

## SQL Compatibility – What's New?

Mike Springgay

IBM Db2 Warehouse Common Engine Architect

[springga@ca.ibm.com](mailto:springga@ca.ibm.com)

## Please note :

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

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 many factors, including 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 results similar to those stated here.

# Notices and disclaimers

•© 2021 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

•U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

•Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

•IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.”

•Any statements regarding IBM’s future direction, intent or product plans are subject to change or withdrawal without notice.

•Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

•References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

•Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

•It is the customer’s responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer’s business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

## Agenda

- What is meant by SQL Compatibility?
- Database Conversion Workbench
- Oracle Compatibility
- Netezza Compatibility
- Db2 Family Compatibility

## Multiple SQL dialects

**Db2: Polyglot SQL Capabilities**

- Breaks the confusion by providing:
  - SQL and stored proc. language compatibility with many variants
  - Db2
  - Oracle
  - Netezza
  - PostgreSQL

Traditionally Db2 has focused on the SQL standard adding its own set of extensions to meet customer demands. Of course the other major vendors also provide extensions and not always in compatible ways. And opensource databases have not always embraced or necessarily followed the standard at all – they are getting better!

Db2 starting in 9.7 choose to embrace this polyglot world by bringing support for the commonly used syntax and features of other dialects.

Allowing for greater and easier portability and better skills reuse.

Breaking the lock-in to a single dialect or the headache of dual/triple maintenance required in supporting different vendors

Initial compatibility focused on Oracle but recently a number of extensions added to better align with PostgreSQL and Netezza.

## SQL Compatibility: What does it mean?

- Syntax toleration
  - Where no semantic conflicts exist
- New features
  - Close gap between Db2 and other vendors
- Compatibility mode
  - Overcome differences between Db2 and other SQL dialects
- Application Enablement Extensions

But what does compatibility really mean?

Direct syntax toleration when no semantic conflicts. If conflicts provide support under a compatibility switch

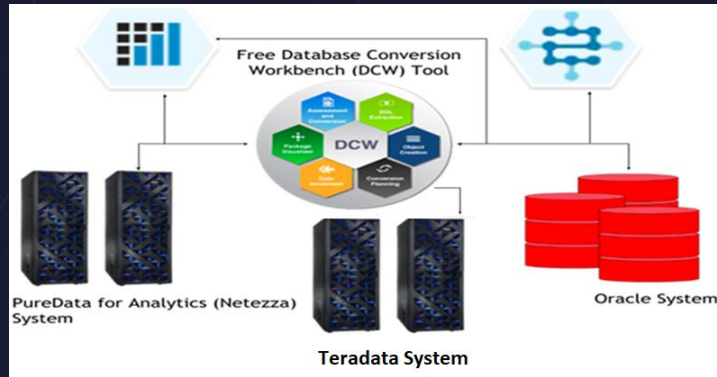
Missing features added to provide similar capabilities

Compatibility modes to allow enabling things that conflict with Db2 native behavior

Client side Application Enablement extensions eg. OCI or Pro-C support.  
And Command lines CLPPlus or dbsql .

# The place to start is with Database Conversion Workbench

IBM Db2 / IBM Db2 Warehouse



Database conversion workbench is the place to start on any conversion project.

Supports evaluating Oracle to Db2 or Db2 Warehouse targets And Netezza and Teradata to Db2 Warehouse

## What is Database Conversion Workbench

- Is an offline tool – No catalog / schema context!
  - Evaluates each statement for syntax compatibility only
  - Does NOT move data – use appropriate data movement tooling for project
- Evaluation of DDL and DML compatibility
  - Provide a level of confidence around conversion
- Translation of DDL
  - Always translate DDL using tooling


The IBM Database Conversion Workbench is a tool for evaluating DDL and DML statements compatibility.

The reports provide a level of confidence around the level of compatibility expected for a specific database or application.

It also provides automation of conversion of incompatibilities or guidance on what changes may need to be made when automatic conversion not straight forward.

Generally most gaps in DDL can be automatically converted and in general DCW is relied on for making them compatible.





