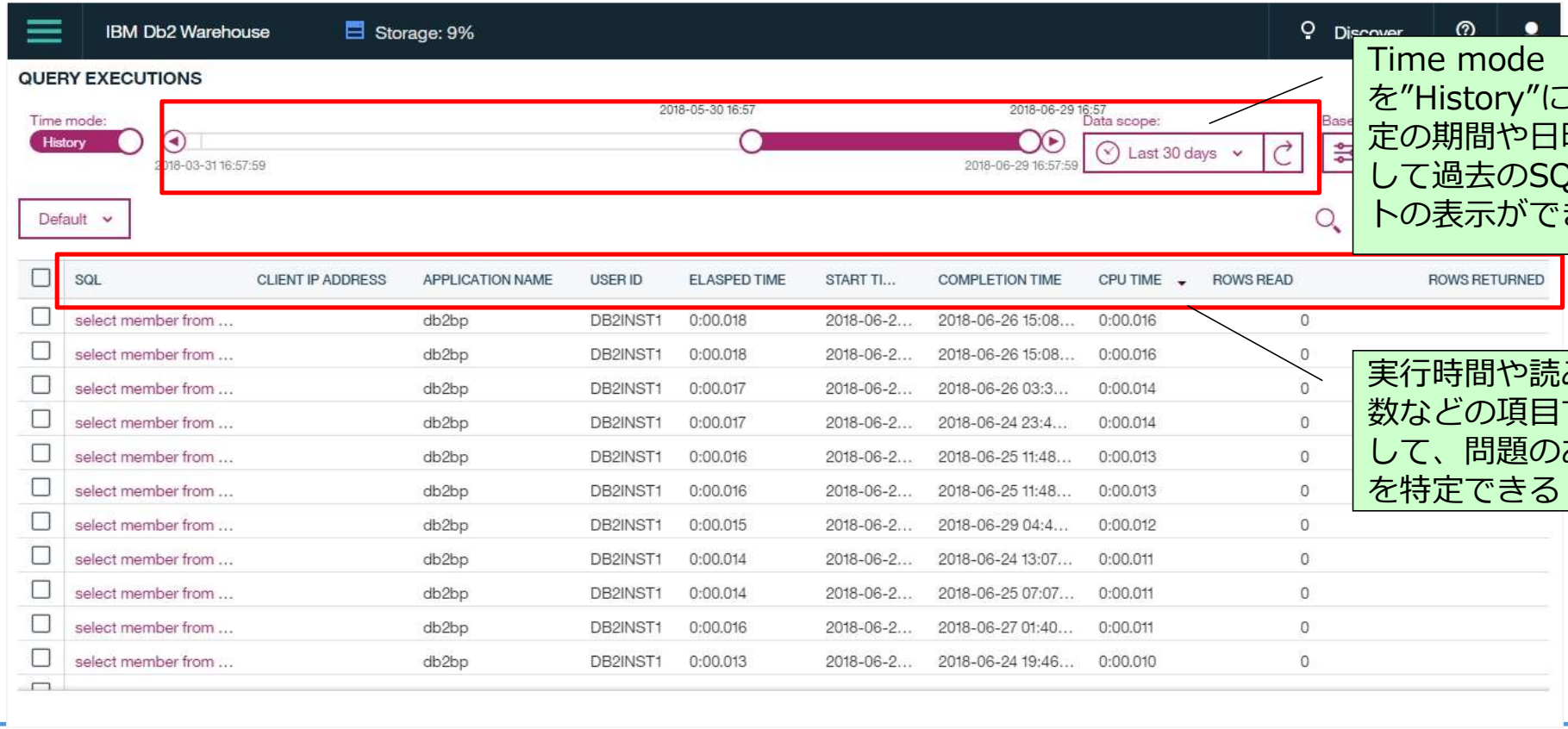
Webコンソール左端メニュー “MONITER” → “Workloads” → “Package Cache” をクリック

SQL	NUMBER OF EXECUTI...	TOTAL SO...	TOTAL ACTIVITY ...	TOTAL CPU TIME	TOTAL ROWS R...	TOTAL ROWS RETU...	TOTAL ROWS MODI...	EXTERNAL T...
select a.*,current timezone ...	27,648	442,368	0:35:06.606	0:16:55.151	27,648	27,648	0	
select member from table(...	6,060	6,060	0:07:07.741	0:05:23.488	0	48,480		
with unicode_fix(x) as (sele...	59,807	0	0:04:08.204	0:03:27.564	0	59,807		
SELECT * FROM ORDER2 ...	3	0	0:13:01.332	0:01:09.835	127,306	0		
select member from table(...	5,991	5,991	0:01:01.051	0:43.603	0	47,928		
select hadr_role from table(...	14,344	0	0:42.120	0:41.856	0	0		
SELECT * FROM ORDER3 ...	1	0	0:04:44.393	0:15.779	42,246	8,259	0	
SELECT * FROM ORDER1 ...	1	0	0:01:12.397	0:13.061	42,516	8,288	0	
SELECT (BIGINT(SUM(RE...	22	506	0:04.386	0:04.053	0	0	0	

Webコンソールを使用したクエリーモニタリング 2/2

- 実行中および実行されたSQLの一覧

Webコンソール左端メニュー “MONITER” → “Workloads” → “Query Execution” をクリック

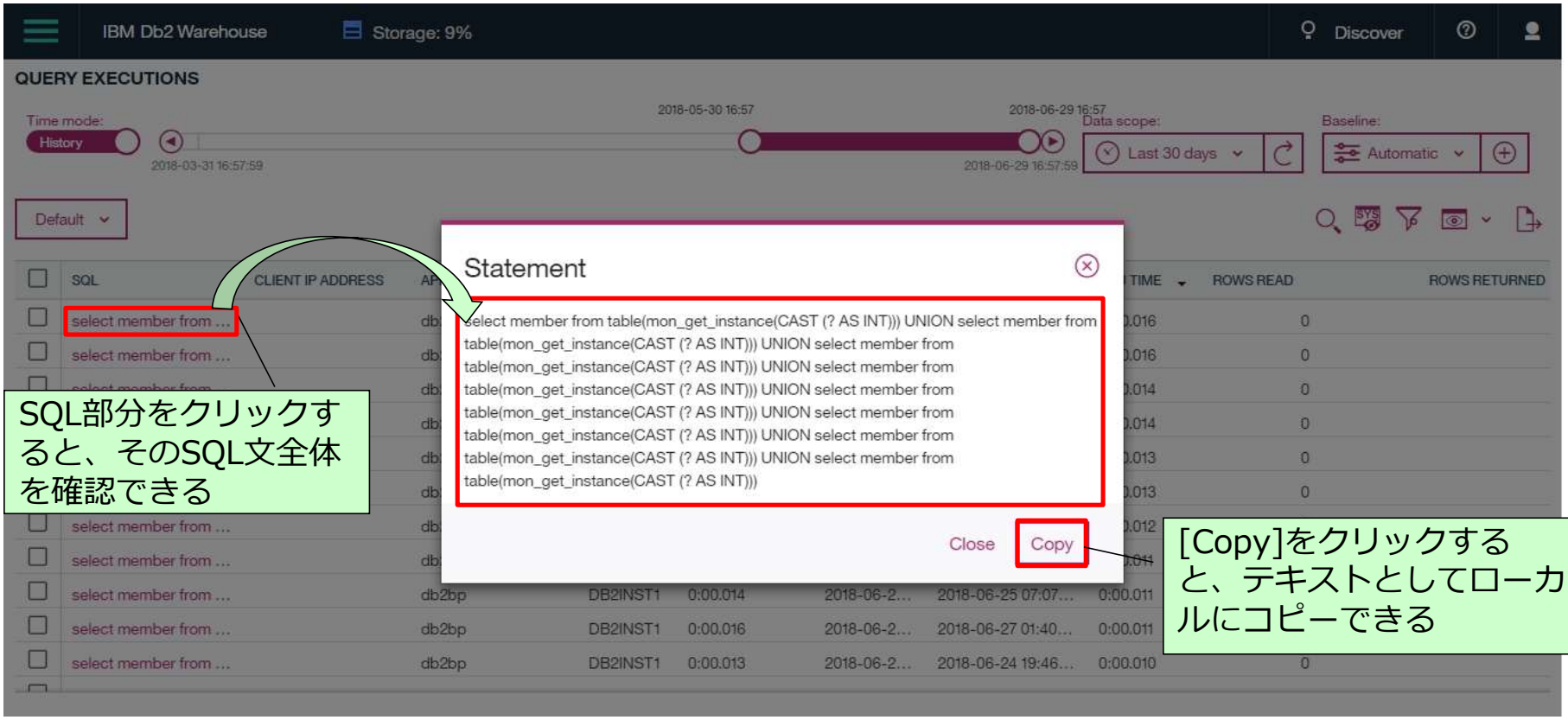


Time mode を”History”にすると特定の期間や日時を指定して過去のSQLのリストの表示ができる

実行時間や読み取り行数などの項目でソートして、問題のあるSQLを特定できる

Webコンソールを使用したクエリーモニタリング 2/2

- チューニング対象SQLの確認



アクセスプランの取得

db2exfmtの使用

- db2exfmt
  – EXPLAIN表の内容をフォーマット出力するユーティリティー

db2exfmt実行手順

1. EXPLAINモードを有効にする
   SET CURRENT EXPLAIN MODE EXPLAIN
2. アクセスプランを取得したいSQLを実行
   →SQLは実際に実行されず、アクセスプランを含むEXPLAIN情報がEXPLAIN表に格納される
3. EXPLAINモードを元に戻す
   SET CURRENT EXPLAIN MODE NO
4. db2exfmtコマンドを実行し、必要なEXPLAIN情報を取り出す

例)

```
db2 SET CURRENT EXPLAIN MODE EXPLAIN
db2 –tvf  join01.sql
db2 SET CURRENT EXPLAIN MODE NO
db2exfmt –d BLUDB -1 –o join01.exfmt
```

データベース名を指定

EXPLAIN表内の最新のEXPLAINを出力する指定

出力ファイル名を指定

アクセスパスの確認方法

Explain機能による提供される情報

① Explainの実行環境に関する情報
  -DB2バージョン、データベース名、Explain取得時刻 etc.
② 実行したSQLに関する情報
  -動的か静的か、最適化レベル、クエリー番号, 処理内容 etc.
③ ユーザーが実際に実行したSQL
④ オプティマイザーが書き換えたSQL
⑤ アクセスプラン
⑥ オペレーター処理の演算子の説明
⑦ Explain取得時の追加情報 (ワーニングなど)
⑧ プラン詳細(各オペレーター処理の詳細情報)
⑨ アクセスプランで参照しているDBのオブジェクト情報

Explain全体図 1/4

```
DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp. 1991, 2017
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool



********************* EXPLAIN INSTANCE *********************

DB2_VERSION:       11.01.9
FORMATTED ON DB:   BLUDB
SOURCE_NAME:       SQLC2O27
SOURCE_SCHEMA:     NULLID
SOURCE_VERSION:
EXPLAIN_TIME:      2018-07-11-17.12.34.350775
EXPLAIN_REQUESTER: BLUADMIN

Database Context:
----------------
        Parallelism:          Intra-Partition & Inter-Partition Parallelism
        CPU Speed:            1.180861e-07
        Comm Speed:           100
        Buffer Pool size:     275637
        Sort Heap size:       103917
        Database Heap size:   7978
        Lock List size:       4096
        Maximum Lock List:    15
        Average Applications: 1
        Locks Available:      19660
```

①Explainの実行環境に関する情報

```
Package Context:
---------------
        SQL Type:          Dynamic
        Optimization Level: 5
        Blocking:          Block All Cursors
        Isolation Level:   Cursor Stability

---------------- STATEMENT 1  SECTION 201 ----------------
        QUERYNO:       7
        QUERYTAG:      CLP
        Statement Type:   Select
        Updatable:        No
        Deletable:        No
        Query Degree:     -1

Original Statement:
------------------
SELECT
  *
FROM
  ORDER1 O,
  CUSTOMER C
WHERE
  O.CUST_NO = C.CUST_NO and
  name = 'a' and
  gender ='f'


Optimized Statement:
-------------------
SELECT
  Q2.ORDER_NO AS "ORDER_NO",
  Q2.CUST_NO AS "CUST_NO",
  Q2.ORDER_DATE AS "ORDER_DATE",
  Q1.CUST_NO AS "CUST_NO",
  Q1.NAME AS "NAME",
  Q1.GENDER AS "GENDER"
FROM
  DBUSER1.CUSTOMER AS Q1,
  DBUSER1.ORDER1 AS Q2
WHERE
  (Q1.GENDER = 'f ') AND
  (Q1.NAME = 'a') AND
  (Q2.CUST_NO = Q1.CUST_NO)
```

②実行したSQLに関する情報

③ユーザーが実際に実行したSQL

④オプティマイザーによって書き換えたSQL

Explain全体図 2/4

```
Access Plan:
-----------
        Total Cost:             282.329
        Query Degree:             2


                Rows
               RETURN
               (   1)
                Cost
                 I/O
                 |
               1644.32
               LTQ
               (   2)
               282.75
                 40
                 |
               1644.32
               CTQ
               (   3)
                282.6
                 40
                 |
               1644.32
               DTQ
               (   4)
               282.591
                 40
                 |
                               |
                             71.4921
                              HSJOIN
                              (   5)
                              281.967
                                 40
                          /---- ---+-------¥
                         331                 298.928
                       TBSCAN                TBSCAN
                       (   6)                (   7)
                       140.921               141.043
                         20                    20
                         |                     |
                         331                  1384
CO-TABLE: DBUSER1  CO-TABLE: DBUSER1
       ORDER1              CUSTOMER
         Q2                    Q1

Operator Symbols :
------------------

  Symbol      Description
  ---------   -----------------------------------------
   ATQ      : Asynchrony
   BTQ      : Broadcast
   CTQ      : Column-organized data
   DTQ      : Directed
   LTQ      : Intra-partition parallelism
   MTQ      : Merging (sorted)
   STQ      : Scatter
  RCTQ      : Column-organized data with row as the source
   XTQ      : XML aggregation
    TQ*     : Listener
```

