# DB2's SQL Compatibility Features:

An alternative approach to traditional migration

Maria N. Schwenger
Database professional
Las Vegas, USA

Kamen N. Tsvetkov
Las Vegas Academy
Las Vegas, USA

*Abstract*-**In today's highly competitive marketplace, many relational database management system (RDBMS) vendors are constantly competing for a bigger share of the market by both attracting new customers (new sales), and by pursuing competitive "win-backs" or, simply said, migrations to their own RDBMS. When advancing such migration opportunities most vendors face strong client demands to prove low cost, effort, and risk of the migration process. The abundance of commonly available "migration tools" has not been able to reduce cost and risks. This paper discusses the unique way of migration to IBM® DB2® for Linux, UNIX and Windows (DB2 LUW) introduced with the SQL compatibility features of DB2 LUW. By retaining the other RDBMS' specific features, this new migration concept (often called simply "enablement") allows an easy, less costly and nearly risk free migration process. This paper outlines the capabilities of the Oracle compatibility feature in offering native execution and support for the proprietary Oracle SQL and PL/SQL syntax. Secondly, the paper outlines the DB2 SQL SKIN feature for applications written in Sybase ASE (Sybase compatibility) by using compatibility emulation. Although each of these features provides a different approach, they both aim to achieve higher compatibility and thus to diminish the migration efforts.**

***Keywords-database migration, database conversion, SQL dialects, enablement, multi language support, SQL database extensions, procedural language, SQL, SQL compatibility***

## I. INTRODUCTION

Although most of the RDBMS vendors today support the bulk of the international standards for database definitions and language processing, almost every vendor aims to provide more robust functionality by offering a large number of non-standard language and interfaces extensions. If we attempt to move an application that was coded to utilize these extensions to a new database platform, a number of those extensions would need to be changed or replaced to preserve the desired functionality. This process of moving an application from one database to another is called a database migration. The higher the number of proprietary features there are, the lower the portability (or compatibility) of the code. This "locked in" model usually leads to a higher cost and risk of the migration process and, in many cases, prevents the clients to attempt migrations. Traditionally, the vendors attempted to solve this issue by offering a variety of migration tools, which can facilitate low cost and mitigate the risk of the migration process by providing automated ways of conversion, deployment and testing, however, even with automation and tooling, the migrations are still expensive, time consuming, and present risk to the business. With version 9.7, DB2 LUW offers a new unique alternative to the traditional migration approach. This version introduced an extended support for SQL and procedural syntax (Oracle's PL/SQL, Sybase's T-SQL, other unsupported by DB2 SQL), which allow the usage of these no longer proprietary language extensions straight against the DB2 LUW engine. This paper discusses the two different SQL compatibility features of DB2 9.7 and the unique way each one addresses the compatibility issues compared to the traditional migration processes.

The rest of the paper is organized as follows: Section 2 gives a background overview of the database compatibility and its role in the traditional migration process; Section 3 describes the DB2 LUW SQL compatibility of Oracle's SQL, PL/SQL, administrative views, and application interfaces; Section 4 outlines DB2 LUW SQL Compatibility with SQL Skin for Sybase ASE emulation as an alternative to the traditional migration approach, and finally, Section 5 concludes the paper with a brief summary of the business and technical values delivered by the two SQL compatibility features offered by DB2 LUW: native language support for Oracle's SQL and PL/SQL and SQL Skin for Sybase ASE emulation.

## II. BACKGROUND

### *A. Background on SQL Standards*

As soon as the concepts of the modern RDBMS and Structure Query Language (SQL) standards were established in the late 70s (first relational model by E.F. Codd at IBM [4], first specification of SEQUEL by Donald D Chamberlin and Raymond F. Boyce [2], and others [3]), the early RDBMS vendors started adding proprietary language extensions. Initially, these extensions were related to the SQL Data Definition Language (DDL) and SQL Data Manipulation Language (DML) [8] statements for leveraging storage capabilities, new data types, and optimized declarations and hints for faster data retrieval. Later on, the emphasis was put on developing more procedural language extensions. By nature, SQL is rather declarative than procedural and adding procedural logic inside of the queries provided more power to encapsulate the business logic closer to the data, minimize context switches between the SQL interpreter/compiler and the storage runtime, reducing network round trips to applications. These extensions included control-flow structures ((begin-end, if-then-else) and 3G-like languages statements (local variables, temporary tables, etc.). There were three major requirements that drove the evolution of the relational databases - the need for management of larger data volumes, performance improvements, and reduction in query response time.

Initially, the absence of SQL standards did not appear to be ambiguous. Each vendor was looking for more optimal means of data processing and no one was actively seeking standardization. Although it may sounds strange, some vendors today still considered this a winning strategy and many businesses are currently locked in proprietary functionality. Other RDBMS vendors (i.e. DB2 LUW) who strictly followed the standardization rules, have been perceived as lacking functionality only because for a long time they were staying close to the established SQL and procedural standards.

Neither the establishing of standards for stored procedures (ISO Persistent Stored modules (SQL/PSM) [10]), or the multiple efforts for SQL standardization (SQL-86, SQL-89, SQL-92/SQL2, and SQL-99/SQL3 [6]) stopped the RDBMS vendors from enhancing their own independent versions of DDL, DML and procedural extensions. And, although most vendors today support the bulk of the ISO SQL standards, there are many cross platform incompatibilities such as data types, procedural logic and control-of-flow statements, locking and transaction handling, recursive queries, triggers, XML and XQuery (SQL/XML [9]), and many others.