**IBM Database Conversion Workbench**

- Release 5.0 brings:
  - A new React and Carbon based user interface
  - Consistent feel with other IBM tooling
  - No more Eclipse!
  - db\_harmony\_profiler now dcw\_lite and command line only

Database Conversion Workbench continues to evolve since original introduction as MEET

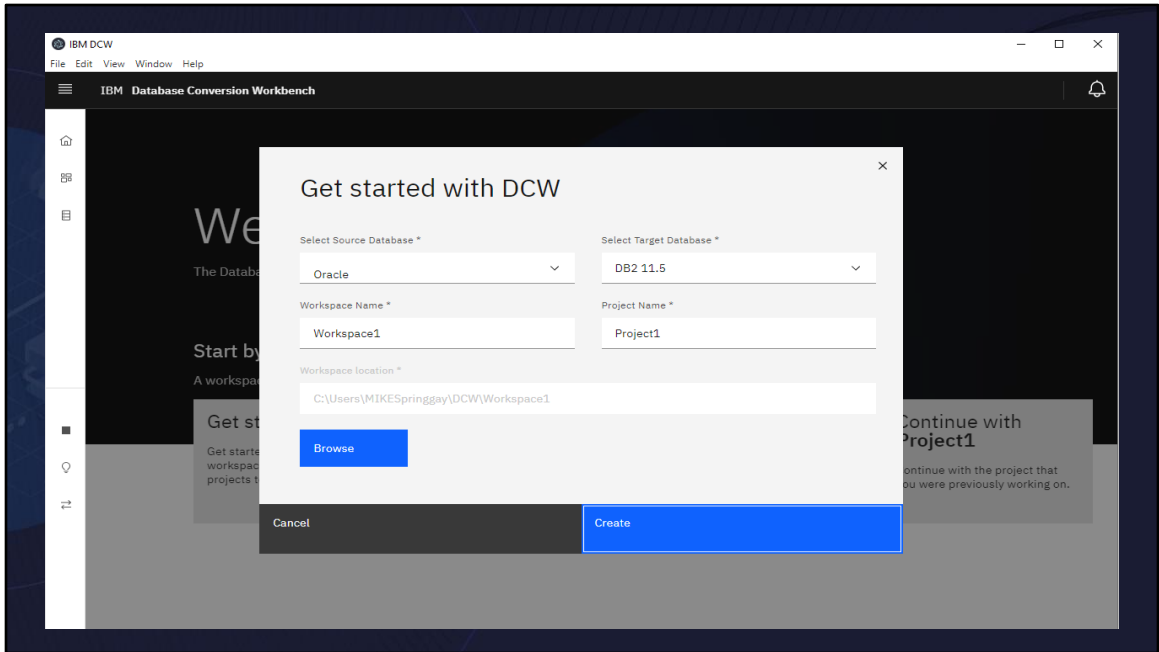
Actually comes in two different packages now

Database Harmony Profiler a very light weight GUI that is great for quick up front evaluations and simple conversions

And the more traditional eclipsed based package for when a full blown IDE required to complete the conversion project

The eclipsed base package is no longer tied to Data Studio. It is now delivered as a self contained eclipse application.

Decoupling provided a simpler installation experience and has led to being able to deliver that can run on an Apple Macintosh



IBM DCW  
File Edit View Window Help

IBM Database Conversion Workbench

Home / Workspaces / Workspace1 / Project1 /

# Project1

Last Modified: Today

1 Assets    Oracle Source    DB2 11.5 Target

Assets    Overall Report    Settings    Configuration

## SQL Assets

Filter table Upload Files

<input type="checkbox"/>	Name	Last Modified	Evaluation Report	Converted SQL
<input type="checkbox"/>	SQL <a href="#">tpipe.sql</a>	Aug 31, 2020 at 06:59 AM		

IBM DCW  
File Edit View Window Help  
IBM Database Conversion Workbench

Home / Workspaces / Workspace1 / Project1 / tpipe.sql /

**Export report** → [Export](#)

**Summary** ← **Overall score top left corner**

**100.0%**

0 out of 34 PL/SQL statements are not compatible  
0 of 0 SQL statements are compatible  
62 out of 62 (100.0%) lines of code analyzed