⑤アクセスパス

⑥オペレーター処理の演算子の説明

Explain全体図 3/4

```
Extended Diagnostic Information:
--------------------------------

No extended Diagnostic Information for this statement.


Plan Details:
-------------


        1) RETURN: (Return Result)
                Cumulative Total Cost:          282.75
                Cumulative CPU Cost:            4.47821e+06
                Cumulative I/O Cost:            40
                Cumulative Re-Total Cost:       282.179
                Cumulative Re-CPU Cost:         3.20661e+06
                Cumulative Re-I/O Cost:         40
                Cumulative First Row Cost:      281.992
                Cumulative Comm Cost:           29
                Cumulative First Comm Cost:     0
                Estimated Bufferpool Buffers:   20

                Arguments:
                ---------
                BLDLEVEL: (Build level)
                        DB2 v11.1.9.0 : s1805111000
                ENVVAR  : (Environment Variable)
                        DB2_ANTIJOIN=EXTEND
                HEAPUSE : (Maximum Statement Heap Usage)
                        112 Pages
                PLANID  : (Access plan identifier)
                        cea8b4a7242f1ebb
                PREPNODE: (Prepare Node Number)
                        0
                PREPTIME: (Statement prepare time)
                                8 milliseconds
                SEMEVID : (Semantic environment identifier)
                        431f78d03d9bb07e
```

⑦Explain取得時の追加情報 (ワーニング等)

⑧プラン詳細 (各オペレーター処理の詳細情報)

```
STMTHEAP: (Statement heap size)
                16384
        STMTID  : (Normalized statement identifier)
                75334202f56e4c1a

                        ～ 中略 ～

7) TBSCAN: (Table Scan)
        Cumulative Total Cost:          141.043
        Cumulative CPU Cost:            1.20692e+06
        Cumulative I/O Cost:            20
        Cumulative Re-Total Cost:       0.135492
        Cumulative Re-CPU Cost:         1.1474e+06
        Cumulative Re-I/O Cost:         0
        Cumulative First Row Cost:      29.8836
        Cumulative Comm Cost:           0
        Cumulative First Comm Cost:     0
        Estimated Bufferpool Buffers:   20

        Arguments:
        ---------
        CUR_COMM: (Currently Committed)
                TRUE
        JN INPUT: (Join input leg)
                INNER
        LCKAVOID: (Lock Avoidance)
                TRUE
        PREFETCH: (Type of Prefetch)
                NONE
        ROWLOCK : (Row Lock intent)
                SHARE (CS/RS)
        SKIP_INS: (Skip Inserted Rows)
                TRUE
        TABLOCK : (Table Lock intent)
                INTENT SHARE
        TBISOLVL: (Table access Isolation Level)
                CURSOR STABILITY

                        ～ 中略 ～
```

Explain全体図 4/4

```
Objects Used in Access Plan:
---------------------------

        Schema: SYSIBM
        Name:   SYN180706172515224406_CUSTOMER
        Type:   Table (reference only)

        Schema: SYSIBM
        Name:   SYN180706173511401274_ORDER1
        Type:   Table (reference only)

        Schema: DBUSER1
        Name:   CUSTOMER
        Type:   Column-organized Table
                Time of creation:             2018-07-06-17.25.15.177492
                Last statistics update:       2018-07-11-17.10.25.570586
                Number of columns:            3
                Number of rows:               1384
                Width of rows:                26
                Number of buffer pool pages:  20
                Number of data partitions:    1
                Distinct row values:          No
                Tablespace name:              USERSPACE1
                Tablespace overhead:          6.725000
                Tablespace transfer rate:     0.320000
                Source for statistics:        Single Node
                Prefetch page count:          4
                Container extent page count:  4
                Table overflow record count:  0
                Table Active Blocks:          -1
                Average Row Compression Ratio:  -1
                Percentage Rows Compressed:     -1
                Average Compressed Row Size:    -1
```

```
        Schema: DBUSER1
        Name:   ORDER1
        Type:   Column-organized Table
                Time of creation:             2018-07-06-17.35.11.276566
                Last statistics update:       2018-07-11-17.00.15.060390
                Number of columns:            3
                Number of rows:               331
                Width of rows:                24
                Number of buffer pool pages:  20
                Number of data partitions:    1
                Distinct row values:          No
                Tablespace name:              USERSPACE1
                Tablespace overhead:          6.725000
                Tablespace transfer rate:     0.320000
                Source for statistics:        Single Node
                Prefetch page count:          4
                Container extent page count:  4
                Table overflow record count:  0
                Table Active Blocks:          -1
                Average Row Compression Ratio:  -1
                Percentage Rows Compressed:     -1
                Average Compressed Row Size:    -1
```

⑨アクセスプランで
参照しているDBの
オブジェクト情報
(表,索引など)

Explainを実行した環境やSQLに関する情報

- 実行時のメモリやロックの状況、実行SQLの特徴を確認する

```
DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp.
1991, 2017
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool



********************* EXPLAIN INSTANCE *********************

DB2_VERSION:       11.01.9
FORMATTED ON DB:   BLUDB
SOURCE_NAME:       SQLC2O27
SOURCE_SCHEMA:     NULLID
SOURCE_VERSION:
EXPLAIN_TIME:      2018-07-11-17.12.34.350775
EXPLAIN_REQUESTER: BLUADMIN

Database Context:
----------------
        Parallelism:          Intra-Partition & Inter-Partition Parallelism
        CPU Speed:            1.180861e-07
        Comm Speed:           100
        Buffer Pool size:     275637
        Sort Heap size:       103917
        Database Heap size:   7978
        Lock List size:       4096
        Maximum Lock List:    15
        Average Applications: 1
        Locks Available:      19660
```

CPU、

DBメモリ、
(単位:ページ(4KB))

ロックの情報

```
Package Context:
---------------
        SQL Type:          Dynamic
        Optimization Level: 5
        Blocking:          Block All Cursors
        Isolation Level:   Cursor Stability

---------------- STATEMENT 1  SECTION 201 ----------------
        QUERYNO:       7
        QUERYTAG:      CLP
        Statement Type:   Select
        Updatable:        No
        Deletable:        No
        Query Degree:     -1
```

最適化クラス

分離レベル

最適化クラスとは？

アクセスパスを決定する際に使用される要素のひとつ。設定値を高くすると、DB2がアクセスパスを決定するまでに必要とする資源（CPU）や時間がより多くかかるが、実行内容に適した高度なアクセス手法をとることができる。(DFT_QUERYOPT (DB構成パラメーター)で設定。デフォルトは5　※詳しくは当資料P.67 ご参照)

分離レベルとは？

トランザクションの中でどの程度厳密にデータの一貫性を保つかを決定するレベル。ISOが定義する分離レベルには、Uncommitted Read、Cursor Stability、Read Stability、Repeatable Read の4つがある。利用するロックのモードやロックを保持する期間を変更することで、それぞれの分離レベルを実現する。

Optimized Statement (オプティマイザーが書き換えたSQL)

- SQLはオプティマイザーによってRewriteしてから実行される
  - 異なる書き方のSQLでも
    オプティマイザーによって同じSQLに
    書き換えられて実行されることもある
  - WITH句の中が先に実行されるわけではない

```
Original Statement:
------------------
SELECT
  *
FROM
  ORDER1 O,
  CUSTOMER C
WHERE
  O.CUST_NO = C.CUST_NO and
  name = 'a' and
  gender ='f'
```

ユーザーが実行したSQL

```
Optimized Statement:
-------------------
SELECT
  Q2.ORDER_NO AS "ORDER_NO",
  Q2.CUST_NO AS "CUST_NO",
  Q2.ORDER_DATE AS "ORDER_DATE",
  Q1.CUST_NO AS "CUST_NO",
  Q1.NAME AS "NAME",
  Q1.GENDER AS "GENDER"
FROM
  DBUSER1.CUSTOMER AS Q1,
  DBUSER1.ORDER1 AS Q2
WHERE
  (Q1.GENDER = 'f ') AND
  (Q1.NAME = 'a') AND
  (Q2.CUST_NO = Q1.CUST_NO)
```

オプティマイザーが書き換えたSQL

アクセスプランの基本的な読み方

確認ポイント

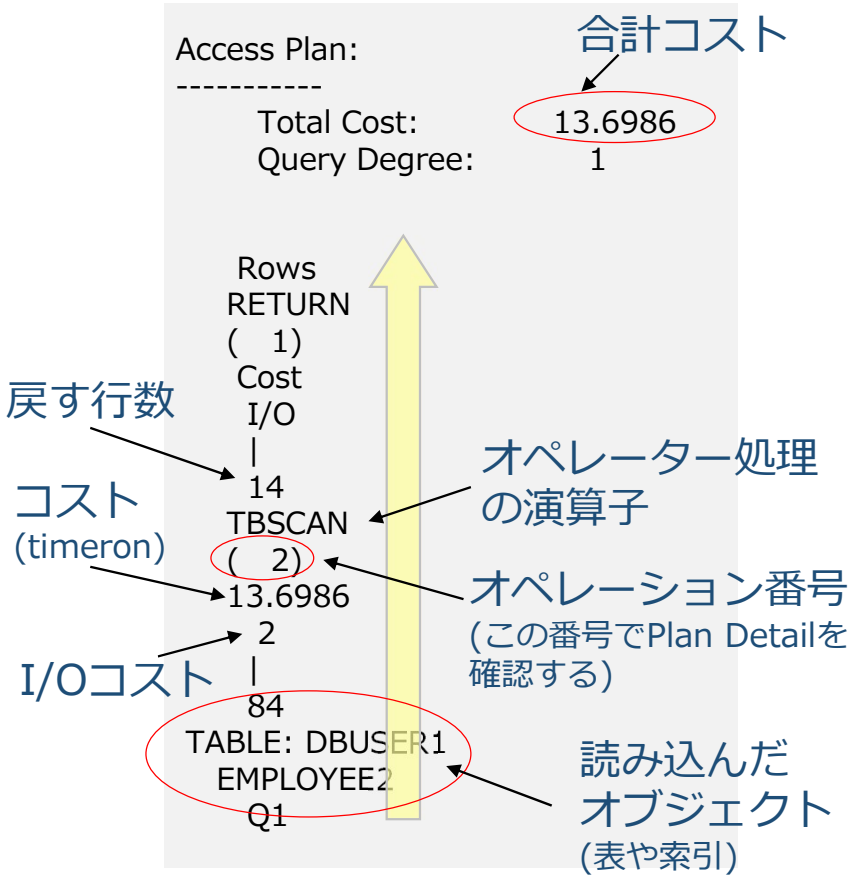
✓下から上（左下から右上）に読む

✓コストは下から上に加算(累積)されていく

✓どの処理でコストが高くなっているのか

✓合計コストに対する割合が高い処理がないか

✓後続の詳細情報を見るためにコストの高いオペレーション番号を確認

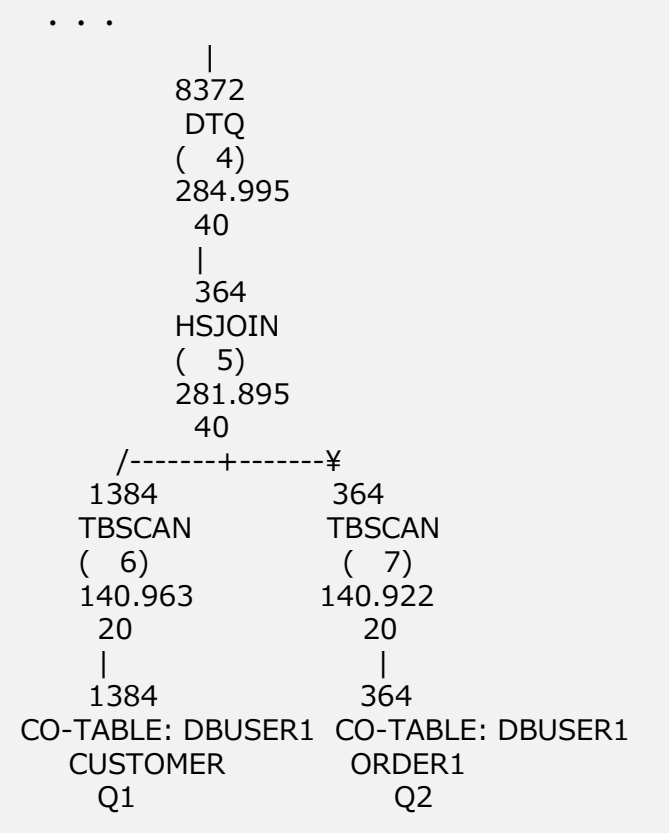


一般的なオペレーター処理の演算子

- TBSCAN：表スキャン
  - テーブルを直接読み取る
- IXSCAN：インデックス・スキャン
  - 最初にインデックスにアクセスする
- FETCH：フェッチ
  - 特定のレコードID (RID)を使って、表から列を取り出す
- RIDSCN：リッドスキャン
  - 索引から取得されるレコードID(RID)のリストを走査する
- SORT:ソート
  - 与えられたデータを昇順、降順に並べ替える
  - 重複をなくすためにも使用される

■ジョイン・メソッド (p.xx～詳細説明)