Figure 1 shows a diagram that symbolizes how the SQL standards are only a part of the functionality supported by the different RDBMS vendors.

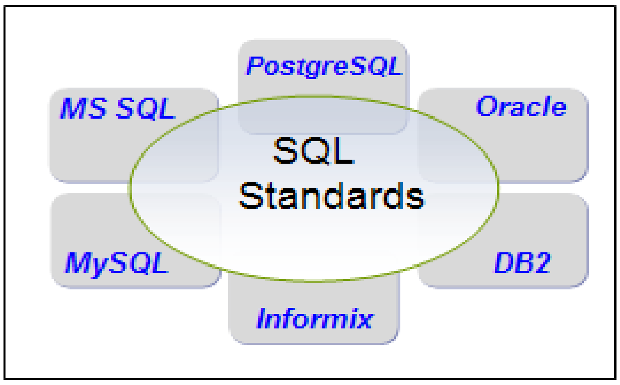


Figure 1. SQL Standards and Language Specifics

## *B. Traditional Migration Approach and SQL Compatibility*

In this article we use a simplified definition of the usually widely defined term "migration". A migration or conversion process includes all activities directed to move an application instance and its underlined database infrastructures to a new setup based on a new application and/or database support. The searched outcome is to obtain the same traversal and performance capabilities as the former application and database.

Traditionally, all migrations or conversions are manual or semi manual processes starting with an evaluation of the risk and amount of efforts for converting the application and database code, testing cycles needed, and redeployment on the new systems. In many cases the cost of such a migration is prohibitive.

SQL compatibility is a special feature of DB2 LUW that allows us to execute previously proprietary syntax constructions directly against a DB2 LUW database. The goal of these compatibility features is to simplify the migration processes making them straightforward, faster and less risky to the business. The compatibility level measures the number (or percent) of the syntax constructions that are recognized by DB2 LUW and could be executed immediately and with no changes onto DB2 LUW.

The relation between the migration processes and the level of compatibility of the migrated code is usually in an opposite proportion, i.e. the higher the compatibility level is- the lower the migration effort. In this context a high compatibility level means higher application and database portability, less manual work, faster migration time to value, simplified testing, etc. However, we have to note that in some rare cases a high compatibility does not necessary mean a lower migration cost or less work. For example, it is possible to have a compatibility level of 98% and still spend a few months on the remaining 2% - it all depends on the features included in the 2% bracket. This brings up the importance of accurate initial evaluation, which is examined in detail later.

A traditional migration approach normally includes several phases starting with the assessment of the application(s) and database portability, actual code migration for database and applications, testing, and production roll out. Each of them presents its own requirements and possible implications to the overall migration process. The SQL compatibility features of DB2 LUW and some new tooling developed to assist with the migration process, greatly simplify the traditional migration efforts on each of these phases. More details on these tools and the role of the compatibility are provided in each section below:

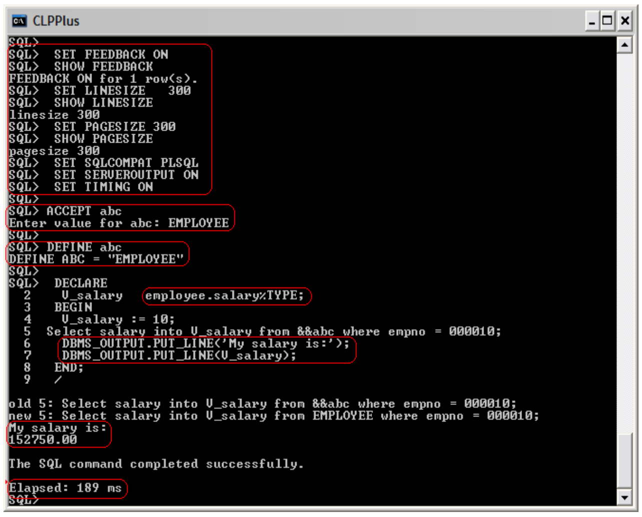
- Assessment phase – Every migration usually starts with evaluation of the application(s) and the source and target database systems. The main goal is to evaluate and determine the essential steps and tools necessary for successful migration of the existing application(s) and the underlined database management system to the new setup. The initial evaluation and the consequent planning for a migration should provide answers to important questions such as what is the portability level of application(s) and databases, when is the best time for the migration to occur, what are the internal and external dependencies (as part of the impact analysis), is there a potential risk for business users, what are the base requirements for the upgrade, etc. Many companies today offer comprehensive evaluation questionnaires and some automated or semi-automated tools to aid the evaluation process and to asses the risk for the business. Both DB2 LUW compatibility features utilize a special automated evaluation tool called MEET (Migration Enablement Evaluation Tool for DB2). MEET for DB2 LUW processes the syntax constructions and reports back not only the total compatibility percent, but also every unrecognized syntax structure. It will also suggest a possible work around for it. This includes evaluation of the Data Definition language (DDL) of the database objects, Data Manipulation Language (DML) constructs, as well as the procedural code. MEET for DB2 provides an easy/automated and reliable way to asses what percent of the objects will compile without changes and how much time will be needed to make the changes to the objects that have no direct support.

