DDS and SQL - The Winning Combination for DB2 for i

Business and technical benefits of a modern DB2 for i database

Kent Milligan
Dave Hendrickson

IBM Systems and Technology Group ISV Enablement

November 2014

Table of contents

Abstract

This white paper explains the business and technical benefits of modernizing IBM DB2® for i databases. Modernizing a DB2 for i database with Structured Query Language (SQL) and data-centric programming delivers benefits that range from reduced IT expenses to enhanced developer productivity and improved application performance and scalability. This paper explores these benefits in greater details as well as educates you on the traits of a modern database.

Foreword - A customer perspective

Modernizing Your IBM i: Use it or lose it!

For those of us who have spent a career on the IBM i, we have a deep appreciation for the incredible platform IBM created. And yet, we are letting that very success become an impediment to its future. From the launch of the IBM® System/38 in 1979, IBM has woven a rich set of features into the OS. Where other platforms require separate solutions for everything from security to communications, IBM chose to create an environment and tool set that allowed us to build robust applications in a singular ecosystem.

For many companies, the migration path from the System/38, the IBM AS/400®, and now the IBM i has provided forward and backward compatibilities that would be impossible on other platforms. As a result, legacy systems have remained viable, in some cases for decades, allowing organizations to run with lean budgets, stable applications, and very small IT staffs. We've had a logical progression of building applications using DDS (Data Description Specifications) for database objects and RPG as our programming language of choice.

Don't get me wrong, we often expressed our desire for modernization, particularly for the user interface. Some shops have been active adopters of various new technologies as they became available. But the roadmap IBM presented to us just never seemed to make it into our annual budgets. We attend conferences and came back fired up about what we heard, but it was deemed too difficult to incorporate those technologies into our never-ending backlog of projects and deadlines. We are allowed to nibble at the edges, but real modernization remains a struggle.

Outside the IBM i community, different standards, market forces, and trends sent much of IT in another direction. The green screen applications that are so common and productive in our world have graphical user interfaces on other platforms. Rather than DDS, much of world builds relational databases following the standards of SQL.

Meanwhile, IBM i shops have upgraded from one OS release to the next without taking advantage of the many enhancements for modernization that IBM has introduced. Be it SQL, XML, SOA, or any number of other features that promote openness, IBM has advanced the IBM i with each and every release.

The misconception that the IBM i is antiquated is in many ways one of our own making. How can we expect to draw new talent and solutions to the platform if we aren't taking advantage of the most advanced tools the IBM i has to offer?

Most IBM i modernization tools and projects seem to focus on the user interface. But true modernization happens at three levels. The database, the development environment, and the user interface. It can be argued that the biggest gains and the foundation for real modernization should begin with the database.

SQL adoption has lagged on the IBM i despite the fact that, for the past 10 years, IBM's modernization of DB2 for i has clearly focused on SQL over the traditional implementation of DDS. In contrast, the ubiquitous nature of SQL's adoption on other platforms is obvious when you search online for SQL programming books.

Why? It's not the platform. The reliability of the IBM i as a server is legendary, with price, performance, and scalability that competes head to head with any other platform. It's a combination that makes it an ideal server for SQL workloads. It was one of the first platforms to be 100% compliant with the core level of the ANSI 2008 SQL Standard. And it offers many features that make it relatively easy to port databases from MySQL, SQL Server, Oracle, and others.

Generally speaking, I believe we need to educate ourselves and our management on the business and technical advantages of modernizing with SQL. This will allow us to move forward with the new features, the openness, and the interoperability with other platforms that comes with adopting SQL.

The initial task of modernizing with SQL has to remind you of the challenges we faced with Y2K, another modernization project that many wanted to avoid. The requirement was obvious. And yet organizations looked at their backlogs and budgets and hoped it would go away. Eventually, resources were allocated and the task was completed. There was an absolute deadline that we could not avoid.

The same absolute deadline does not exist for modernizing your applications. But the eventual results will be the same. Many applications that could not support Y2K date logic were replaced rather than modified. I would suggest to you that the same forces are at work here.

If you intend to continue developing software, you will be doing so on an SQL database. It's really just a question of ***when,*** not ***if***. It is also a question of what platform will be running that SQL database.

Will it be the IBM i or something else?

Dave Hendrickson, President
Tree Line Data, LLC

The new DB2 for i

Stability and reliability are key reasons why the IBM i platform is viewed as one of the best servers in the industry on which you can run your business. Applications just keep running despite changes to the underlying processor, memory, storage, and database technologies. This holds true for new IBM i applications as well as those applications originally developed for its predecessor platforms: IBM System i®, IBM iSeries®, AS/400, and System/38.

Ironically, the greatest strengths of IBM i have caused its greatest weakness. Because applications continue to run without change and the system is so easy to manage, many IBM i clients have failed to modernize their systems along with the platform. Specifically, a great many have overlooked the fact that the system's renowned integrated relational database, known as DB2 for i, has been changing right before their eyes. This transformation has been taking place with advancements in the SQL support for over a decade. Just as IBM was visionary in delivering one of the first commercial database products with relational capabilities with System/38, IBM has clearly chosen SQL as the strategic database interface for IBM i – just as the rest of the industry has standardized on SQL.