- HSJOIN：ハッシュ・ジョイン
- NLJOIN：ネステッド・ループ・ジョイン
- MSJOIN：マージ・スキャン・ジョイン



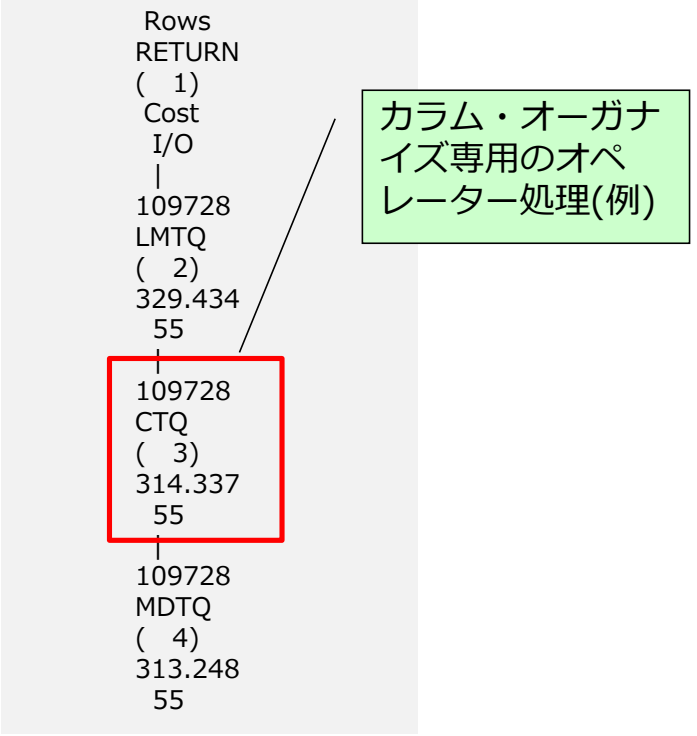
Db2 Warehouse 特有の演算子

■カラム・オーガナイズ表に関する演算子

- CTQ
  - カラム・オーガナイズデータ処理と行オーガナイズデータ処理の間の移行処理

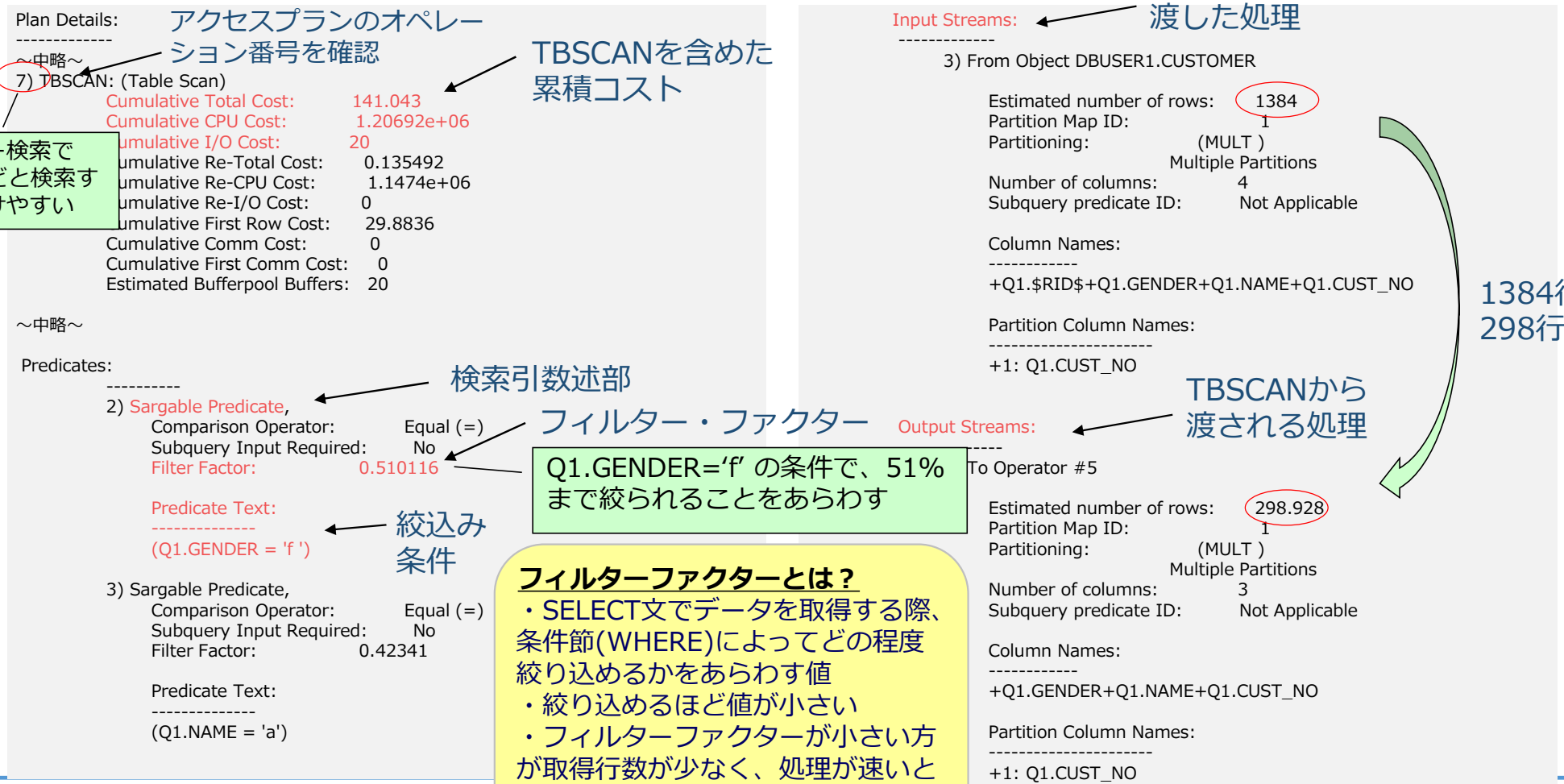
■パーティションDBに関する演算子

- DTQ（指示表キュー）
  - レコードを再ハッシュしてキューに入れる
- BTQ（ブロードキャスト表キュー）
  - レコードを再ハッシュせずに全てキューに入れる
- LTQ（イントラパーティション表キュー）
  - 表キューをパーティション内で利用する
- MTQ（マージ表キュー）
  - ソートのために表キューを利用する



プラン詳細(Plan Details)の見方 1/2

- TBSCAN（表スキャン）



プラン詳細(Plan Details)の見方 2/2

- HSJOIN（ハッシュ・ジョイン）

```
Plan Details:
-------------
～中略～
5) HSJOIN: (Hash Join)
        Cumulative Total Cost:          281.967
        Cumulative CPU Cost:            1.41842e+06
        Cumulative I/O Cost:            40
        Cumulative Re-Total Cost:       281.967
        Cumulative Re-CPU Cost:         1.41842e+06
        Cumulative Re-I/O Cost:         40
        Cumulative First Row Cost:      281.967
        Cumulative Comm Cost:           0
        Cumulative First Comm Cost:     0
        Estimated Bufferpool Buffers:   20

        Arguments:
        ---------
        EARLYOUT: (Early Out flag)
                NONE
        SEMIJOIN: (Semi-join flag)
                FALSE

        Predicates:
        ----------
        4) Predicate used in Join,
                Comparison Operator:        Equal (=)
                Subquery Input Required:    No
                Filter Factor:              3.14149e-05

                Predicate Text:
                --------------
                (Q2.CUST_NO = Q1.CUST_NO)

        Input Streams:
        -------------
                2) From Operator #6

                        Estimated number of rows:       331
                        Partition Map ID:               1
                        Partitioning:                   (MULT )
                                                        Multiple Partitions
                        Number of columns:              3
                        Subquery predicate ID:          Not Applicable

                        Column Names:
                        ------------
                        +Q2.ORDER_DATE+Q2.ORDER_NO+Q2.CUST_NO

                        Partition Column Names:
                        ----------------------
                        +1: Q2.CUST_NO

                4) From Operator #7

                        Estimated number of rows:       298.928
                        Partition Map ID:               1
                        Partitioning:                   (MULT )
                                                        Multiple Partitions
                        Number of columns:              3
                        Subquery predicate ID:          Not Applicable

                        Column Names:
                        ------------
                        +Q1.GENDER+Q1.NAME+Q1.CUST_NO

                        Partition Column Names:
                        ----------------------
                        +1: Q1.CUST_NO

        ～省略～
```

アクセスパスにあったオペレーション番号

HSJOINを含めた累積コスト

HSJOINのタイプ

フィルターファクター

結合キーが”CUTO_NO”で、この条件によって、3/100000に絞られることがわかる

結合条件

処理番号 (6)から渡された処理

処理番号 (7)から渡された処理

使用オブジェクト(Objects Used in Access Plan)の見方

- Explainが参照しているDBのオブジェクト(表や索引)の情報

```
Objects Used in Access Plan:
---------------------------

    Schema: SYSIBM
    Name:   SYN180706172515224406_CUSTOMER
    Type:   Table (reference only)

    Schema: SYSIBM
    Name:   SYN180706173511401274_ORDER1
    Type:   Table (reference only)

    Schema: DBUSER1                          <- 表名：CUSTOMER表
    Name:   CUSTOMER
    Type:   Column-organized Table           <- タイプ：カラム・オーガナイズ表
            Time of creation:            2018-07-06-17.25.15.177492   <- 表作成日時
            Last statistics update:      2018-07-11-17.10.25.570586   <- 最終Runstats実行日時
            Number of columns:             3
            Number of rows:             1384
            Width of rows:                26
            Number of buffer pool pages:    20
            Number of data partitions:      1
            Distinct row values:           No
            Tablespace name:             USERSPACE1
            Tablespace overhead:          6.725000
            Tablespace transfer rate:     0.320000
            Source for statistics:       Single Node
            Prefetch page count:           4
            Container extent page count:    4
            Table overflow record count:    0
            Table Active Blocks:           -1
            Average Row Compression Ratio:  -1
            Percentage Rows Compressed:     -1
            Average Compressed Row Size:    -1
```

表の列数、行数、バッファープールや表スペースの情報

表の圧縮に関する情報(行圧縮の場合のみ)

```
Schema: DBUSER1                          <- 表名：ORDER1表
    Name:   ORDER1
    Type:   Column-organized Table       <- タイプ：カラム・オーガナイズ表
            Time of creation:            2018-07-06-17.35.11.276566   <- 表作成日時
            Last statistics update:      2018-07-11-17.00.15.060390   <- 最終Runstats実行日時
            Number of columns:             3
            Number of rows:             331
            Width of rows:               24
            Number of buffer pool pages:    20
            Number of data partitions:      1
            Distinct row values:           No
            Tablespace name:             USERSPACE1
            Tablespace overhead:          6.725000
            Tablespace transfer rate:     0.320000
            Source for statistics:       Single Node
            Prefetch page count:           4
            Container extent page count:    4
            Table overflow record count:    0
            Table Active Blocks:           -1
            Average Row Compression Ratio:  -1
            Percentage Rows Compressed:     -1
            Average Compressed Row Size:    -1
```

表の列数、行数、バッファープールや表スペースの情報

表の圧縮に関する情報(行圧縮の場合のみ)

確認ポイント

①きちんとRunstatsが実行されているか

②Explainが参照している各オブジェクトが同様のタイミングでRunstats実行されているか

⇒ 異なるタイミングの場合、列の値の範囲などにもズレが生じ、予期せぬフィルターファクターになる可能性がある

MPP特有の結合処理

分散キーが結合キーと一致している場合

⇨ コロケーテッド結合

- DBパーティション内で結合処理が完結する
- Joinのためのレコード移動が必要なく、DBパーティション間でのレコード送受信が発生しない

分散キーとJoinキーが異なる場合

・DBパーティション内で結合処理を簡潔できないため、レコード移動が発生

⇨ 指示結合 (DTQ)

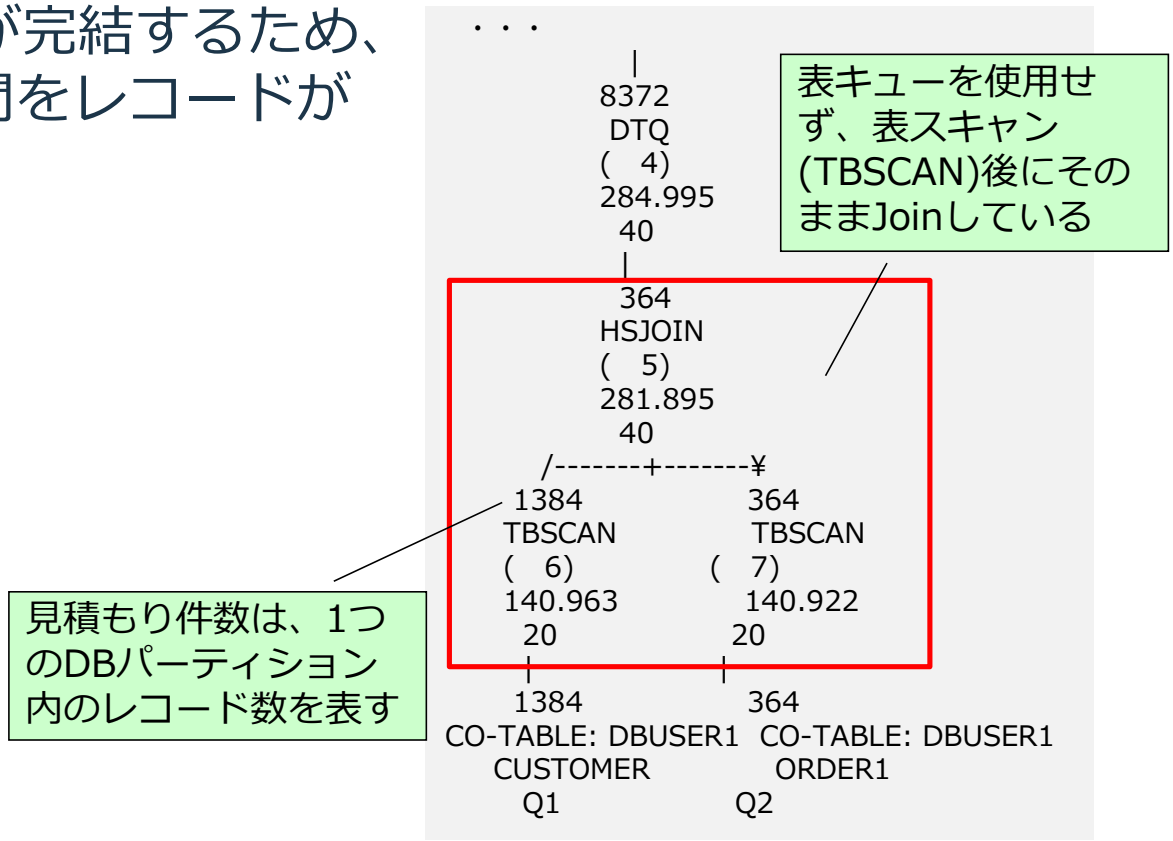
- 分散キーと結合キーが一致しない表のレコードを再ハッシュさせ、DTQを利用して対象パーティションに送信
- 対象レコードが少なければ、コストの高い処理ではない
- 使用されている条件によるレコードの絞込みの見積もりが適切に評価されていれば問題ない