- Code migration phase – This is the next logical phase in the migration process that includes two stages - database migration and a migration of the application code. The conversion of the database usually precedes the application conversion for which a fully functioning database is required. Many tools on the market today offer migration automation for converting from one RDBMS to another. [12, 14] IBM Data Movement Tool (IDMT) is a development tool freely available for download from IBM [11]. IDMT uses a series of Java programs and shell scripts (both Windows and UNIX) to access the source of various databases and generate database scripts necessary to recreate the objects in DB2 LUW. It also extracts the data from the source database tables using a multi-threaded technique and generates scripts to populate the corresponding DB2 tables with the extracted data using the DB2 Load Utility. [11] The IDMT can also be used for the interactive deployment of PL/SQL or T-SQL objects such as triggers, functions, procedures and packages. The PL/SQL or T-SQL source code is extracted as part of the Extract DDL/Data operation when the DDL check box is selected. Once the Extract DDL operation is complete, the extracted DDL and PL/SQL or T-SQL could be deployed to DB2 LUW from the Interactive Deploy panel of the tool. Beginning with DB2 LUW V9.7, the IBM Migration Toolkit (MTK) [7] is no longer recommended for migrating applications from Oracle and Sybase to DB2 LUW.

  Another tool, IBM® Data Studio, provides to database developers and database administrators an integrated, eclipse-based environment for managing, administration and development in heterogeneous database environments. The tool also supports the DB2 LUW compatibility features. A great example is the "drag & drop" (or "copy & paste") functionality, which lets developers select an object such as a table, sequence, procedure, trigger, etc. from an Oracle database and drop it straight into a DB2 LUW database. The objects will be converted and created automatically in the database with DB2 supported syntax. If there are any exceptions, they will be displayed to the user. If a table contains data, the data could be also transferred automatically. IBM® Data Studio also provides a great development environment for creating, debugging, troubleshooting and performance tuning of procedural code, including the compatibility mode.

  The migration of the application code usually follows the database/environment migration. Not too many tools are available to help with the evaluation or the automated conversion of the applications code and usually this is a manual process. To facilitate limited to no application code changes, the Oracle compatibility feature of DB2 LUW provides enhanced drivers and libraries support. The Sybase Compatibility feature provides a way for the applications to send calls to the new DB2 databases, which will be passed through or converted to DB2 recognizable syntax if they are not compatible. This will allow to the Sybase applications to work against DB2 with limited changes or, in many cases, without recompilation at all.

- The testing phase will also benefit from the DB2 LUW compatibility features. A high compatibility would allow a reduced testing effort, since the existing test cases and scripts are expected to continue to work as before because there are no (or only limited) changes to the calls from the application, to the results set coming from the database, or to the output coming back to the application, etc. The Oracle compatibility feature in DB2 LUW provides a tool called CLPPlus - a specialized equivalent to Oracle's SQL*Plus. Analogically to the name, the CLPPlus could be used as a command line to access DB2 LUW databases with functionality and syntax equivalent to the Oracle tool. That guarantees all testing scripts commonly written for Oracle's SQL*Plus could be reused and executed straight against DB2 LUW.



```
SQL>
SQL>  SET FEEDBACK ON
SQL>  SHOW FEEDBACK
FEEDBACK ON for 1 row(s).
SQL>  SET LINESIZE   300
SQL>  SHOW LINESIZE
linesize 300
SQL>  SET PAGESIZE 300
SQL>  SHOW PAGESIZE
pagesize 300
SQL>  SET SQLCOMPAT PLSQL
SQL>  SET SERVEROUTPUT ON
SQL>  SET TIMING ON
SQL>
SQL> ACCEPT abc
Enter value for abc: EMPLOYEE
SQL>
SQL> DEFINE abc
DEFINE ABC = "EMPLOYEE"
SQL>
SQL>  DECLARE
  2    V_salary   employee.salary%TYPE;
  3   BEGIN
  4    V_salary := 10;
  5  Select salary into V_salary from &&abc where empno = 000010;
  6    DBMS_OUTPUT.PUT_LINE('My salary is:');
  7    DBMS_OUTPUT.PUT_LINE(V_salary);
  8   END;
  9   /

old 5: Select salary into V_salary from &&abc where empno = 000010;
new 5: Select salary into V_salary from EMPLOYEE where empno = 000010;
My salary is:
152750.00

The SQL command completed successfully.

Elapsed: 189 ms
SQL>
```

Figure 2 CLPPlus Window in DB2 LUW running SQL*Plus commands

With the Sybase compatibility feature, the result sets returned to the application will be translated to T-SQL code if they are not directly supported by DB2 engines, which guarantee compatible results.

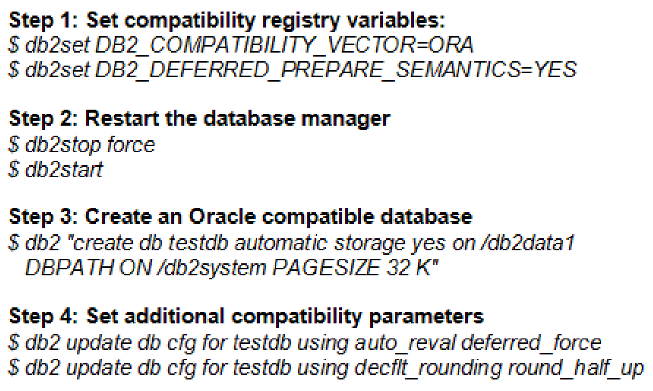
- Production roll-out phase: The reduction in efforts here is minimal as this is usually a standard process defined by the application and database specifics and it will have similar phases on any RDBMS. However, due to the tooling and automation provided for objects creation and data movement to DB2, this process is also simplified and more secure. Some of the tasks like tuning the new database and application before the cut off phase could be setup and completely left to the DB2 LUW autonomics. Knowing that only minimal changes have been made to the application and database structures also give a higher level of confidence in this phase.