Consequently, almost all of the new features and functions IBM has added to DB2 for i requires the usage of SQL in order to benefit from them. Table 1, which contains a comparison of the major DB2 enhancements made to the SQL and non-SQL interfaces since Version 5 Release 1 (V5R1), illustrates the commitment from IBM. The new SQL-only features offer a large assortment of advanced functionality that ranges from automatic key generation with Identity columns and Sequences to OnLine Analytical Processing (OLAP) capabilities, to high-speed linguistic text searches, to self-learning query optimization.

Those IBM i clients who have failed to keep track of this SQL metamorphosis are missing out on a plethora of new SQL-based functions that can make it easier for their IT teams to meet the ever-changing list of business requirements. In addition to taking advantage of the new functionality, organizations who embrace SQL typically find that:

- Developers can deliver on requirements faster because the modern SQL features allows the DB2 engine to do more work on behalf of the application.
- More business processing performed by DB2 means that developers have to write and maintain less application code.
- Improved performance and scalability can be a side benefit as the DB2 engine can perform some business processing faster and more efficiently than a high-level language application because the engine operates at the lowest levels of the system with sophisticated algorithms.

Enhancements to non-SQL interfaces	SQL enhancements
Unicode support - UTF-16 and UTF-8	Unicode support - UTF-16 and UTF-8
Binary character data type	Binary character data type
CRTLF PAGESIZE parameter	CREATE INDEX PAGESIZE keyword
Larger decimal support	Larger decimal support
SSD enablement for physical and logical files	SSD enablement for tables and Indexes
	XML, National Character, and ROWID data types
	Identity column attribute
	Hidden and Automatic Timestamp column attributes
	Field Procedure column attribute
	Sequence object
	Column-level and Instead-Of triggers
	Merge statement
	OLAP and Super Group expressions
	Create Table from Select and Insert from Select
	SQL functional indexes
	XML Publishing and Decomposition functions
	IBM OmniFind Text Search Server
	SQL Query Engine (SQE)
	SQE Result Set Caching
	SQE Autonomic Indexes
	SQE Self-Learning Query Optimization and Adaptive Query Processing
	SQE Encoded Vector Index fast path for aggregate processing
	SQE In-memory Database Enablement
	IBM i Navigator Plan Cache Tool

Table 1: Comparison of DB2 enhancements since Version 5 Release 1 (V5R1)

The efficiency of data-centric programming

This approach of using SQL to have DB2 do more work on behalf of the application is known as data-centric programming. A better understanding of this data-centric approach with the traditional record-level access methodology can be garnered by comparing the graphical representation of these two approaches, shown in Figure 1.

Moving data processing into the database

With the traditional approach, the developer uses the native record-level access interfaces to specify the exact step-by-step instructions for DB2 to follow in the consolidation of data from the four different database tables in this example. This means that not only does the developer have to create application code that implements the requested business functionality, but the developer also must invest time in directing and controlling the relational database processing. A contrast to this approach and effort is the portrayal of the SQL data-centric programming model. A developer utilizes SQL to submit a high-level request to DB2 to consolidate the data from the four database tables. At this point, it is up to the DB2 query optimizer to analyze the request and select the best performing low-level step-by-step implementation. This data-centric approach enables developers to focus their time on the creation of application code for business processes while leaving the data processing for DB2.