⇨ ブロードキャスト結合 (BTQ)

- 分散キーと結合キーが一致しない表の全てのレコードをBTQを利用して対象パーティションに送信
- BTQにより送信される対象が小さい表であるならば、コストの高い処理ではない
- BTQで送信されるレコード数が適切に評価されていれば問題ない

結合パターン：コロケーテッド結合

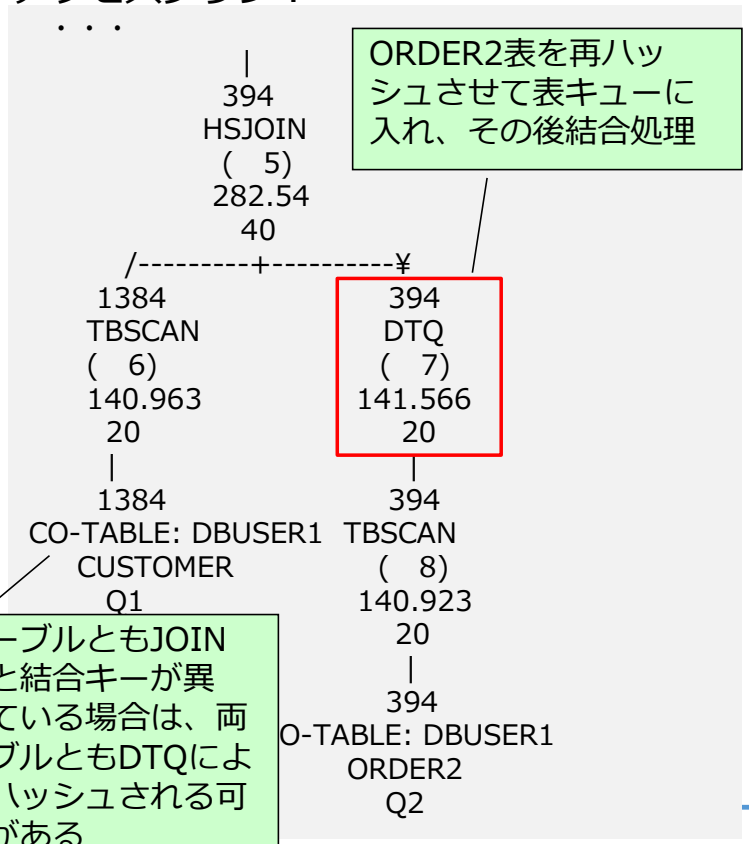
- 分散キーが結合キーと一致している最も効率的な結合処理
  - DBパーティション内で結合処理が完結するため、JoinのためにDBパーティション間をレコードが送受信されることがない。



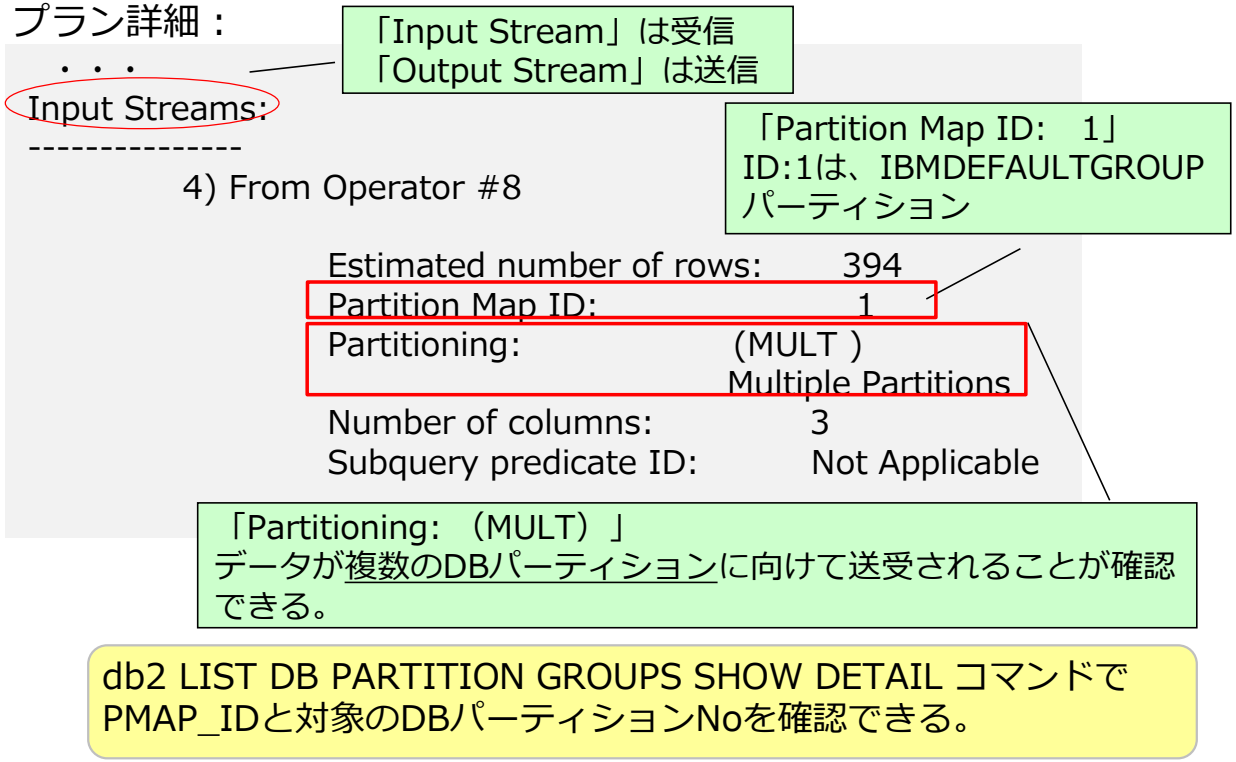
結合パターン：指示結合 (DTQ)

- 各DBパーティションから複数のDBパーティションに向けてデータが送受信される
- 対象データが多い場合には高コストの処理となる傾向がある

アクセスプラン：



プラン詳細：

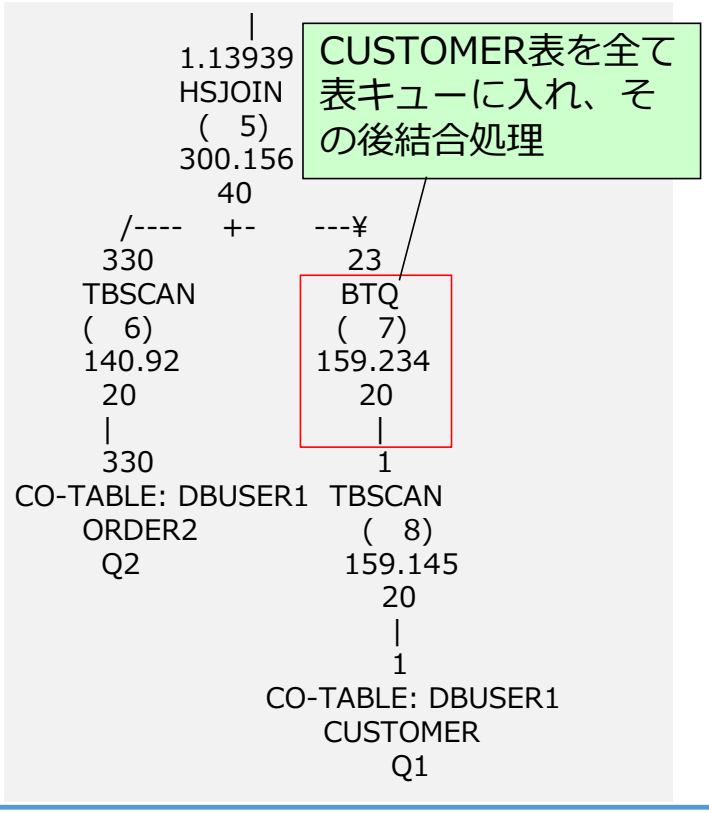


db2 LIST DB PARTITION GROUPS SHOW DETAIL コマンドで
PMAP_IDと対象のDBパーティションNoを確認できる。

結合パターン：ブロードキャスト結合 (BTQ)

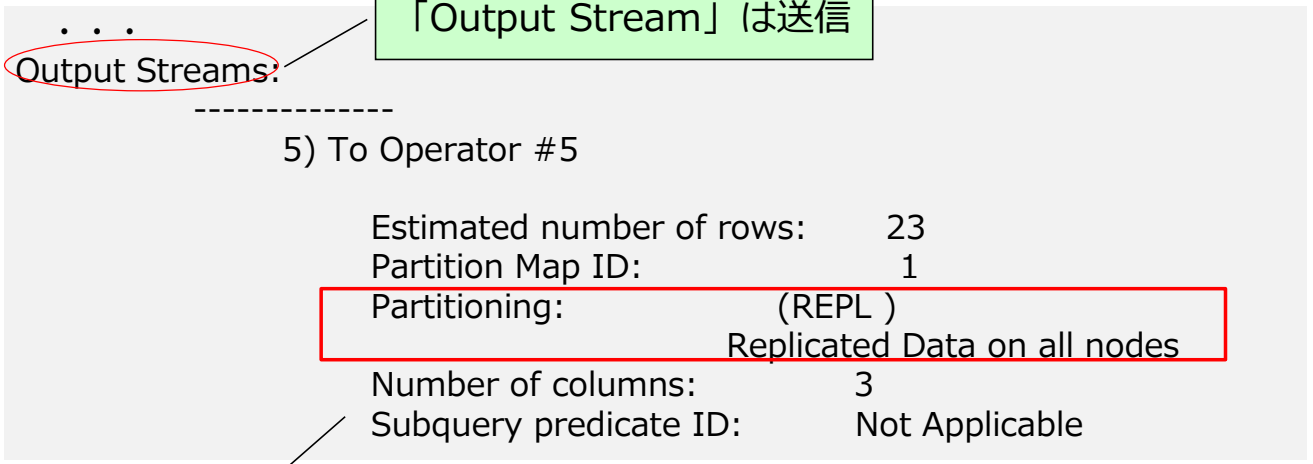
- 同じデータが各DBパーティションに向けてが送信される
- 対象データが多い場合には高コストの処理となる傾向がある

アクセスプラン：



CUSTOMER表を全て表キューに入れ、その後結合処理

プラン詳細：



「Input Stream」は受信
「Output Stream」は送信

「Partitioning: REPL」
データが複数のDBパーティションに向けてコピーされることが確認できる。

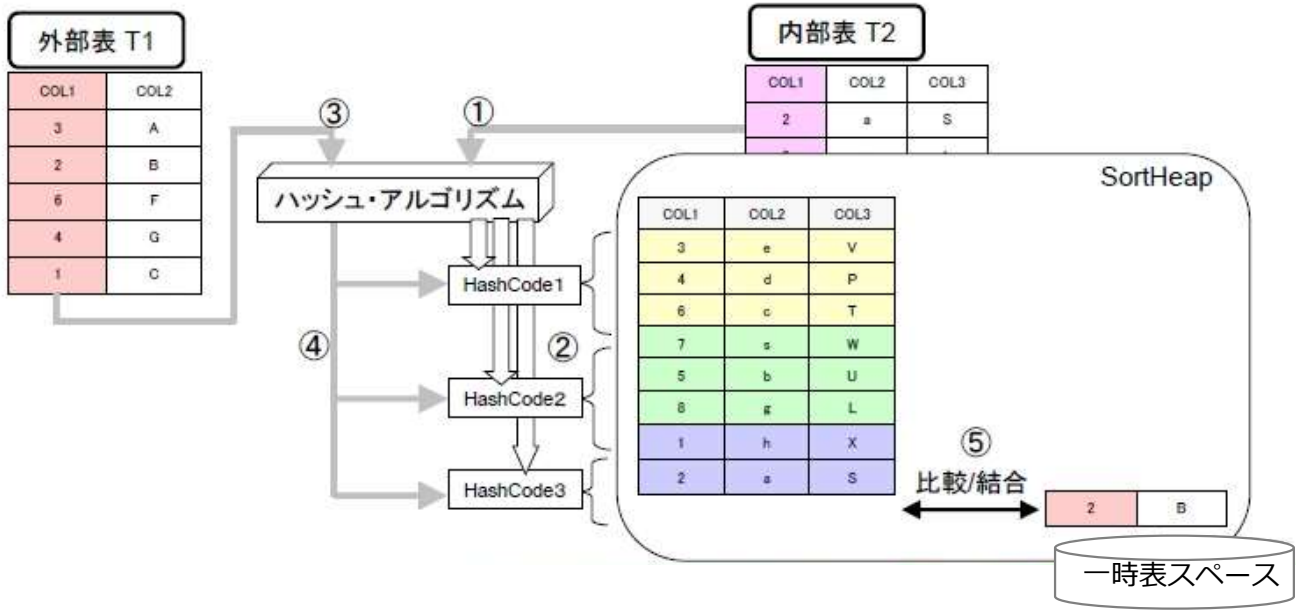
ジョインの種類とその特徴

- **ハッシュ・ジョイン（HSJOIN）**
  - MPP環境では最もよく選択されるジョインで、基本的に推奨される
  - メモリーがあれば、I/O効率が高く効率的
  - JOINする表の一方が多く、もう一方が少なめな場合に選択される
  - バッファープールが大きめの場合に選択される
- **ネステッド・ループ・ジョイン (NLJOIN)**
  - 総当りでの結合のため、大きな表には効率が悪い場合がある
  - 条件によって、絞り込まれる行数が少ない場合でかつ主キーや索引によって、マッチングする時に1件又は少ない件数しかHITしないような場合は、総当りしなくて済むので効率的
  - 見積もり誤差が発生すると性能に多大な影響を及ぼしやすいアクセスプラン
- **マージ・ジョイン（MSJOIN）**
  - ソートが発生する
  - JOINする表の該当行が両方とも多い場合に選択される
  - バッファープールが小さい場合も選択される
  - ソートが結果的にＩ／Ｏ効率が悪く、性能が悪化する場合もある