## III. Oracle Compatibility Feature in DB2 LUW

DB2 LUW 9.7 includes extensive native support for the previously proprietary PL/SQL procedural language, new data types, scalar functions, improved concurrency, implicit casting and type resolution, built-in packages, OCI libraries, SQL*Plus functionality, and more. These features dramatically improve the ease of developing applications that run on both DB2 and Oracle, and simplify the process of moving applications from Oracle to DB2 LUW.

### *A. Setting up the Oracle comaptibility vector*

In order for us to utilize the benefits of the Oracle Compatibility feature, we need to set the compatibility flag to indicate which compiler option should be used. Fig. 3 shows a set of commands that allows a setup of the Oracle compatibility feature in a UNIX environment. As you may notice, there is an important sequence of the operations. We first have to setup the registry variable and the compatibility flag, secondly - restart the database engine, and, lastly - create the database. The two last commands set some additional compatibility parameters at the database level such as adjusting the rounding behavior to match that of Oracle or allow deploying objects out of dependency order by setting the revalidation semantics to "DEFERRED_FORCE".



**Step 1: Set compatibility registry variables:**
```
$ db2set DB2_COMPATIBILITY_VECTOR=ORA
$ db2set DB2_DEFERRED_PREPARE_SEMANTICS=YES
```

**Step 2: Restart the database manager**
```
$ db2stop force
$ db2start
```

**Step 3: Create an Oracle compatible database**
```
$ db2 "create db testdb automatic storage yes on /db2data1
   DBPATH ON /db2system PAGESIZE 32 K"
```

**Step 4: Set additional compatibility parameters**
```
$ db2 update db cfg for testdb using auto_reval deferred_force
$ db2 update db cfg for testdb using decflt_rounding round_half_up
```

### *B. Review of Oracle compatibility feature functionality*

The goal of the Oracle compatibility feature is to provide high compatibility eliminating the need for manual changes to the application code and database objects. However, this includes many different levels such as proprietary SQL language dialect and PL/SQL procedural language, built in packages, JDBC and OCI client functionality, scripts, and more. In this section we only highlight the most widely used new features provided by DB2 LUW version 9.7 and this is by no means a complete list.

One of the very first implications of the migration process is the differences in the data types supported by the different RDBMS. Oracle has many proprietary non-standard basic types in all categories - numerics, strings, and dates, all together with some complex types commonly used in Oracle's proprietary procedural language - PL/SQL. To address this, DB2 LUW 9.7 introduced new data types support such as NUMBER, VARCHAR2, NCHAR, NVARCHAR2, NCLOB, Oracle DATE format and TIMESTAMP(n), as well as support for BOOLEAN, VARRAY, ROW TYPE, Ref Cursor type, INDEX BY, etc. DB2 LUW and Oracle also exhibit different behavior when comparing, assigning and operating with different data types. A feature called implicit casting (also knows as weak typing) was added to DB2 9.7 to allow more flexible data type manipulation, including some more flexibility in the usage of the un-typed NULL. Extending the topic of data types manipulation come the new extended built-in functions, which DB2 LUW added to its own set of functions enfolding areas like conversion and casting functions (TO_DATE, TO_CHAR, TO_CLOB, TO_NUMBER, etc.), date calculation (ADD_MONTHS, MONTHS_BETWEEN, NEXT_DAY, etc.), string manipulation (LPAD/RPAD, INSTR, INITCAP, several proprietary extensions to SUBSTR, etc.), as well as some unique only to Oracle functions like NVL/NVL2, LEAST/GREATEST, DECODE, HEXTOROW, etc. Even with this limited list of examples, it is clear how rich the built-in functions library of DB2 LUW is.

Oracle SQL dialect is widely proprietary. DB2 LUW started to address some of these features in a earlier release (DB2 9.5 FP2). It started with just a few constructs and kept adding more - today we have support for important syntax like pseudo columns (ROWNUM, ROWID), DUAL table, (+) join syntax, recursion (CONNECT BY, ROWNUM), TRUNCATE table statement, MINUS SQL operator, etc. DB2 LUW 9.7 also adds more mobility in creating objects with "CREATE OR REPLACE" object statement, public synonyms, CREATE TEMPORARY TABLE, etc.

Compiling natively PL/SQL code is another very important area of the Oracle Compatibility feature. To guarantee native execution, the DB2 LUW engine now has its own PL/SQL compiler. This fact is important for several different reasons. No translation or emulation means native execution with great performance, similar to DB2 LUW monitoring and debugging, etc. It also allows the users to mix and match database objects created with different syntax and compiled by the different compilers. For example, a "DB2 style" SQL/PL procedure could call an "Oracle style" PL/SQL procedure and vice versa. It will also allow to the PL/SQL developers to continue to write PL/SQL code against DB2 LUW and apply their skills against a different RDBMS. The easiest way to sum the PL/SQL support is to look at the PL/SQL packages in DB2 LUW 9.7. We see the full syntax support from CREATE [OR REPLACE]

PACKAGE/PACKAGE BODY and the typical objects definitions (local/global variables and constants with %TYPE and %ROWTYPE anchoring, cursors, procedures and functions, types' declarations, etc.) to the exception handling and PRAGMA AUTONOMOUS, synonym creations, and others. Have to note also many special constructs of the PL/SQL syntax support not available in DB2 LUW before such as assignments (:=), WHILE loops and IF-THEN-ELSE logic, FORALL/BULK COLLECT, IN and OUT parameters on procedures and functions with parameter association and defaulting, and many others.