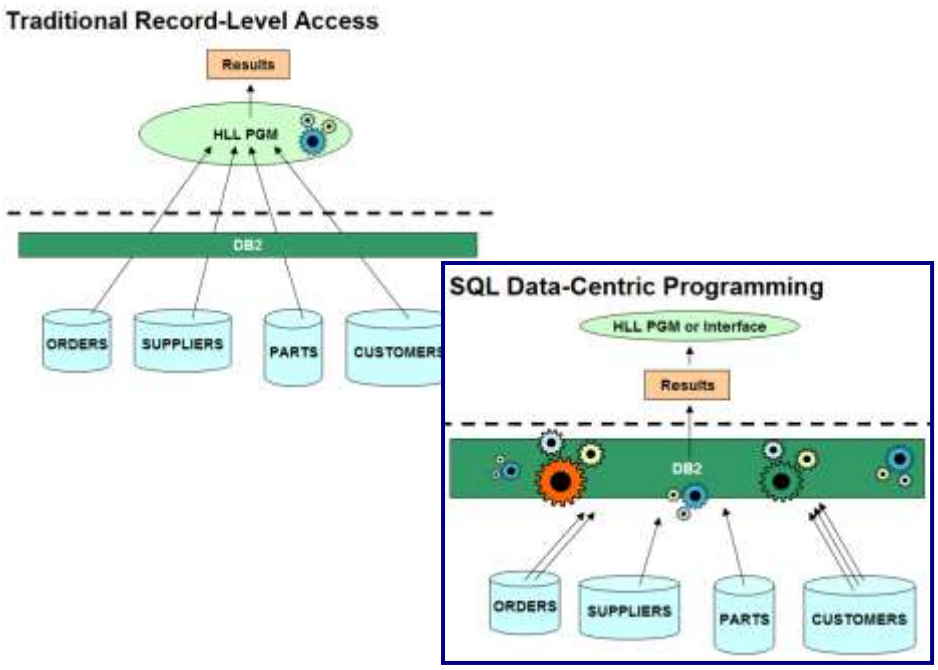

Figure 1: Comparison of traditional and data-centric programming

Moving business rules into the database

Another benefit of the modern data-centric approach is enhanced data integrity. Data-centric development enables programmers to embed simple business rules into the database objects themselves, meaning that those business rules will be enforced regardless of the interface (green-screen, web, and so on) used to access the data. For instance, any application that generates an order needs to have logic to validate that the order references a valid customer account. A data-centric developer moves and centralizes that logic into the database so that validation is enforced for every interface and application that generates an order. With the traditional approach, the validation

logic would need to be replicated each time that a new interface is added to the system. Not only does this duplication of logic require extra time from the developer, but it also increases the chances of data integrity issues. If the development team forgets to replicate the logic, then orders can be placed into system without any validation.

What SQL means for most businesses

In summary, the benefits of using a data-centric approach that features the usage of SQL and modern database functionality are:

- **Faster IT response to business requirements**
  Data-centric programming enables developers to focus their efforts on delivering business logic and relieves them from the responsibilities of implementing relational data processing. The result is a reduction in the amount of code that developers have to write as they rely more on the DB2 engine to take care of the implementation details. The SQL interface also offers a much larger set of features and functions to choose from which provides developers more opportunities to offload processing into the DB2 engine instead of creating application logic to perform the processing.

- **Reduced software development and maintenance costs**
  The movement of common business rules and processes into the database engine results in a smaller amount of software code that the IT teams need to write and maintain. These centralized business rules and processes are automatically reused instead of being recreated as new applications and business functions are brought online.

- **More reliable data to run your business on**
  Embedding common business rules into your database objects means that these rules are enforced regardless of the interface used to access the data. This centralization of rules within the database eliminates the exposure of tools and devices bypassing the rules and controls associated with your business data.

- **Improved performance and scalability**
  A data-centric approach results in DB2 performing tasks that are normally done at the application layer. Performance of the applications can be improved overall because the DB2 engine is doing the processing at a lower level in the system than the original application logic. In addition, data-centric applications benefit from the fact that IBM is putting almost all of its investment in SQL performance enhancements, which include self-tuning capabilities.

- **Sensitive data protection**
  Sensitive data is more easily protected with encryption using the SQL Field Procedure support. This technology permits the deployment of column-level encryption solutions with minimal disruption to existing applications.

- **Minimize risk with clear documentation**
  The documentation and structure of many DB2 for i databases probably exist only in the minds of developers. With the proliferation of data modeling tools for SQL databases, complete and easy-

to-understand documentation is just a click away after moving your databases over to SQL.

- **Access to a larger pool of talented developers to build IBM i solutions**
  While it is a fact that there are fewer programmers versed in the IBM i DDS and record-level access interfaces coming out of universities, the vast majority of new programmers do have SQL experience. Using DB2 for i support for procedural SQL objects, such as stored procedures and functions, a new SQL developer can quickly contribute to the delivery of an IBM i application with a very small learning curve.

Keeping your data on IBM i

At the time DDS and the native record-level interfaces were introduced, they were powerful features for relational data access, and they served many IBM i clients well. In fact, the capabilities of the integrated relational database was a key reason that such a large number of business solutions and applications were developed to support IBM i and its predecessor platforms in the first place.

Increasingly, though, the DDS and record-level access interfaces exhibit limitations in supporting new application requirements because these interfaces are not being enhanced with modern relational database features, such as XML integration and publishing, or with other data-centric features that enable some business processing to be performed by DB2.

The DDS and native record-level access capabilities for DB2 are not going away, but they also are not providing new features that make life easier for developers. Thus, adoption of SQL is a critical success factor in being allowed to keep your company's data stored in the highly-secure DB2 for i databases that you have grown to trust. When development teams are not able to meet business requirements, they run the risk of applications and data being moved to another system. In a long-term, it is really not a question of whether or not you should use SQL, it is a question of which platform your company will use for SQL. Very simply, **SQL is the strategic database interface for nearly all platforms, including IBM i**.

Evolving from DDS to SQL

True to its legacy, the SQL implementation on DB2 for i followed the lead of the original relational database support when it comes to integration and simplicity. The SQL runtime engine is integrated and built into the IBM i operating system, making SQL available on every IBM i system. In terms of simplicity, the SQL implementation includes features that maintain the reputation of DB2 for i for being simple to use and manage on a daily basis. These SQL ease-of-use features include self-tuning capabilities, such as automatic statistics collection and self-learning query optimization.