ハッシュ・ジョイン （HSJOIN）

- 結合列の値よりハッシュ値が計算され、ハッシュ値の等しい行同士が比較される
  - 単一 または 複数述部による等価結合で利用
  - ソート不要（ただし、ハッシュ値比較のためにSortHeapを利用）

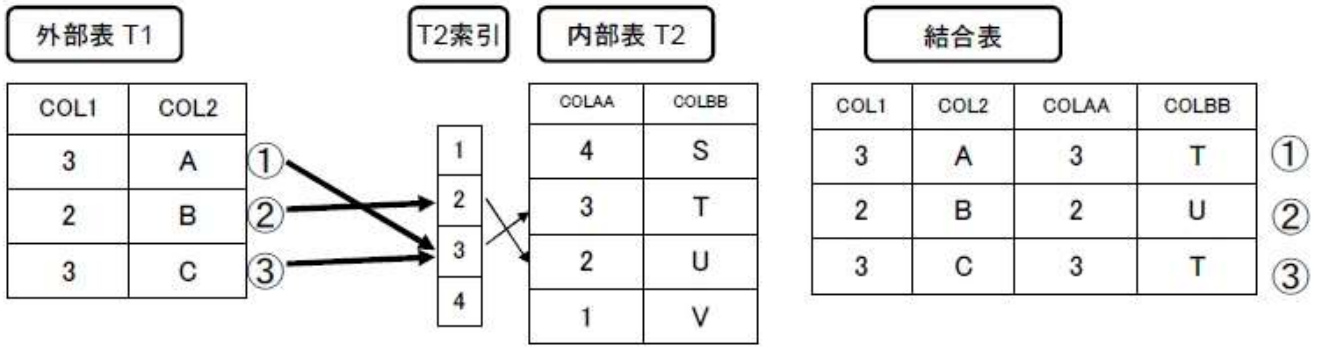
SQL文（例） SELECT ＊FROM T1, T2 WHERE T1.COL1 = T2.COL1



ネステッド・ループ・ジョイン(NLJOIN)

- 外部表から取得された1行1行について内部表をスキャンする
  - あらゆるタイプの結合で利用
    - 等価結合(where t1.c1=t2.c1)
    - 非等価結合(where t1.c1 < t2.c1)
    - 複雑な結合述部（where t1.c1+5 = t2.c2 or t1.c3=t2.c4）
  - 外部表が小さい場合に選択される

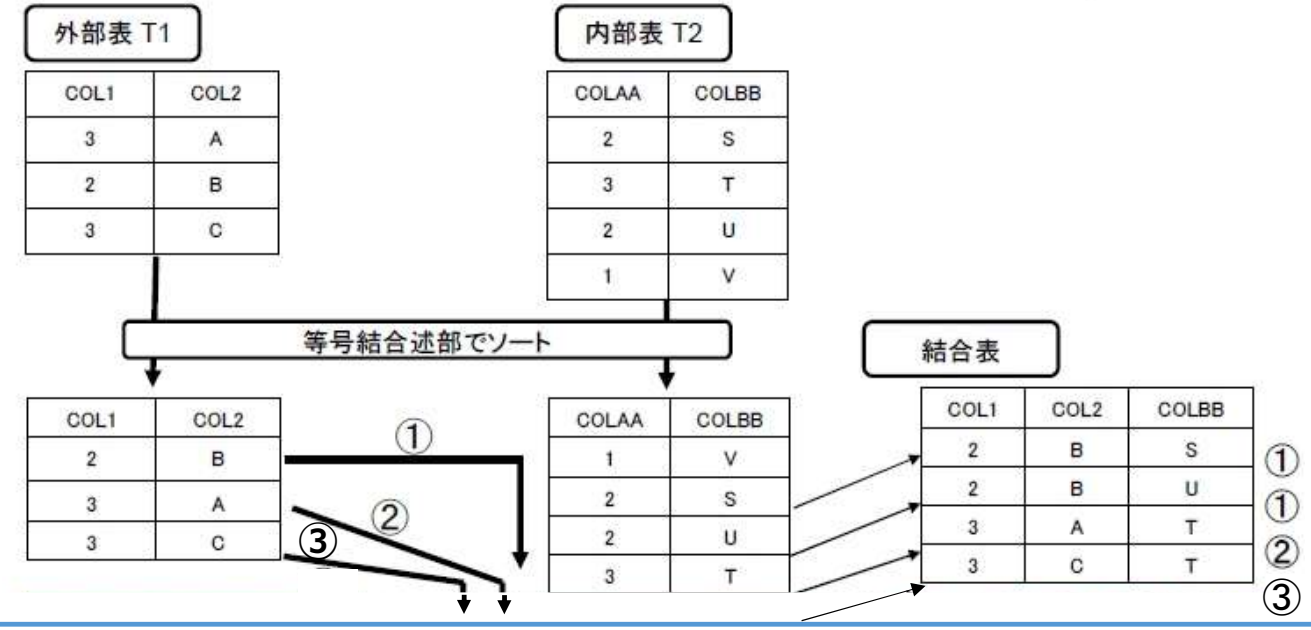
SQL文（例） SELECT COL1, COL2, COLBB FROM T1, T2
WHERE COL1 = COLAA



マージ・ジョイン（MSJOIN）

- 結合する双方の表について、結合列でソート＆マージする
  - 単一の等価結合のみ利用
  - MPP環境ではPKや副次索引が使われた場合に選択される傾向がある

SQL文（例）
SELECT COL1, COL2, CO L BB FROM T1, T2 WHERE COL1 = COLAA



実行時の統計取得（セクションactuals）

実行時の統計情報取得（セクションactuals ）

- 実行時に実際使用したアクセスプランおよび各オペレーターでの実処理件数の取得
  - 通常のExplainで表示されている見積もり結果件数と実際のクエリで返される件数に乖離があり、性能に問題がある際に取得を検討する

- 実行時統計情報の取得方法：
  ① DB構成パラメータ SECTION_ACTUAL=BASE を設定する
  ② ワークロードとアクティビティ・イベント・モニターを利用してアクティビティ情報を収集する
  ③ 収集した情報を**EXPLAIN_FROM_ACTIVITYプロシージャー**を利用してExplain表に書き出す
  ④ db2exfmtを使用してEXPLAIN表からフォーマットする

実行時統計情報 取得例 1/5

- EXPLAIN_FROM_ACTIVITYストアード・プロシージャーの使用例

1. SECTION_ACTUALS構成パラメーターを「BASE」に設定

```
$ db2 get db cfg for bludb | grep ACTUALS
 Section actuals setting              (SECTION_ACTUALS) = NONE

$ db2 update db cfg for bludb using SECTION_ACTUALS BASE
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

$ db2 get db cfg for bludb show detail| grep ACTUALS
 Section actuals setting              (SECTION_ACTUALS) = BASE                    BASE
```

2. Workloadにアクティビティ・データを収集するように設定 ※1

```
$ db2 alter workload SYSDEFAULTUSERWORKLOAD collect activity data on all with details, section
DB20000I  The SQL command completed successfully.
```

3. アクティビティを記録するためにイベント･モニターを作成し、モニター活動化

```
$ db2 create event monitor actevmon for activities write to table
DB20000I  The SQL command completed successfully.
$ db2 set event monitor actevmon state 1
SQL20156W  The event monitor was activated successfully, however some monitoring information
may be lost.  SQLSTATE=01651
```


※1 コマンドは一例です。Workloadの設定は内容に寄ってシステム負荷が高くなるため、設定内容は環境や取得対象を考慮し、検証後は元の設定値に戻してください。

実行時統計情報 取得例 2/5

4. Explainを取得したい処理(SQLやSQLを含むファイル)を実行

```
$ db2 "SELECT * FROM ORDER2 O, CUSTOMER C WHERE O.CUST_NO = C.CUST_NO"
```

5. イベント・モニター表から実行したSQLの情報を参照

```
$ db2 "select appl_id,uow_id,activity_id,substr(stmt_text,1,10)stmt_text from
activitystmt_actevmon"
APPL_ID                                   UOW_ID      ACTIVITY_ID    STMT_TEXT
----------------------------------------- ----------- -------------- ----------
*N0.db2inst1.180710050848                           3              1  set event
*N0.db2inst1.180710050848                          13              1   SELECT * F
2 record(s) selected.
```

実行したSQLの行を確認

6. EXPLAIN_FROM_ACTIVITY ストアード・プロシージャーを実行 (Explain表にExplain情報を格納)

```
$ db2 “call explain_from_activity(‘*N0.db2inst1.180710050848’,13,1,’ACTEVMON’,’SYSTOOLS’,?,?,?,?,?)”
  Value of output parameters
  --------------------------
  Parameter Name  : EXPLAIN_SCHEMA
  Parameter Value : SYSTOOLS

  Parameter Name  : EXPLAIN_REQUESTER
  Parameter Value : DBUSER1

  Parameter Name  : EXPLAIN_TIME
  Parameter Value : 2018-07-10-14.50.53.307787
   ～省略～
```

イベントモニター

エクスプレイン表スキーマ

EXPLAIN_FROM_ACTIVITY プロシージャーのパラメーター

・手順5の結果を元に、以下()内を順に指定し、青字項目は「**?**」を入力する。

(‘appl_id’, uow_id, activity_id, ‘activity_evmon_name’, ‘explain_schema’, explain_requester, explain_time, source_name, source_schema, source_version)

実行時統計情報 取得例　3/5

7. db2exfmtの実行 (6の操作でExplan表に格納されたExplain情報を出力)

```
$ db2exfmt -d bludb -1
<省略>
Access Plan:
-----------
        Total Cost:             283.606
        Query Degree:           16

            Rows
           Rows Actual
            RETURN
            (   1)
             Cost
              I/O
              |
             7567
             24777
            LTQ
            (   2)
            283.606
              NA
              |
             7567
             24777
            CTQ
            (   3)
            282.922
           <省略>
```

前項でExplain表に格納された情報を出力させるために、最新のEXPLAIN情報の出力を指定する「-1」オプションをつけて実行

実際に処理した行数がアクセスプランに表示される

オプティマイザーは統計情報をもとに、7567行をCTQの結果として戻されると見積もったが、実際(Actual)には24777行戻されていることがわかる

確認ポイント

・アクセスプランの下から順に見ていき、見積もりと実際の件数がどの処理で変わったかを確認する。

・このようなオプティマイザーによる過小見積もりが発生している場合、適切なアクセス・プランがされていない可能性があるため、統計情報の取得方法を再検討することが推奨される。

実行時統計情報 取得例　4/5

```
Plan Details:
-------------
～省略～
        2) TQ    : (Table Queue)

～省略～

                Input Streams:
                -------------
 8) From Operator #3

                        Estimated number of rows:      7567
                        Actual number of rows:        24777
                        Partition Map ID:              -100

                Output Streams:
                --------------
                        9) To Operator #1

                                Estimated number of rows:      7567
                                Actual number of rows:        24777
                                Partition Map ID:              -100

                Explain Actuals:
                ----------------
                        DB Partition number  Cardinality
                        -------------------  -------------
                                          0         24777
```

Plan DetailsにもEstimateとActualの件数が表示される

実行時統計情報 取得例　5/5

8. (任意：イベント・モニターが不要な場合) 作成したイベント・モニターを非活動化し、ターゲット表とイベント・モニターの削除する

```
$ db2 SET EVENT MONITOR ACTEVMON STATE 0

$ db2 "SELECT VARCHAR(TABSCHEMA,10) TABSCHEMA,VARCHAR(TABNAME,50)
TABNAME FROM SYSCAT.EVENTTABLES WHERE EVMONNAME = 'ACTEVMON'"

$ db2 DROP TABLE [TABSCHEMA].[TABNAME]
  e.g.  db2 drop table dbuser1.ACTIVITY_ACTEVMON
          DB20000I  The SQL command completed successful

$ db2 DROP EVENT MONITOR ACTEVMON
DB20000I  The SQL command completed successfully.
```