In addition to the custom created packages, Oracle provides some proprietary libraries, knows as built-in packages. DB2 LUW 9.7 supports some of the most widely used built-in packages like DBMS_OUTPUT, DBMS_SQL, UTL_FILE, UTL_MAIL,UTL_SMTP, DBMS_UTILITY, DBMS_ALERT, DBMS_PIPE, DBMS_JOB, DBMS_LOB, DBMS_DDL, etc.

Anonymous blocks are also supported and could be executed from an editor window or from the command line using either CLP or CLPPlus.

Prior to DB2 LUW version 9.7, there were considerable differences in the concurrency control between Oracle and DB2 LUW. Today, DB2 has a new concurrency control called Currently Committed that provides a concurrency behavior identical to that of Oracle. The name comes from the design model, in which DB2 actually reads the last, currently committed version of the changed row. Effective version 9.7, this is the default behavior of DB2 LUW for all DB2 databases as well.

Since general availability, with every new Fix Pack, DB2 LUW introduced more comprehensive compatibility features. For example, Fix Pack 5 introduced support for NVL2, SUBSTR2, and HEXTORAW functions, Pro*C, support for BOOLEAN in ROW and ARRAY types, nested ROW and ARRAY support for PL/SQL and JDBC support for them, and others. We are expecting the upcoming Fix Packs or releases of DB2 LUW to keep increasing the Oracle compatibility levels.

## IV. Sybase Compatibility Feature in DB2 LUW

The Sybase compatibility feature (often also abbreviated as IBM DB2 SSacSA) targets another proprietary SQL dialect - Transact SQL (or T-SQL), most of which syntax is also not supported by DB2 LUW and/or does not have an equivalent in the native for DB2 SQL/PL. However, the approach taken by DB2 development and its partner, ANTS Software INC. in implementing the Sybase Compatibility feature is quite different compared to the implementation of the Oracle compatibility. Instead of building a native compiler like in the Oracle compatibility feature, the Sybase Compatibility is implemented as a wrapper that will handle both external application calls and server side precompiled objects. To minimize the amount of manual rewrites on both the application and database side, IBM's Sybase compatibility feature utilizes the so called Compatibility Service, which runs on the database server. This service has a dual function. On one side, if the T-SQL syntax is recognized by DB2, the service will simply pass it to the database. The T-SQL syntax that is not immediately recognized by DB2 will be converted to an equivalent call that DB2 can understand. A built in intelligence ensures that the output sent back to the application is formatted the way the application expects it based on the T-SQL syntax that was issued originally.

### *A. Installing Sybase Comaptibility feature*

The installation of the Sybase compatibility feature is a little bit more complex compared to the setup of the Oracle compatibility feature. It requires creation of DB2 metadata repository database for storing compatibility/mapping information, installation of the server and client components, and configuration of the client drivers, server side (DB2 and Sybase environmental parameters), and IBM DB2 SSacSA Central utility. The configuration of the IBM DB2 SSacSA Central utility is needed in order to be able to move the Sybase metadata from a source Sybase database to the target IBM DB2 SSacSA metadata repository created at the first step of the installation process. After the three configuration steps are completed, the metadata could be loaded using IBM DB2 SSacSA Central.

### *B. Review of Sybase compatibility feature architecture*

The objective of the Sybase compatibility feature (IBM DB2 SSacSA) is to provide a seamless way of running the existing Sybase applications against DB2 LUW by transparently replacing the Sybase ASE components and the T-SQL language features that are not directly supported on DB2 LUW. The solution utilizes a compatibility service (IBM DB2 SSacSA) preserving the original application architecture as much as possible. This allows the actual application code to remain mostly, if not entirely, unchanged. Several architectural compatibility service components are needed to support this design as follow:

- IBM DB2 SSacSA metadata catalog - a catalog of metadata (mapping data) maintained by the IBM DB2 SSacSA and stored in DB2 LUW tables. It consists of information on how to map objects (schemas, tables, procedures, views, data types, column names, etc.) from the Sybase source to the DB2 LUW target and the associated with them Sybase security model (privileges, authorities, etc.). The metadata also includes information required to convert DB2 SQL error codes into equivalent Sybase codes expected by the application. Some Support Functions that are accessed by the application stored procedures during their execution are also stored in the metadata catalog as well as some information used to emulate Sybase system tables. This catalog is accessed by the IBM DB2 SSacSA translation routines during the process of translation.
  The IBM DB2 SSacSA metadata catalog is managed by a tool (graphical user interface) called IBM DB2 SSacSA Metadata Manager. An administrator could monitor which DB2 objects (stored procedures, functions, tables, views) are exposed as Sybase objects and could modify the existing mapping information

about database objects, data types, arguments of procedures and functions, etc. It could also manage the user logins emulated by IBM DB2 SSacSA.