The methodology: Low-impact evolution

Another unique capability of the DB2 for i integrated design is that it provides a single relational database with two interfaces. This is a huge benefit for IBM i clients because it means that SQL usage can be a gradual **evolution** instead of a **revolution**. The advantages of SQL can be realized without first having to convert all of your databases to SQL. DB2 objects created with DDS can be accessed with SQL and objects created with SQL can be accessed with the native record-level access interface. In fact, a methodology exists for converting object definitions to SQL without

requiring any changes to existing applications. This methodology has proven itself repeatedly over the last 10 years and has been refined by IBM based on modernization engagements with clients.

This methodology not only supports a transparent conversion of DDS to SQL, but also provides a framework where you can enhance the SQL version with newer capabilities, such as identity columns – again **without any impact to the set of existing applications**.

If your modernization project also includes re-engineering or changing the applications that reference the new SQL database, then a transparent conversion methodology is not required. New technologies such as the *IBM Rational Open Access: RPG Edition* product have emerged which allow applications to migrate to SQL data access with a minimal amount of application changes.Instead of requiring a full-blown conversion to SQL, you can adopt SQL for those applications and databases where it can deliver maximum business value. The ability to mix and match these two DB2 for i interfaces provides IBM i customers tremendous flexibility as they modernize their databases and applications with SQL.

Education and new tools

As with any new technology, SQL requires a change in thinking and the judicious use of tools to fully exploit its potential. Tools are available to automate the SQL conversion process and model your databases. Education offerings exist to help developers leverage the set processing capabilities of SQL and shift to a data-centric approach where business processing is moved from the applications into DB2. The resources are all in place for a smooth transition to SQL, which places your team in position to be more responsive to business requirements.

The impact of adopting SQL: A closer look

In addition to benefiting your organization to be more judicious and provide faster time to value in a dynamic business climate, IBM i developers clearly benefit from the functional advantages of SQL. Even if none of the SQL-only features of DB2 for i are utilized, SQL usage still provides benefit because it is an industry standard. Using SQL DDL to define and create DB2 for i databases puts you in the mainstream of all other relational database management systems. Being part of the larger SQL community gives IBM i developers access to a larger array of database management tools. Out of this expanded toolset, database modeling tools hold the most potential.

Mainstream data modeling and database documentation

Data modeling tools provide a graphical diagram and navigation interface that supports both logical modeling and physical modeling. A modeling tool provides organizations the ability to quickly create, organize, and visualize the data store that is needed for a specific business requirement.

A logical data model is used to represent all the definitions, characteristics, and relationships of business data in an application or enterprise environment. A logical data model normally uses entity-relationship diagrams to represent business data entities, relationships, and attributes in a natural manner that is independent of an underlying technology or platform. Figure 2 shows an example of a logical data model. The logical modeling process also makes it easier to create a normalized data model, which improves the efficiency of your database by eliminating data redundancy.

The physical data model is a representation of the database design that ties it to a specific technical implementation of the database. The physical data model takes into account the capabilities of the underlying database management system (DBMS) and contains all of the DBMS objects needed to support and enforce the various data relationships. The physical model is used to generate the SQL source code needed to create the database on the specified DBMS – a big time saver for your developers. A physical model is typically derived from the logical data model, but can also be reverse-engineered from an existing database. Multiple physical models can also be generated from the logical model for the same DBMS or other DBMSs.