イベント・モニターの非活動化

イベント・モニター表を照会し、対象のアクティビティー情報を確認

確認したイベント・モニター表の削除

作成したイベント・モニターの削除

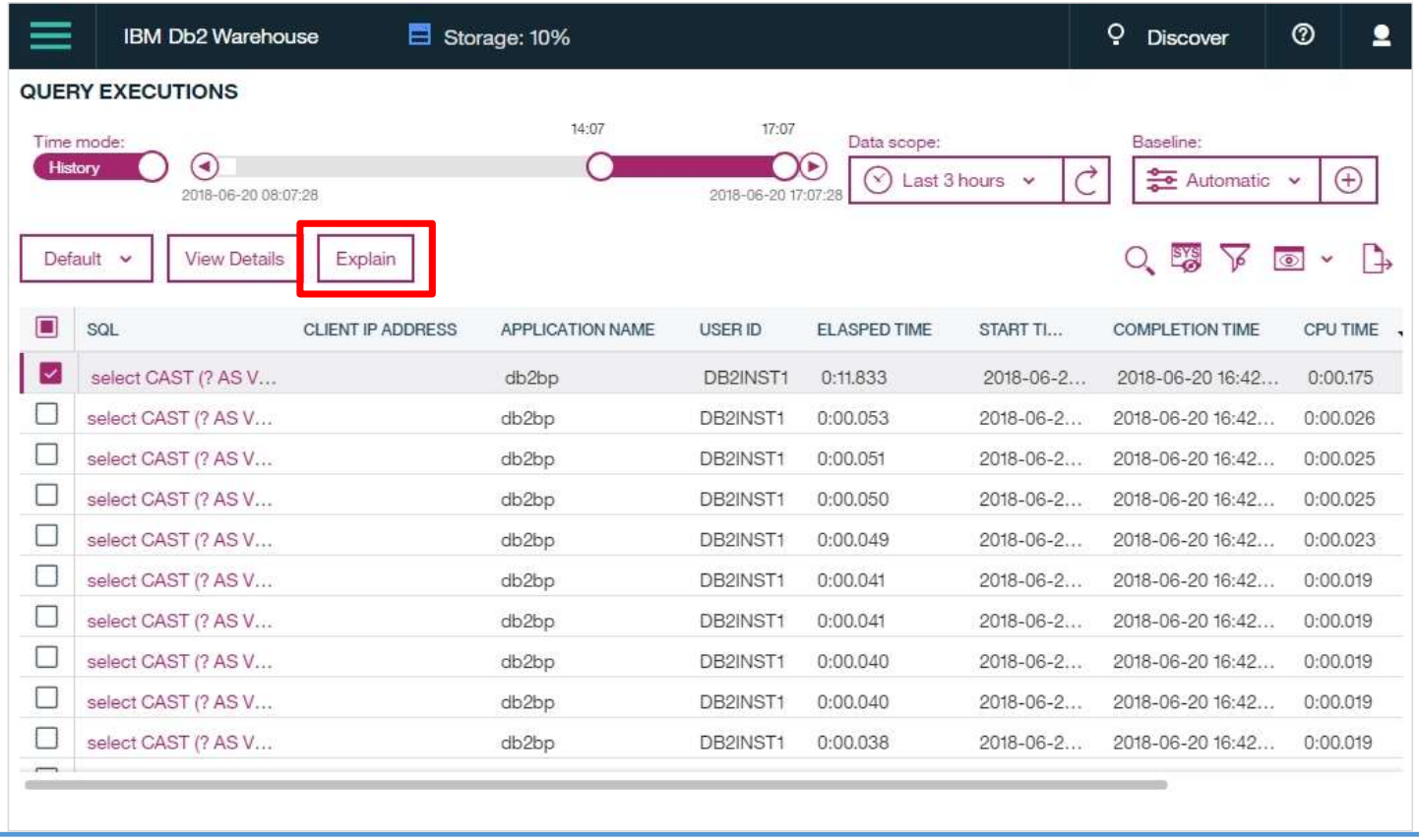
9. (任意) 手順2で設定したWorkloadの設定を元に戻す

※ Db2 Warehouse Knowledge Center のガイドもあわせてご参照ください
ステートメントがデータベース・オブジェクトに及ぼす影響の調査
https://www.ibm.com/support/knowledgecenter/ja/SS6NHC/com.ibm.swg.im.dashdb.admin.mon.doc/doc/t0058655.html

参考

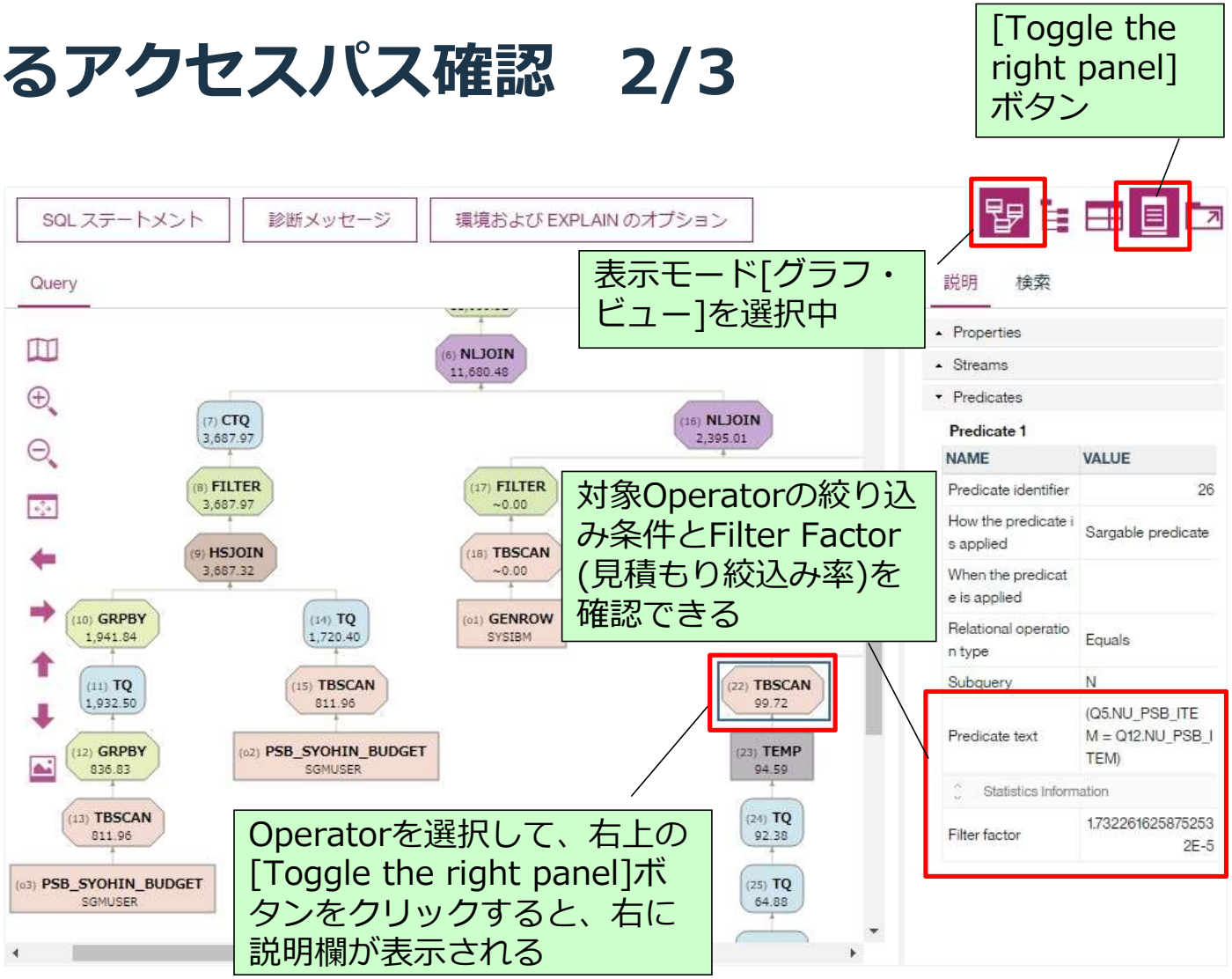
Visual Explain によるアクセスパス確認　1/3

- Webコンソールからアクセスパスを確認可能
  - 確認すべきSQLを特定し、そのSQLにチェックを入れ、「Explain」をクリックすることでVisual Explainを表示



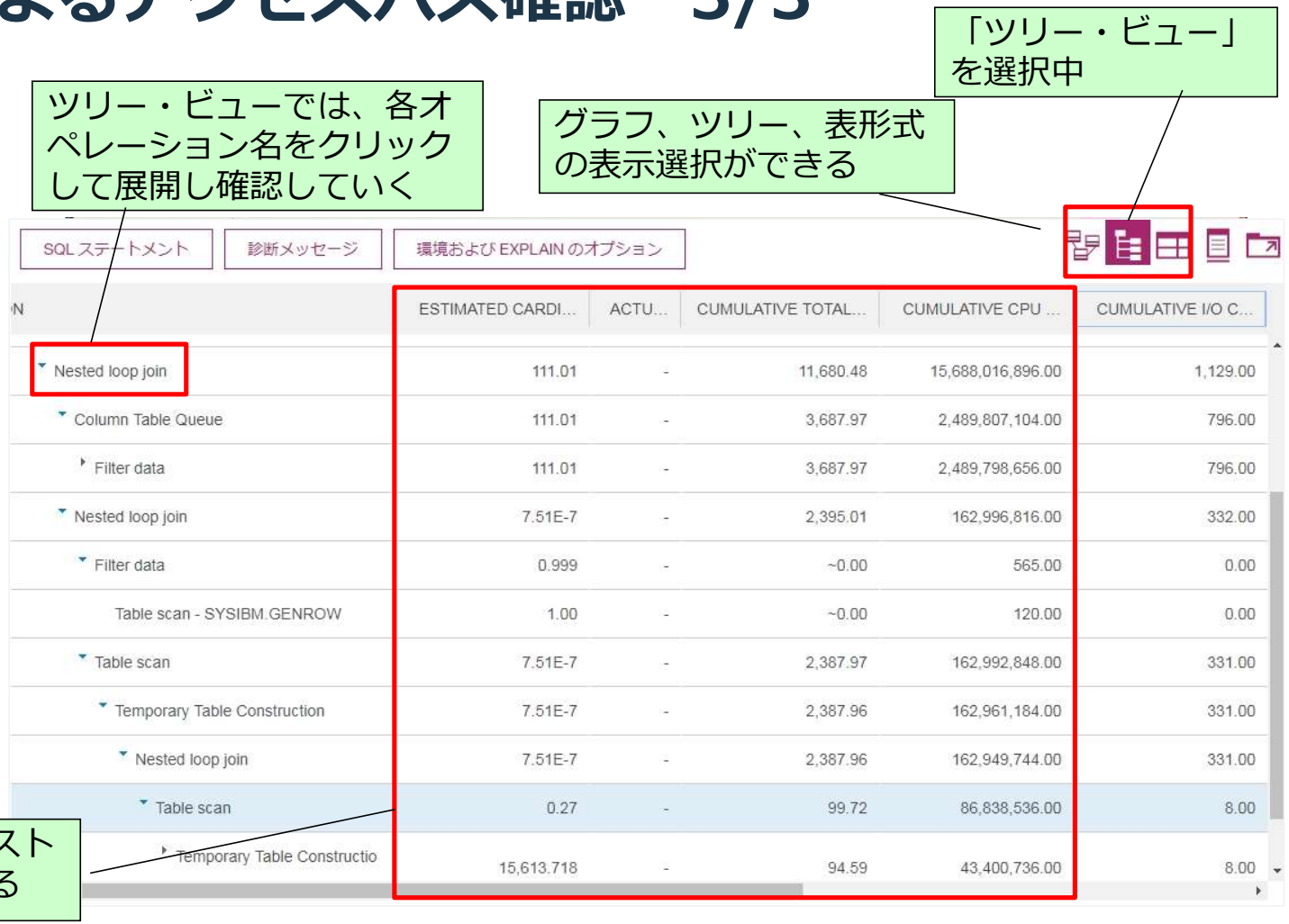
Visual Explain によるアクセスパス確認　2/3

- Visual Explainとして、グラフ表示され、確認したいOperator(処理内容)を選択すると、右側に詳細な情報が表示される
- Joinの順序、WHERE条件の適用箇所、見積もり行数、見積もりコストが想定どおりか確認できる



Visual Explain によるアクセスパス確認　3/3

- 表示モードを、「グラフ・ビュー」「ツリー・ビュー」「表形式ビュー」から選択できる
- グラフ・ビューでは全体の流れが確認でき、ツリーや表形式では、各オペレータでの見積もり行数やCPUコストの確認がしやすい

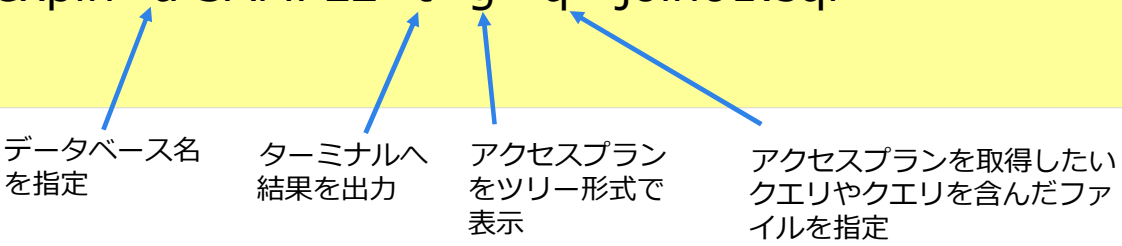


db2explnによるアクセスプランの取得

- db2explnコマンド
  - 事前準備が不要で、コマンド実行1回ですぐにアクセスプランを取得できる
  - db2exfmtコマンドと比較して取得できる情報が少ないため、確認しやすい

例)

```
db2expln -d SAMPLE -t -g –q   join01.sql
```



パフォーマンス・チューニング手法

-アクセスパスのチューニング

目次

- アクセスパスのチューニング
  - 統計情報の収集
    - 列グループ統計、統計ビュー
  - 最適化クラスの調整
  - SQLの変更／分割
  - 定義済み／作成済みユーザー一時表の利用
  - インフォメーショナル参照制約
  - 組み込み最適化ガイドラインの検討
  - WLMによるコスト監視
- まとめ

統計情報の収集

- 統計情報の収集とは
  - オプティマイザーがアクセスパスを選択する際に、索引定義、索引による絞り込み、クラスター状態、表の行数、列値の種類数、列値の分布　などコスト計算の元となる情報を、実際の表、索引をスキャンすることで収集し、カタログ表に格納する処理

```
RUNSTATS ON TABLE test1 WITH DISTRIBUTION AND DETAILED INDEXES ALL;
```

  - WITH DISTRIBUTION
    - データの分散状況（頻出値、分布）が収集される
  - コマンドを実行したDBパーティションのみで収集される
    - レコードが均等に分散している前提で、統計情報は掛け算される
    - 実行DBパーティションに表が跨っていなければ先頭DBパーティションから収集

最適なアクセスパスのためには、正確な統計情報が必要

統計情報の収集

- 統計情報の収集のタイミング
  - 表のレコードが１０～２０%変動した場合、統計情報の収集が推奨される
  - 同じタイミングでRunstatsを実行する
  - レコードが0件の場合のRunstatsは避ける
  - レコード数が想定された件数のタイミングで実行する
  - 更新負荷が低いタイミングで実行する