- T-SQL Sybase translation routines – the role of the translation routines is to provide managed (compatibility) calls where there is no straight analog in the functionality between Sybase and DB2 LUW. For example, in some cases the T-SQL calls cannot be converted directly to specific DB2 functions. In others, the expected input or output to/from the call might be different than what DB2 provides. In most cases, we need a translation of the DB2 SQL return/error codes expected by the application. In all these cases the IBM DB2 SSacSA uses compatibility (managed) routines usually built as an external C++ DB2 UDF. This allows calls to the DB2 servers that closely emulate the original Sybase calls.
- Native and Managed Stored Procedures – Depending on the compatibility level, when moved to DB2, some Sybase stored procedures can be converted to either native stored procedures, or managed stored procedures.

  Managed Stored Procedures – the managed stored procedures usually require translation of the T-SQL code or of their input/output parameters to emulate the exact input/output that the application expects. When IBM DB2 SSacSA receives a CREATE PROCEDURE DDL statement, it creates a managed stored procedure automatically converting the Sybase T-SQL stored procedure into an IBM DB2 SSacSA DB2 managed procedure. After the translation is done, these stored procedures are called by IBM DB2 SSacSA in the same fashion as native procedures with the limitation that the managed stored procedures can be only used by the IBM DB2 SSacSA.

  Native stored procedures - the IBM DB2 SSacSA can be also configured to utilize in application calls native to DB2 LUW stored procedures created directly onto Db2 or migrated manually from Sybase (for example, with IBM Data Movement Tool).
- IBM DB2 SSacSA Client Driver - IBM DB2 SSacSA provides a replacement library for all major client drivers such as Open Client CT-Library, Open Client DB-Library, ODBC, JDBC, ADO.NET, etc and mimics the original APIs utilized by the application to communicate with the DB2 database.

  Figure 3 illustrates the main components of the IBM DB2 SSacSA architecture outlining the usage of compatibility managed (translated) and native to DB2 LUW objects. This is the unique approach in providing high application and database compatibility taken by IBM and their business partner ANTs Software, Inc.

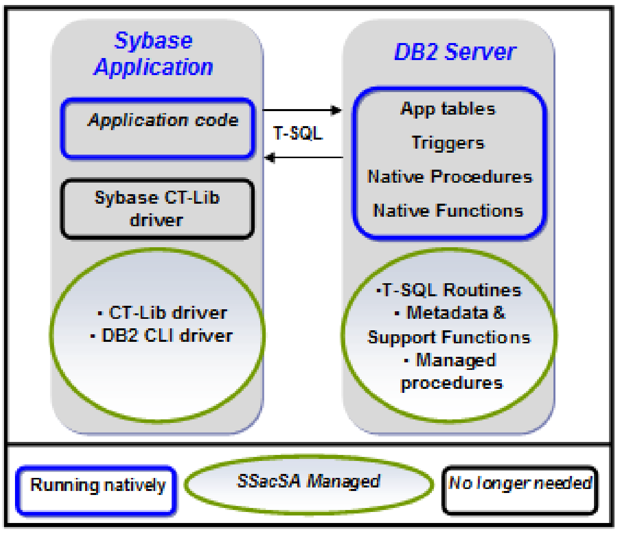


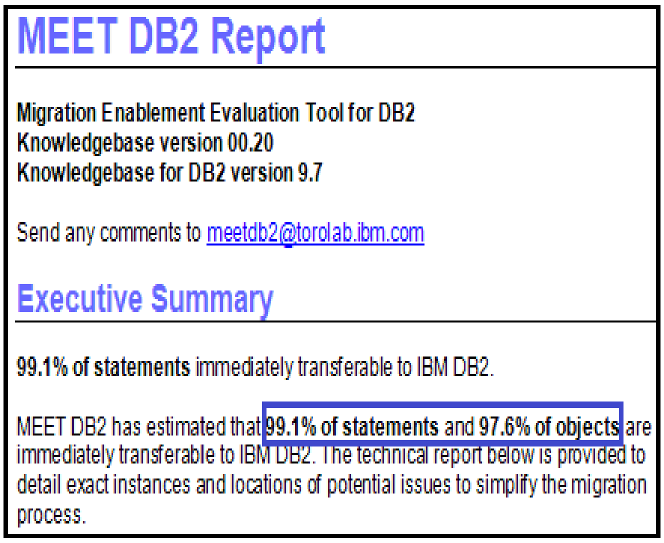
Figure 5. The main components of the IBM DB2 SSacSA architecture

## V. Migration Scenario

As outlined, DB2 LUW's approach for multi language support challenges the traditional concepts of application migration in several different directions: easily port custom applications to DB2 LUW, maintain single source across packaged application, re-use existing skills with not preference of the dialect, etc. However, where is the bigger bang for the buck coming from? Let's take a look at a sample migration scenario for moving a custom application named TUM from Oracle to DB2. TUM is a Java application written by the author of this article for his final project in Computer Science class. The application provides information tracking and reporting about employees, attendance, activities, and assignments of multiple departments and companies. The project included 12,500 lines of code and was performed solely by a person with no prior migration experience. The second author of the article provided only limited guidelines and supervision.

The first step of every migration process is to evaluate how much migration work needs to be done. This step includes evaluating a variety of information related to multiple levels, but mainly to the database side (DDL, DML, and procedural code) and application/middle tier layer (APIs, embedded SQL, etc.). Some additional information such as interactions with other systems, performance requirements, auditing, security, etc. are also important to be evaluated.