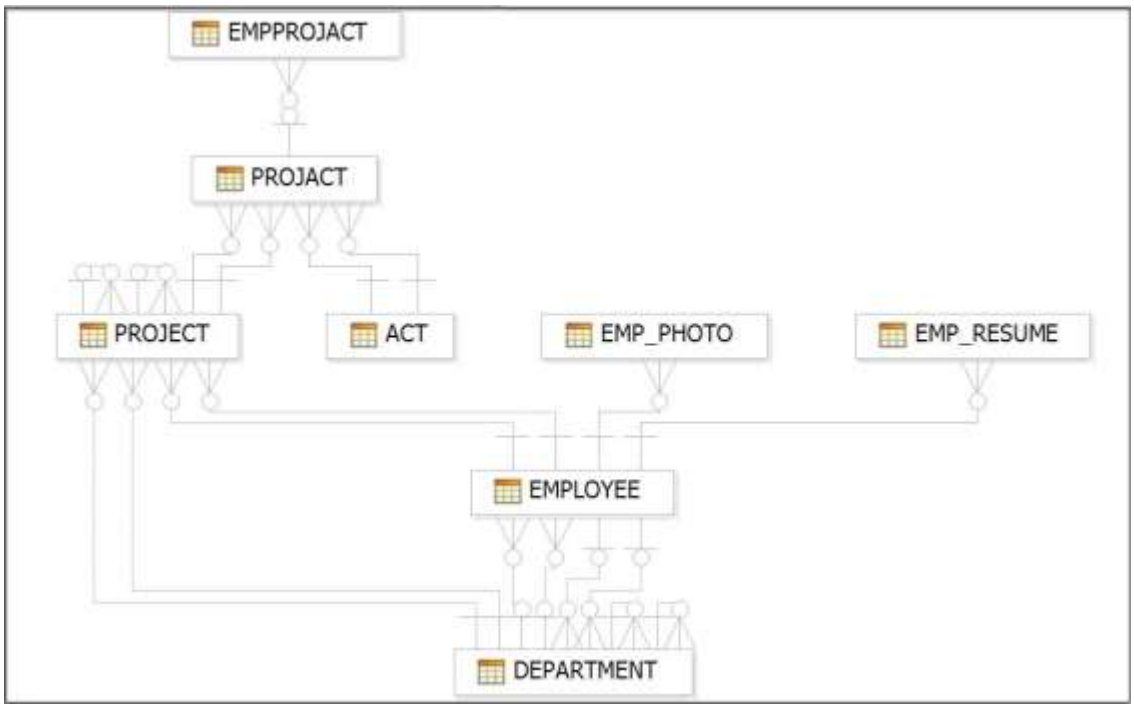

Figure 2: An example of a logical data model from IBM InfoSphere™ data architect

Facilitating communication and teamwork

Whether the data model is built for a new application or reverse-engineered from an existing database, these graphical models can serve as a valuable piece of documentation for an organization. The high-quality data diagrams produced by modeling tools make it easier for teams to understand and refine their database design visually. In addition, these diagrams make it simpler to describe the database relationships and dependencies to the business users. This can help everyone in a company to understand what business data entities and relationships exist and how they can be best utilized to drive business value.

Facilitating database changes

Most database modeling tools actually go well beyond the function of providing a visual representation of your databases. Modeling tools also help manage the life-cycle of a database. When a new business requirement requires a change in the database design, a modeling tool can make it easier for a developer to implement these changes. Making the database *design* change is just the first step. The next step, which can be much more complex and time consuming, is determining the *impact* of that change. As illustrated in Figure 3, one simple change to a table might impact a large set of dependent SQL objects, such as views, triggers, constraints, and indexes. Data modeling tools not only perform impact analysis to show the scope of a database change, they then automatically generate code to implement a change while preserving any existing data. The change impact analysis features of a data modeling tool enables your team to implement database changes with speed and accuracy.

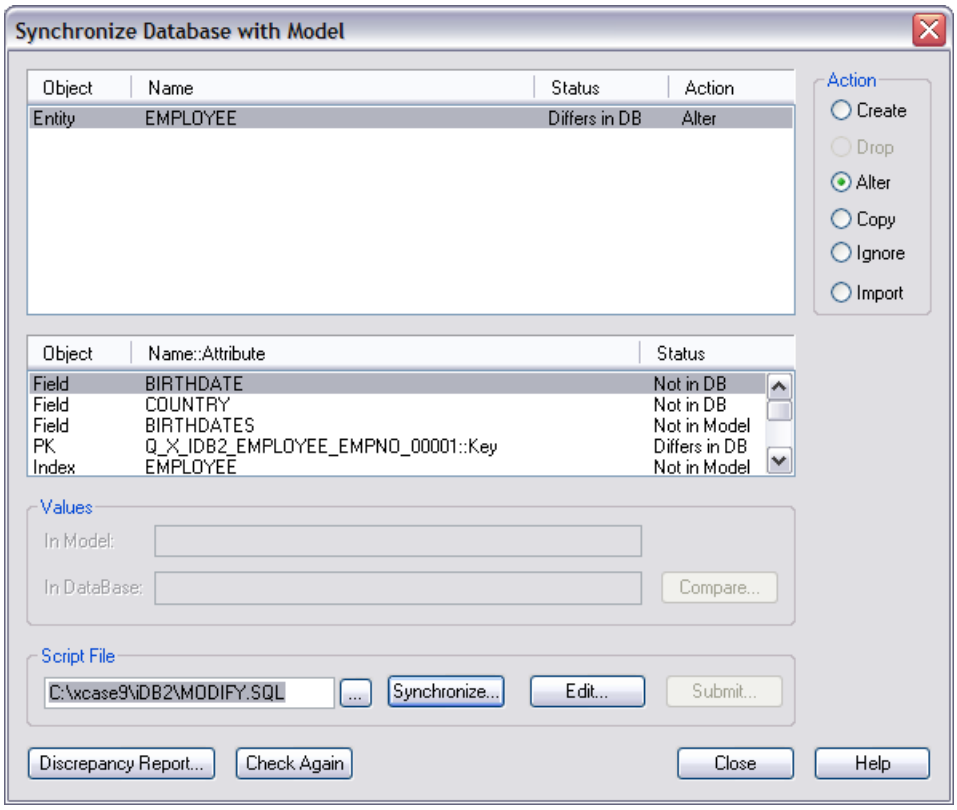

Figure 3: Impact analysis of proposed database change from Resolution Software's Xcase for i

From data modeling to database modification, a data modeling tool enables these tasks to be completed efficiently and accurately. This is a clear advantage for those using SQL to create databases on IBM i. Teams that continue to create and maintain DDS-created databases will find their efforts hampered by a void of modeling tools that fully support DDS.