特にJoinする表の場合、タイミングが大きく異なると絞り込みを間違って見積もることがある

Work表のようにレコードがクリアされる場合には、実行タイミングをバッチジョブと連動させる

■Runstats運用例

初回Runstats：移行作業内で実行
基盤定期実行：日次や週次の業務バッチ時間帯以外に実行
バッチジョブ連動：ワーク表などは業務と連動

統計情報の収集

- 統計情報の収集のチューニング
  - 大規模表では、Runstats自体の実行時間、負荷を考慮する必要がある
    - サンプリングオプションの検討

```
RUNSTATS ON TABLE test1 WITH DISTRIBUTION TABLESAMPLE SYSTEM (0.1) AND
DETAILED INDEXES ALL;
```

  - TABLESAMPLE SYSTEM (*0.1*)
    - ページレベルで、この例では0.1%をサンプリングして統計情報を取集
    - 例えば、表サイズが100MB程度になることを基準としてサンプリング率を計算
    - SYSTEMではなく、BERNOULLIと指定することで、レコードレベルでのサンプリングも可能

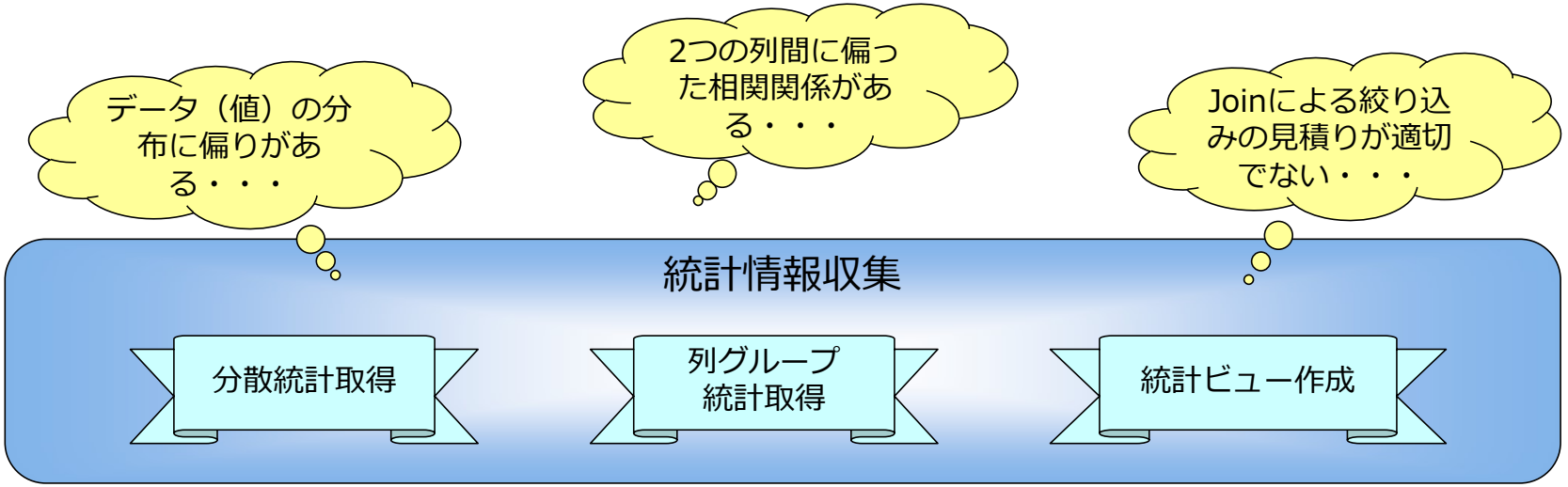
■Runstats運用例

Runstats対象データが約100MBになるように、表毎にサンプリング率を計算して実行する

統計情報の精度に問題があり、アクセスパスが不適切になった場合には、BERNOULLI指定への変更やサンプリング率を調整

統計情報の収集

- Runstatsの収集オプション
  - 最適なアクセスパスを選択させるための収集オプション
  - 不適切なアクセスパスを選択している場合に、追加の収集オプションを検討



統計情報の収集

表に複数の条件がある場合、列グループ統計を収集することで、表の絞込み件数を、より正確に見積もれる。結果的に**適切な結合順序**が期待できる。

- 列グループ統計の収集
  - 複数列間の相関関係の統計情報を収集できる
    - 複数列の条件指定での絞り込みを、正確に見積もることができる

（例）商品マスター表の商品名列と色列の値に相関関係がある
```
select * from 商品マスター where 商品名='A' and 色='白'
```

  - 相関関係とは、列の値の組み合わせに規則性があること を指す

（例）商品種類：100種類、色：10種類
ただし、8割を占める商品A　の色は、白 or 黒のみの特性

商品= A は、全体の80%
色 = 白 は、1/10の絞込み
8%

列グループあり

商品= A ⇒ 全体の80%
色 = 白 ⇒ そのうちの半分
40%

  - RUNSTATSの ON COLUMNSオプションで列グループを指定

```
RUNSTATS ON TABLE test1 ON COLUMNS ((商品,色)) AND DETAILED INDEXES ALL;
```

索引がなくても列間の相関関係を収集できる

統計情報の収集

- 統計ビューの収集
  - ビューに対して統計情報収集（RUNSTATS）が可能
  - 表間の相関関係の統計情報を収集できる
    - 表の結合による絞り込みを、正確に見積もることができる
  - 直接ビューに対する照会でなくても有効
    - アプリケーションからビューの存在を意識する必要はない
  - 統計ビューもTABLESAMPLEによるサンプリング可能
  - Viewの属性を変更してからViewへのRunstats実行

複数表の複雑な結合のSQLがある場合、表間の相関関係を統計情報として収集しておくことで、結合による絞込み件数を、より正確に見積もれる。結果的に**適切な結合順序**が期待できる。

結合した結果をRunstatsするため、表の統計情報取集よりコスト増の傾向

```
CREATE VIEW ビュー名 AS select ・・・・・;
ALTER VIEW ビュー名 ENABLE QUERY OPTIMIZATION;
```

```
RUNSTATS ON VIEW vtest1 WITH DISTRIBUTION;
```

Joinする際の表間の相関関係を収集できる

表の再圧縮（辞書再作成）

移行前のテストデータで辞書ができた場合、
移行作業で辞書再作成を検討

- 圧縮辞書作成のタイミング
  - ある程度の行数をINSERT（自動辞書作成：ADC）
  - LOADのANALYZEフェーズ
    - ANALYZEフェーズは以下のLOAD時に発生するため、データの大幅な変化によって圧縮率が劣化した場合には、**REPLACE**か、**空にしてからのLOAD**を検討する

LOAD処理	辞書再作成
LOAD with REPLACE	**有り**
LOAD with INSERT(new、empty table、Load /dev/null)	**有り**
LOAD with INSERT(row delete、truncate table、Import /dev/null)	**なし**

```
SQL3500W  The utility is beginning the "ANALYZE" phase at time "10/03/2017
15:33:36.148905".
```

最適化クラスの調整

単純なSQLは低く、複雑で長時間のSQLは高くすることを検討する

- アクセスパスを選択する際の最適化クラスを設定できる
  - ０～９の値にて表現 （0、1、2、3、 5 (default)、7、9）
  - 数字が高いほどより複雑な最適化を行う
    - より良いアクセスパス発見の可能性は高まる。SQLをコンパイルする時間は長くなる
  - 単純なSQLが大量に実行されるOLTP環境では、2を推奨
    - より良いアクセスパスを探索する以前にSQLを実行してしまう
    - 複雑なSQLでは、一般に応答時間も長いため、より良いアクセスパスを探索する価値がある。（5または7を試す）
  - 様々な設定方法が利用可能
    - DB構成パラメーター （dft_queryopt)
    - bind/prepコマンドのオプション (queryopt)
    - SQLステートメント （set current query optimization）
    - db2cli.iniファイル （DB2CLIOPTIMIZATION）

SQLの変更／分割

統計情報の少ない変化でもアクセスパスが変化する可能性が高まる

- Join数が多く、条件も多い場合、不適切なアクセスパスが選択されるリスクも高まる
  - 絞り込みの条件を強くできないか検討
    - 条件の追加：より絞り込みの高い条件の追加など
    - 条件の強度を高める：LIKE条件で指定する文字列の最低限文字数を設定など
    - 条件をリテラルにする：削除フラグなど偏りのある条件はパラメータマーカーをやめる
  - SQLの記述方式を変更
    - 履歴データを含めた表において、最新のレコードのみを選択
      - 副照会でMAX()を最新レコードを取得するのではなく、ROW_NUMBERを利用する。
    - カーソルを使用したループではなく、一括処理
      - ループ処理はシーケンシャルとなり、MPPの並列処理特性が活かせない
      - JOINやMERGEなどを利用して一括処理できないか検討する

SQLの記述方式を変更することで、効率的なアクセスパスへの変更を検討

SQLの変更／分割

- SQLの変更例
  - 例）最新レコードを取得する方法
    - 副照会のMAX()で最新レコードを取得する例

2回の表アクセスが発生

```
SELECT cust_id, seq, price
  FROM Receipts R1
 WHERE seq = (SELECT MAX(seq)
                FROM Receipts R2
               WHERE R1.cust_id = R2.cust_id);
```

    - ROW_NUMBER関数を利用して最新レコードを取得する例

```
SELECT cust_id, seq, price
  FROM (SELECT cust_id, seq, price,
                       ROW_NUMBER() OVER (PARTITION BY cust_id ORDER BY seq) AS row_seq
          FROM Receipts ) WORK
 WHERE WORK.row_seq = 1;
```

SQLの変更／分割

- 効率的な順番でJoinさせるべく、SQL分割して段階的に処理
  - DECLARE GLOBAL TEMPORARY TABLEによって一時表を用意し、段階的な処理にする
  - DGTTは、同じセッション内で再利用可能。
  - 行オーガナイズ表となるため、UOW内で索引追加＆Runstatsが利用可能

- 各DBパーティションで、カーソル処理を並列実行
  - アプリケーションロジックの必要性からカーソル処理が必要な場合、各DBパーティションで、DBパーティション内のみのカーソルにより並列実行

```
DECLARE c1 CURSOR FOR SELECT cust_id, seq, price
FROM Receipts
WHERE DBPARTITIONNUM(cust_id) = current DBPARTITIONNUM;
```

接続したDBパーティション内のデータのみを対象

参照制約（Informational制約）の利用

- Informational制約としての参照制約を作成できる
  - 表間のリレーションがわかるため、Joinなどで正確な見積りができるようになり、より適切なアクセスパスが選択される。

```
alter table DISTCUST11
add constraint FK_DISTCUST11
foreign key(CUST_ID,NAME) references DISTCUST12
(CUST_ID,NAME)
NOT ENFORCED NOT TRUSTED ;
```

NOT ENFORCED：データの追加検査はされず、SQLコンパイラーに利用されるのみ
NOT TRUSTED：データが、制約に適合していない可能性があることを指定

参照制約によって、Joinでの絞込みが適切に評価される。

```
select a.* from DISTCUST11 a,DISTCUST12 b
where a.CUST_ID=b.CUST_ID and a.NAME=b.NAME ;
```

組み込み最適化ガイドラインの検討

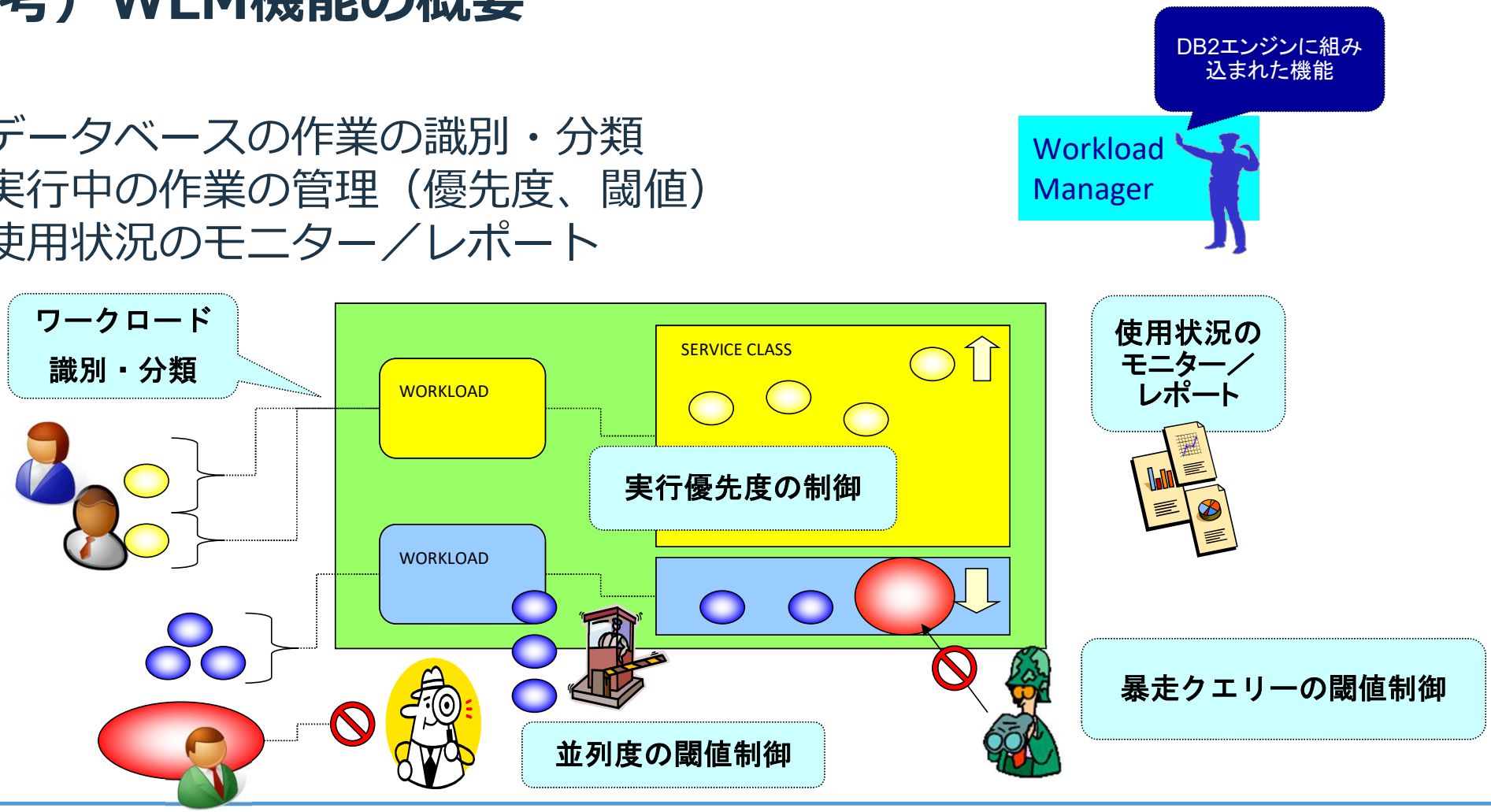
- SQLレベルで、Joinの順序などが指定可能
  - 各種チューニングが全て試された前提で、利用を検討
  - Joinは基本的にHSJOINであるため、Join順序の調整に利用

```
select e.empno,e.lastname,d.deptname
from employee as e
inner join department as d
   on e.workdept=d.deptno
/*
<OPTGUIDELINES>
   <HSJOIN>
      <ACCESS TABLE='e' />
      <ACCESS TABLE='d' />
   </HSJOIN>
</OPTGUIDELINES>
*/
;
```