The evaluation of TUM was simple. The conversion assessment was completed in a few hours and showed that the application is highly compatible with DB2 LUW. According to the initial MEET report, 99.1% of the statements and 97.6% of the objects were immediately transferable to DB2 LUW v 9.7 Fix Pack 1.


MEET DB2 Report

Migration Enablement Evaluation Tool for DB2
Knowledgebase version 00.20
Knowledgebase for DB2 version 9.7

Send any comments to meetdb2@torolab.ibm.com

Executive Summary

99.1% of statements immediately transferable to IBM DB2.

MEET DB2 has estimated that 99.1% of statements and 97.6% of objects are immediately transferable to IBM DB2. The technical report below is provided to detail exact instances and locations of potential issues to simplify the migration process.

Figure 4. Initial MEET Report estimating the compatibility level against DB2 9.7 FP1

This compatibility percent was even higher when the code was evaluated against a later version of DB2 LUW - version 9.7 Fix Pack 4. These percents grew up to 99.8% compatibility for statements and 99.6% - for objects. The graph below shows how the percent of compatibility grows with each new fix pack of DB2 LUW, continually facilitating the migration process. (Note: The testing was started on DB2 9.7 FP1 and completed on FP4. However, the most recent version of DB2 LUW at the time of the publication is DB2 LUW v10.1). For the TUM application, we only had to change a few data type precisions, a few view definitions, and two pipelined functions.

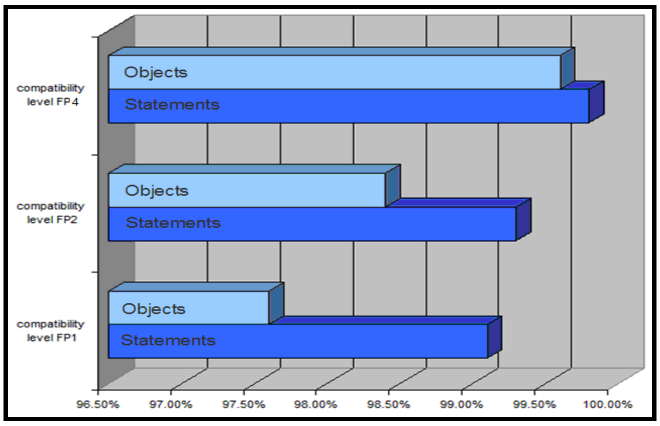


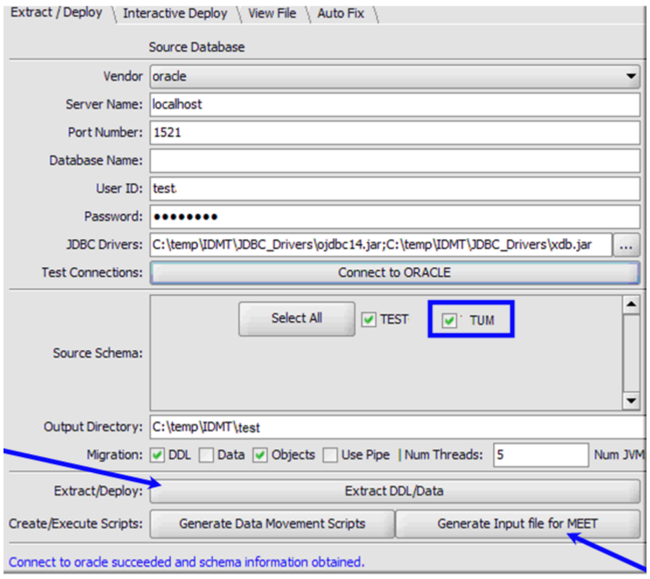
Figure 5. Increased compatibility levels with each Fix pack of DB2 9.7

On the applications side, the conversion questionnaire estimated changes to the connection properties of the applications as well as changes in the way the data was loaded because originally SQL*Loader was employed to load daily the data in the Oracle database. The testing proved that all embedded into the application SQL statements (standard CRUD) were executed without changes against the DB2 LUW engine.

The file with DDL, DML, and procedural code we evaluated with MEET was created with IDMT. The blue arrow on Figure 6 (right down corner) points to a button called "Create Input File for MEET". By clicking on this button after the Oracle connection was established, IDMT created a file extracting all objects from the original Oracle catalog.

IDMT was utilized also to move the code (DDL and PL/SQL) from the Oracle database to DB2 LUW with the exception of the two pipelined functions. It is interesting to note that in the conversion process IDMT automatically corrected the syntax of the views and the data types and precisions. No manual work was required.

Figure 6 presents the part of the screen of the IDMT for connecting to the Oracle database and schemas. Similar information was entered for connecting to DB2 in the second part of the screen (not shown). The migration selection included only the DDL and Objects check boxes. The data was not moved via IDMT as a specific setup and testing of DB2 load utility was in plans for the second part of the project. The extraction of the objects was run by clicking on the "Extract DDL/Data" button. Similarly, the import was invoked by selecting "Import DDL/Data" button located underneath the DB2 selection part of the screen.