Increased data integrity with referential integrity

Referential integrity (RI) is one of the two key data integrity rules associated with the relational database model. The basic intent of the referential integrity rule is to ensure that relationships between tables are accurate and enforced at all times. For example, when an order is placed, your company needs to know that the new order is referencing a valid customer. If not, your company will have no way to bill the customer. Similarly, you would not want to allow a user to delete an existing customer from the database, especially if orders exist for that customer. In the event that business requirements do dictate a deletion, the customer's orders also need to be deleted or orphan records will be created, corrupting the integrity of your database.

DB2 for i supports the enforcement of these simple business rules through foreign key constraints to ensure that referential integrity is maintained. A foreign key constraint would be created on the orders table to guarantee that each order written to the database references a valid customer.

The DB2 implementation of RI is known as declarative referential integrity support because a foreign key constraint is part of your database definition. The integrity and reliability of your business data increases because referential integrity is being enforced at the database level. This centralized enforcement of the

business data relationships means that RI is maintained regardless of the interface used to access this data. Whether the interface is an IBM 5250 emulator, web browser, or database utility, a declarative referential integrity implementation maintains consistency and accuracy across all your data relationships. Gone are the days of a new application or device causing data integrity issues within your databases.

The dangers of traditional referential integrity methods

Traditional IBM i programs have long maintained referential integrity by performing data validation within the application. Application-enforced referential integrity works very well when an application program is used to change the data. However, this method creates a significant data integrity exposure when other interfaces, such as graphical SQL editors or the Data File Utility (DFU), are used to make database updates. The proliferation of new interfaces (for example, web browsers, smart phones, and so on) being linked to IBM i systems makes data integrity an even bigger issue.

Application-based RI enforcement also does not scale very well from an application development investment perspective. As new applications are developed, a programmer must duplicate the referential integrity enforcement logic into their data access routines. After the coding has been completed, the developer must test and verify that the referential integrity enforcement logic works correctly within the new application. If a data relationship ever changes, the development team will have to change and retest every application containing the embedded RI logic.

The simplicity and power of declarative referential integrity

The declarative referential integrity implementation of DB2 reduces these software development expenses. With declarative RI, the enforcement logic is centralized in a single spot – the database engine. The act of enforcing business rules at the database level is the database equivalent of modular programming, which facilitates code reuse and consistency. If you implement this modular approach, all your applications can automatically reuse the simple business rules enforced by referential integrity constraints. When business requirements dictate a change in data relationships, the development organization implements the change by simply modifying the database definition. They no longer have to change, test, and recompile multiple applications.

Another benefit of RI rules being included in the database definitions is that it is much easier for IT organizations to document all of their database relationships. Referential integrity documentation is well centralized within the database descriptions and displayed easily with data modeling tools. In contrast, creating the documentation for application-enforced RI is a daunting challenge because that logic is scattered across multiple program objects. Clear documentation of database relationships makes it easier for developers as they plan for new business requirements and manage the complexities of having to support multiple applications or platforms.

While declared referential integrity can be implemented for DDS-created databases, it is not ideal because the data relationships cannot be encapsulated within the DDS description. The RI relationship is defined and created with a system command that is separate from the DDS definition of the database object. This separation results in the referential integrity relationships being more difficult to detect than those relationships implemented using SQL. This self-describing aspect of SQL definitions is shown in Example 1. A developer reviewing this SQL definition can quickly see that the **orders** table has relationships with **customers** and **parts** tables that are being enforced with foreign key constraints. These declared constraints ensure that any order written to the database references

a valid part and customer; basic business rules that any company would want to be enforced. SQL definitions make referential integrity relationships clear and obvious to the developer as you can embed them within the DB2 object definition.

```
CREATE TABLE orders(
         ord_number    INTEGER PRIMARY KEY,
         ord_quantity  INTEGER CHECK(ord_quantity>0 AND
                                     ord_quantity<1000),
         ord_amount    DECIMAL(7,2),
         ord_partid    CHAR(4),
         ord_custid    CHAR(6),
         CONSTRAINT ordcust FOREIGN KEY(ord_custid)
                            REFERENCES customers(custid),
         CONSTRAINT ordpart FOREIGN KEY(ord_partid)
                            REFERENCES parts(partid) )
```

Example 1: Example of SQL self-describing referential integrity definitions

Example 1 includes a second type of database constraint that you can create to improve data integrity. The check constraint defined on the quantity column enables developers to have the DB2 for i database ensure that the column contains a set of values that are valid from a business perspective. Business rules and processes often dictate that a column or field in a database can only contain a fixed set of allowable values. The check constraint in this example guarantees that the quantity column will contain only an integer value greater than 0 and less than 1000. Thus, the domain of legal values for the quantity column includes the values between 1 and 999, both inclusive. Similar to referential integrity constraints, check constraints are enforced across every interface.