（注）Db2のオプティマイザーによって最適なアクセスパスが選択されないケースは、多くの結合や複雑な条件などが要因の可能性が多く、そのようなケースでは組み込み最適化ガイドラインを利用した指定も複雑となるため、最適なアクセスパスとならない可能性もある

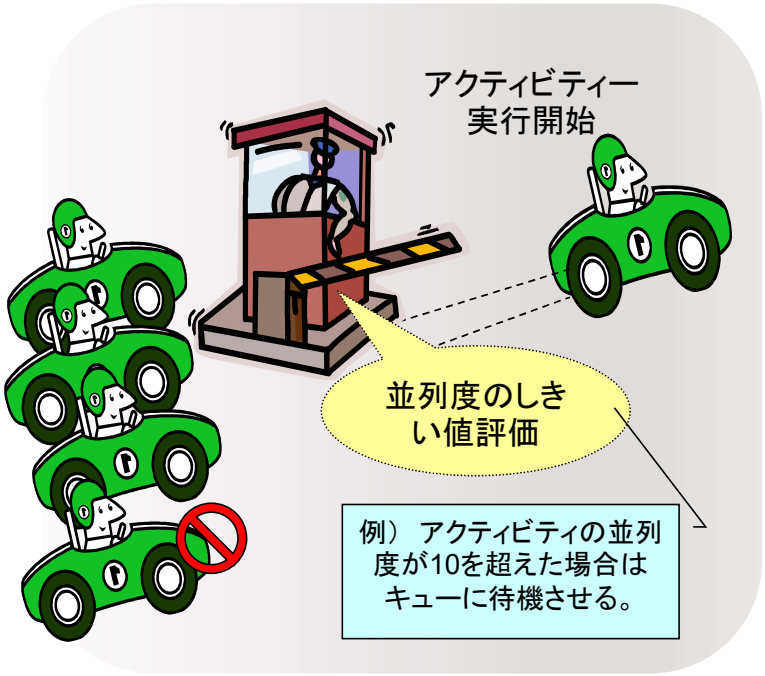
参考）WLM機能の概要

- データベースの作業の識別・分類
- 実行中の作業の管理（優先度、閾値）
- 使用状況のモニター／レポート



WLMの閾値制御

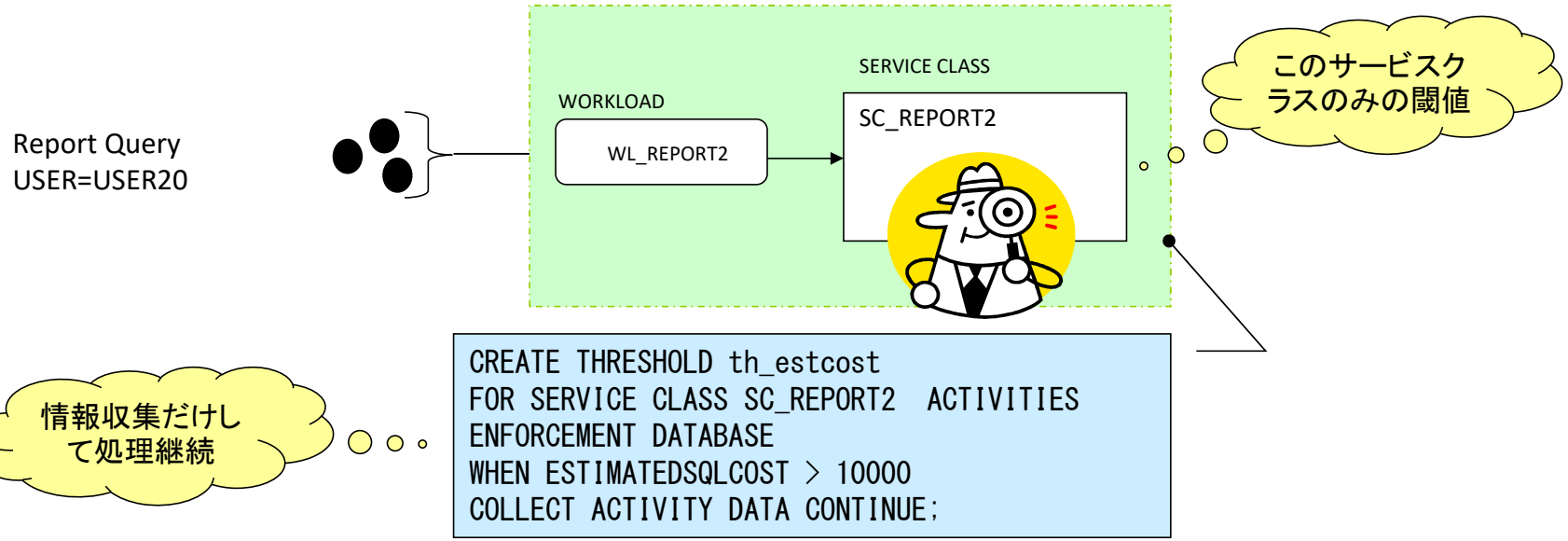
- THRESHOLD
  - 予測的閾値、並列度の閾値、反応的閾値がある
  - 違反時には、停止、REMAP、情報収集などのアクション



WLMのコスト監視　(実行前の閾値チェック)

実行コストの高い処理をモニターすることができる

- 見積もりコスト(ESTIMATEDSQLCOST)による閾値
  - 想定外のコストとなるSQLの詳細情報をモニターする
  - この閾値は、以下の適用範囲を指定できる
    - データベース、サービス・スーパークラス、サービス・サブクラス、ワークアクション、ワークロード、ステートメント



WLMのコスト監視　(実行前の閾値チェック)

- Thresholdの作成

```
CREATE THRESHOLD th_estcost
FOR SERVICE CLASS SC_REPORT2  ACTIVITIES
ENFORCEMENT DATABASE
WHEN ESTIMATEDSQLCOST > 10000
COLLECT ACTIVITY DATA STOP EXECUTION
;
```

EstimatedSQLCostが10000を越えるSQLの詳細情報をモニターし、実行停止

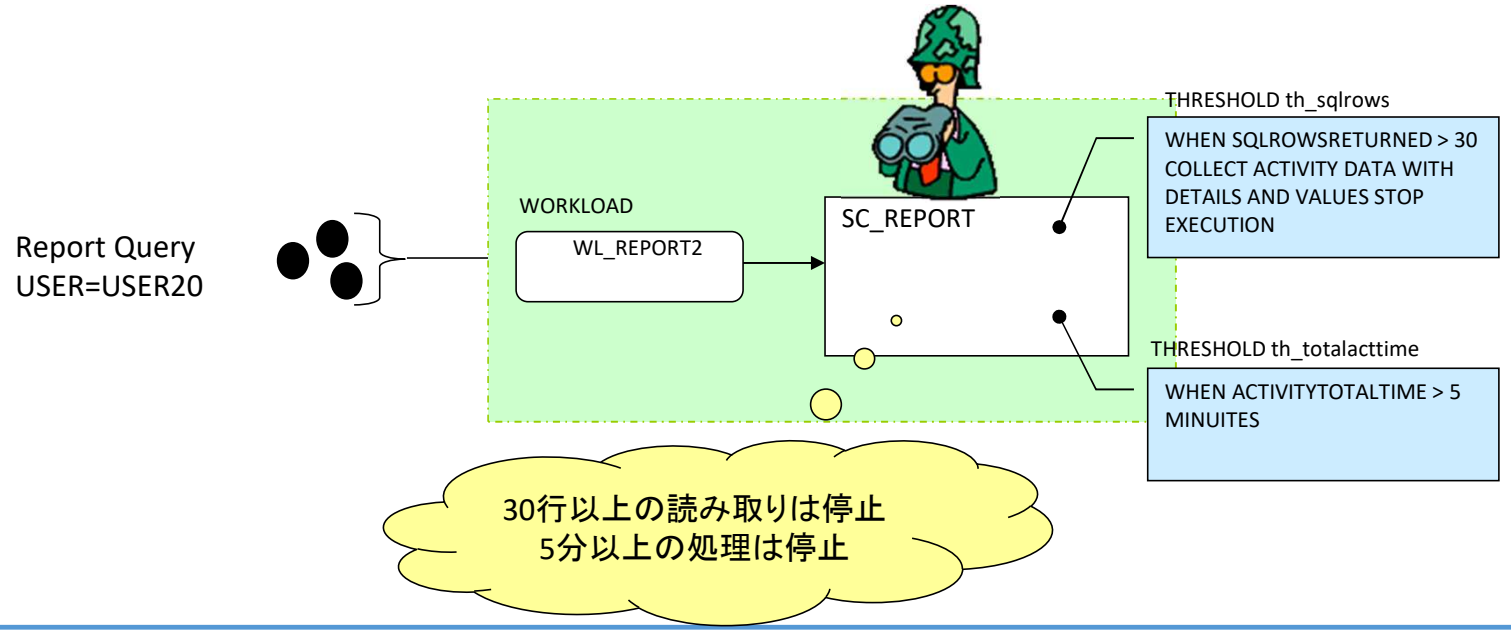
- Threshold 違反の際のエラー出力

```
$ db2 "select * from db2inst1.stock"
SQL4712N  The threshold "TH_ESTCOST" has been exceeded.  Reason code = "7".
SQLSTATE=5U026
```

MEMO： 閾値超えアクション＝STOP EXECUTIONの場合、閾値を超えてエラーが返りますが、接続は保持されます。閾値越えアクション＝CONTINUEとすると、イベントモニターに履歴だけ残してアクティビティーを継続します。

WLMの高負荷SQL監視　(実行後の閾値チェック)

- 高負荷SQL(SQLROWSRETURNED ,ACTIVITYTOTALTIME)の設定
  - 大量の戻し行数、長時間SQLの作業を実行停止させる
  - この閾値は、以下の適用範囲を指定できる
    - データベース、サービス・スーパークラス、サービス・サブクラス、ワークアクション



WLMの高負荷SQL監視　(実行後の閾値チェック)

- Thresholdの作成

```
CREATE THRESHOLD th_sqlrows
FOR SERVICE CLASS SC_REPORT2 ACTIVITIES
ENFORCEMENT DATABASE
WHEN SQLROWSRETURNED > 30
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
STOP EXECUTION
```

30行以上の戻り行で
実行停止

- Threshold 違反の際のエラー出力

```
$ db2 "select * from db2inst1.stock fetch first 40 rows only"

S_I_ID      S_W_ID      S_QUANTITY  S_DIST_01               S_DIST_02               S_DIST_03               S_DIST_04               S_DIST_05
----------- ----------- ----------- ----------------------- ----------------------- ----------------------- ----------------------- ------------
1           1           92 9k8AIyGE9CpuXkn1XvnFttyn I1MguokYDeWl5noesnB4mh19 PM6Dph2Nh722AggBfUKJEdmm O9Z1uFI12PNfTdMjJFB59lvp
1           2           67 5Svwx3dDfDWlecJtiQrOw4fp fVqU8uRbvECo2C1RXKUeoitS We1QRwXNZ3nrOcsCIVJcrOAV LpfppU3FyJujH5HS3WabcJ42
【中略】
1          29           67 5Svwx3dDfDWlecJtiQrOw4fp fVqU8uRbvECo2C1RXKUeoitS We1QRwXNZ3nrOcsCIVJcrOAV LpfppU3FyJujH5HS3WabcJ42
1          30           79 ty9TezycOosdEBLFpZKlreGp Cz89XTPIZxH3GhorUxfNy36X ivggYU1we1WOSZBk1dXYMpeu byZoyDHeoXQ3BM755Qx8GH
SQL4712N  The threshold "TH_SQLROWS" has been exceeded.  Reason code = "8".
SQLSTATE=5U026
```

まとめ

- パフォーマンスの問題はDBサーバーのSQLなのか？
  - OS、NW、アプリケーションロジックなど原因はいろいろある
- 遅いSQLの特定の方法はいろいろある
  - dsmtop、Webコンソール、MON_GET表関数など
- 遅いSQLが特定できたら、アクセスパス確認
  - dsmtop、Webコンソール、db2exfmtなど
  - オプティマイザーの見積りが正しいのか疑ってみる
- チューニング
  - 統計情報、圧縮状況、SQLの見直しなど
  - WLMを利用して高負荷SQL制御、並列実行数、モニターなども可能