Vendor: oracle
Server Name: localhost
Port Number: 1521
Database Name:
User ID: test
Password: ••••••••
JDBC Drivers: C:\temp\IDMT\JDBC_Drivers\ojdbc14.jar;C:\temp\IDMT\JDBC_Drivers\xdb.jar
Test Connections:
Source Schema: ✓ TEST ✓ TUM
Output Directory: C:\temp\IDMT\test
Migration: ✓ DDL ☐ Data ✓ Objects ☐ Use Pipe | Num Threads: 5 Num JVM
Extract/Deploy:
Create/Execute Scripts:
Connect to oracle succeeded and schema information obtained.

Figure 6. Left side of IDMT window showing the connection to the Oracle database and the migration selections made for the TUM project

Although IDMT has a very sophisticated Interactive Deploy window, IBM Data Studio was used to modify and compile the PL/SQL code, which did not compiled immediately with IDMT. The goal was to also explore the integration options between the IBM Data Studio and the possibilities it provides to collaborate with some popular development tools (i.e. Rational Application developer). Debugging of the procedural code was performed with educational and testing purposes from Data Studio as well.

As the conversion work was very limited, the testing cycle was short mainly concentrating on the modules where the changes occurred.

The last part of the conversion project was to load the data and setup the daily load routines. The SQL*Loader functionality was replaced by DB2 Load Utility. Although the initial expectations were that this process with take a few days, due to the similarities in the two tools, this process took only a few hours although it was performed by a person who had no prior experience with the DB2 Load utility.

Functional verification and performance testing concluded the testing cycle and the project on the next day. This part of the project took about 2 hours, all spent to guarantee an exact, on-to-one, comparison between the two running versions of the application – one against Oracle, and one – against DB2.

The general observations are that although relatively simple, the TUM application was specifically written for Oracle with proprietary data types/precision, SQL and PL/SQL statements and objects. The fact that the assessment and the conversion were executed for about one day and by a person with no prior migration experience is really remarkable proof for the high level of simplification provided by the SQL Compatibility Features of DB2 LUW.

## VI. Conclusion

This paper provides an overview of the SQL compatibility technologies provided by DB2 LUW v. 9.7. This functionality has been proven as key in win-backs and competitive migrations because it significantly reduces the time required for performing database migration from a disparate platform to DB2 LUW by affecting all stages of the conversion life cycle. Previously, the migration was a manual or semi-automated process, which required extensive efforts by professionals with multi-platform skills. By offering two different alternative solutions to the migration process (native support when moving from Oracle and emulation - from Sybase), DB2 LUW facilitates the enablement of applications written for other RDBMS vendors to DB2 by requiring far less resources than the traditional migration path including, but not limited to, avoiding source code modification and application rewrite, reducing testing complexity and time, offering phased deployment versus single big conversion deployment, and others. This new approach is called "enablement" [13] (rather than migration before) and it is not limited only to the database definitions and database procedural logic, but it also extends to the application layers, such as those written in Java or C++. Certainly, the SQL compatibility features could not provide 100 percent coverage for all proprietary SQL extensions, but there is support for the most often seen features. The effectiveness of the solution is currently limited to only two of the big RDBMS vendors – Oracle and Sybase. Philip Howard, Research Director of Bloor Research evaluates the compatibility features of DB2 LUW "as a significant step forward" and expects "to see more of this sort of capability introduced in the future." [1]

## Acknowledgment

The authors would like to acknowledge a number of colleagues and teachers who helped them build their knowledge and understanding of the RDBMS technology and the software products described here.

## Disclaimer

## Trademarks

## References

[1] ANTS Software. IBM® DB2® SQL Skin for applications compatible with Sybase ASE (IBM DB2 SSacSA). Retrieved December 19, 2011, from http://www.ants.com/ants.com/index.php/ssacsa.

[2] J. R. F. Boyce and D. D. Chamberlin. Using a structured English query language as a data definition facility. IBM Research Report, RJ1318, 1973.

[3] D. D. Chamberlin and R. F. Boyce. Sequel: A structured English query language. In FIDET '74: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control, pages 249{264, New York, NY, USA, 1974. ACM..

[4] E. F. Codd. A relational model of data for large shared data banks. Commun. ACM, 13(6):377{387, 1970.

[5] T. A. Corbi. Program understanding: Challenge for the 1990s. IBM System Journal, 28(2):294{306, 1989.

[6] A. Eisenberg and J. Melton. Sql: 1999, formerly known as sql3. SIGMOD Rec., 28(1):131{138, 1999.

[7] IBM. IBM Migration Toolkit user's guide and reference, 2008.

[8] ISO/IEC 9075-1:2008. Information technology {Database languages { SQL { Part 1: Framework (SQL/Framework). ISO, Geneva, Switzerland.

[9] ISO/IEC 9075-14:2008. Information technology {Database languages { SQL { Part 14: XML-Related Specifications (SQL/XML). ISO, Geneva, Switzerland.

[10] ISO/IEC 9075-4:2008. Information technology {Database languages { SQL { Part 4: Persistent Stored Modules (SQL/PSM). ISO, Geneva, Switzerland.

[11] IBM. Khatri, V. (Published 2009, June 19). IBM Data Movement Tool: Move data from source databases to DB2 in an easy way. Retrieved December 19, 2011, From http://www.ibm.com/developerworks/data/library/techarticle/dm-0906datamovement/.

[12] IBM. DB2® Information Center. Retrieved December 19, 2011, from http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp.

[13] IBM. Redbooks, Oracle to DB2 Conversion Guide. City: Vervante, 2009.

[14] Microsoft. Microsoft SQL server migration assistant for oracle, facilitating database migration, 2005.