In addition to increased data integrity, declared database constraints offer the potential of improved SQL performance. As the constraint definitions are part of the DB2 table object, the SQL Query Engine (SQE) optimizer has easy access to the constraints. The SQE query optimizer analyzes the definitions to gain a deeper understanding of the values and relationships within your databases. This deeper knowledge enables the DB2 engine to retrieve data more quickly and efficiently.

In summary, modernizing your database with declarative referential integrity offers several benefits. Data integrity issues are caught right away regardless of the interface. The centralization of the business rules reduces software maintenance as well as provides a foundation for complete documentation. And, declarative RI can result in faster performance of business rule enforcement and reports. Although discovering and implementing relationships within an existing database that does not contain declared referential integrity constraints can be a challenging process, modernization tools are available to simplify and automate this difficult task.

Streamlining development by offloading work to DB2

As discussed earlier, the newer SQL-based DB2 features and functions from IBM provide an easy way for development teams to move more work and processing into the DB2 engine, reducing the amount of application code that must be created, tested, and maintained.

The types of processing that can be moved into DB2 can range from simple business rule enforcement to the manipulation of date and time values to the aggregation of various business measurements. Almost every new DB2 feature enables development organizations to deliver on business requirements with fewer lines of application code. Thus, developers who are not adopting SQL are missing out an opportunity to lighten their workload and reduce software development costs.

You will now examine a few SQL-only features that demonstrate this ability where DB2 performs more of the work. First, consider the SQL support for generating key values. Every application needs the ability to uniquely identify a business entity, such as customer number, transaction identifier, and so on. Accordingly, all applications contain logic that generates the next unique identifier value. Usually, you can offload this simple business logic to DB2.

The DB2 for i SQL interface provides two constructs for generating key values − the Identity column attribute and Sequence object. Both of these constructs allow the developer to control the starting identifier value as well as the increment value to use when generating the next identifier value. Instead of writing and testing application code to handle the incrementing and serialization logic, programmers can let DB2 do all the work when it comes to generating the next value for an identifier or key.

Next, as businesses look for trends and insights to give them a competitive edge, there is a greater demand on IT to produce reports that are more complex and sophisticated. SQL offers a huge functionality advantage when it comes to report implementation. The SQL interface offers a wider range of join types and aggregation functionality. One of the most powerful aggregation features is the Super Groups capability. Super Groups (through the ROLLUP and CUBE expressions) enable developers to group and aggregate data in multiple ways in a single query. Example 2 demonstrates the simplicity of this support. The Super Group ROLLUP expression in this example results in the report returning a summary of the sales data at the region level, country level, and a grand total of all countries and regions.

```
SELECT country, region, SUM(amount)
  FROM sales
  GROUP BY ROLLUP (country, region)
```

Example 2: SQL Super Groups ROLLUP expression

Supporting multiple levels of data aggregation and analysis on a single SQL statement means less coding for your development team. Prior to this new SQL support, multiple queries might have to be coded and run separately in order to present data at different hierarchy levels.

Enhancing application performance with SQL and SQE

Improved performance and scalability can be a benefit of using SQL. When processing is moved from the application to DB2, application performance can receive a boost because the DB2 engine is able to perform the processing at a lower-level within the operating system.

The other driver of improved performance is the SQL Query Engine. The re-engineering project that delivered SQE was born from IBM's decision to make SQL the strategic interface for DB2 for i. In order to more quickly support the latest SQL enhancements and incorporate the latest advancements in query optimization techniques, IBM foresaw the need to deliver a new optimizer and engine designed from the ground up specifically for SQL requests. As a result, the SQL interface has received a substantial number of performance enhancements since the introduction of the SQL Query Engine in V5R2 (Version 5 Release 2).

Self-tuning capabilities are an exciting addition delivered by this SQL performance movement. In recent releases, SQE has been enhanced with technology that enables the optimizer to learn from the performance history of an SQL statement and automatically adjust its algorithms to make future runs more efficient. With the 7.1 release, the SQL Query Engine can actually self-tune a long-running query while it is running with a feature called Adaptive Query Processing.

Faster, more flexible reporting

In contrast, the performance improvements have been few and far between for the native record-level access and non-SQL query interfaces such as Query/400 and Open Query File (OPNQRYF). In fact, the SQL performance advantage is a key reason behind IBM's launch of the IBM DB2 Web Query for i solution as the modern alternative to Query/400. Not only does DB2 Web Query offer a more robust, extensible, and productive reporting solution, but this product also implements the graphical reports behind the scenes with SQL. This SQL foundation enables DB2 Web Query reports to reap the benefits of the advanced performance techniques of the SQL Query Engine. Some IBM i clients have seen performance increase orders of magnitude after converting complex Query/400 reports over to DB2 Web Query definitions. Customers who continue running reports with Query/400, OPNQRYF, or third-party reporting tools that use the non-SQL query interface will continue to miss out on these performance benefits.

Advanced indexing

In addition to the advanced performance algorithms in the SQL engine, the SQL interface offers a bigger arsenal of indexing technologies that can be leveraged to boost DB2 performance. For example, the IBM patented encoded vector index (EVI) technology can be tapped only when creating database indexes with SQL. The DB2 for i EVI support offers similar performance benefits as bitmap index solutions in the industry, but without the maintenance overhead. Most recently, IBM also enhanced the encoded vector index support to provide a performance fast-path for summary and aggregate processing. The SQL support for derived key indexes delivered in the DB2 for i 6.1 release is another powerful tool for tuning reports with complex search criteria, especially for those development teams that are trying to deliver reports with case-insensitive search capabilities, such as the ability to find a match on "Acme Company" with a search string of "ACME Company".

External tuning capabilities

Enhancing performance with SQL's advanced indexing support is also a good example that demonstrates how SQL enables applications to be tuned without requiring program modifications. Once a new index is created for performance tuning, the query optimizer has the ability to automatically use the new index to boost performance the next time the application runs – without requiring any changes to the application.

In contrast, tuning the performance of a traditional application that uses record-level access by utilizing a different logical file requires much more time and effort. The program code must first be modified to reference the new logical file and then compiled and tested. This manual process can delay or prevent performance improvements from being realized by your users. With traditional record-level access programming, the application developer needs to write code for both the data request as well as the low-level data access implementation. An SQL statement within a program simply defines the data request, and the query optimizer takes the responsibility for determining which combination of DB2 objects and algorithms is the fastest way to implement the SQL request when the application runs.

This external tuning capability of SQL is a key reason that SQL-based applications have been able to more quickly exploit the industry-leading scalability and performance of IBM POWER6® and IBM POWER7® processor-based systems. By simply using the DB2 Symmetric Multiprocessing

(SMP) licensed feature, most SQL statements can utilize parallel processing to benefit from all of the multiple-core processors available on IBM Power Systems™. Transforming a traditional record-level access application to benefit from parallel processing usually requires extensive changes to the application design and program code. Modernizing your database and applications with SQL definitely offers the exciting potential of faster performance and better scalability in a variety of ways.

Summary

This paper explained the benefits of a modernized DB2 for i database centered around SQL and data-centric programming. A modernized database delivers both business and technical benefits. Programmers are given access to the larger toolset that SQL provides enabling them to streamline their daily development tasks as well as boost the performance and scalability of their applications. As the centralization of business rules and processes reduces software development and maintenance costs, organizations will have lower IT expenses and have a more reliable set of data to run their business on.

Resources

The following websites provide useful references to supplement the information contained in this paper:

- DB2 for i Modernization Workshop
  This workshop can be offered onsite in a customized format which is tailored to meet the requirements of your applications and development team
  ibm.com/systems/i/support/itc/educ/lsdb2mod.html

- Modernizing a DB2 for i Application Case Study
  ibm.com/partnerworld/wps/servlet/ContentHandler/servers/enable/site/education/wp/9e5a/index.html

- Moving from OPNQRYF to SQL
  ibm.com/partnerworld/wps/servlet/ContentHandler/whitepaper/i5os/OPNQRYF_SQL/move?pageId=pw.sellingresources

- Modernizing IBM i Applications from the Database Up
  ibm.com/redbooks/abstracts/sg248185.html

- Database Modernization Services – IBM Systems Lab Services
  ibm.com/systems/services/labservices/

- Database Modernization Tools
  - IBM InfoSphere Data Architect
    ibm.com/software/data/optim/data-architect/
  - IBM DB2 Web Query for i
    ibm.com/systems/i/db2/webquery
  - Adsero Optima Foundation from TEMBO Application Generation Ltd.
    http://www.adsero-optima.com
  - ARCAD-Transformer DB from ARCAD Software
    http://www.arcadsoftware.com
  - iMOD from CSDA Inc.
    http://www.csdainc.com/imod.html
  - MDCMS from Midrange Dynamics
    http://www.midrangedynamics.com
  - X-Analysis from Fresche Legacy
    http://www.freschelegacy.com
  - XCase for i from Resolution Software
    - Product information - http://xcaseforsystemi.com
    - Free modernization diagnostic utility - http://www.xcaseforsystemi.com/index.php?option=com_content&task=view&id=47&Itemid=79

- IBM i Information Center
  http://publib.boulder.ibm.com/iseries

- IBM i on PartnerWorld
  ibm.com/partnerworld/i

About the authors

Kent Milligan is a senior DB2 for i consultant in IBM ISV Business Strategy and Solutions Enablement for the System i platform. After graduating from the University of Iowa in 1989, Kent spent the first eight years of his IBM career as a member of the DB2 development team in Rochester. He speaks and writes regularly on various DB2 for i relational database topics. You can reach Kent at kmill@us.ibm.com

Dave Hendrickson is the President of Tree Line Data, LLC. His software development career started on IBM mainframes and evolved through the first IBM System/38 Model 5s to today's IBM i. In addition to specializing in data-centric programming services for IBM i, Tree Line Data provides DB2 modernization services using Resolution's Xcase for i. Dave can be reached at dave.hendrickson@treelinedata.com

Trademarks and special notices

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel 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.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The

information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.