**Summary** | PL/SQL Evaluation | **SQL Evaluation** | Statistics  
← **Tabs above stats**

PL/SQL  SQL  Effort

Object Type	Total Count	Fixed by DC	Requires A	% Requires
Anonymous Blocks	0	0	0	0.0%
Package Heads	1	1	0	0.0%
Package Body	1	1	0	0.0%

**Code display** →

```
1 create or replace package tpipe is
2
3   type type_char_err is table of varchar2(2000) index by binary_integer;
4
5   type t_vrednost is record(
6     vrednost varchar2(255));
7   type t_vrednosti_err is table of t_vrednost;
8
9   function parse_vrt(
10    o_strnam in varchar2)
11    return t_vrednosti_err pipelined;
12
13 end tpipe;
14 /
15
```

Evaluate Convert

## Evaluation best practices

- The cleaner the input the more accurate and informative the reports:
  - Avoid processing Users, Grants, etc.
    - These will be different regardless but difference has minimal impact to “applications”
  - Only extract / process non-Obfuscated procedures
    - Syntax of Obfuscated procedures can not be evaluated as not present
    - And if procedure is Obfuscated it can't be replayed any way
  - Try to avoid system / tooling generated SQL
    - Example: For Netezza avoid extracting INZA and SQL Toolkit
      - Both of these are provide as built-in routines.

Being a tool the result are highly dependent on input provided.

Make sure the scripts are clean of non-SQL constructs that may trip up the syntax parser

Avoid evaluating items that are not directly related to application portability or in the areas Db2 has focused on for compatabilitiy

Examples are User and Grants can easily dominate a file hiding the true SQL compatibility

Don't extract Obfuscated routines in the source database . These can not be evaluated as the text is not present. Also these can't be replayed either!

Try to avoid capturing SQL that may be generated by the system itself or from tooling that is Database Aware

On Netezza avoid extracting the SQL Toolkit and INZA both are provide as built-in's and the real interest is in which functions are referenced in SQL not evaluated the provide functions

## Database Conversion Work bench on HDM Community

- Always use the latest available
  - <https://community.ibm.com/community/user/hybriddatamanagement/blogs/jordan-hodges1/2021/04/07/dcw-product-hub>
- Watch the tutorial videos
  - <https://community.ibm.com/community/user/hybriddatamanagement/blogs/jordan-hodges1/2020/01/28/dcw-tutorials>

And always use the latest available. Database conversion workbench is on a continuous delivery model releasing monthly

Now located in the HDM Community!

## Db2's Oracle compatibility - Recap

Features	
SQL Dialect	Date Types, Functions, SQL syntax
Scalar Functions	Oracle unique scalar functions
PL/SQL	Native support for most PL/SQL syntax
PL/SQL Packages	Named collection of functions, procedures, types and variables
Built-in Package Library	Common PL/SQL packages provided by default Others available as add-ons
Application Enablement	IBM Data Server Driver extensions
And more ...	

Db2's Oracle Compatibility mode provides greater support with Oracle syntax and behavior

Oracle compatibility is enabled by the compatibility vector but is also hardened at the database during create database.

This is because a number of Oracle features like the treatment of empty strings are fairly divergent from Db2's and lead to different on disk storage making co-existence of both modes of operation incompatible

Data Types :

Varchar2 – including empty string is NULL semantics and differing blank comparison behavior depending on types involved

Number - DECFLOAT simulated

DATE - TIMESTAMP(0) to ensure it always has a time component. But means not true "dates" in the system

Many commonly seen Oracle scalar functions are provided

PL/SQL – Native support for a large degree of Oracle PL/SQL procedural language

Built-in Packages - A large set of the Oracle packages that assist with application development exists in Db2 and more are added based on popularity.

Application enablement extensions in the form of CLPPlus compatible command line, OCI library, Java extensions for Oracle specific types and embedded SQL extensions in support of Pro\*C



## New DBMS Packages (Oracle Package Support)

- UTL\_RAW package provides subprograms for manipulation of binary (“raw”) data.
- DBMS\_APPLICATION\_INFO package
  - Used to add information to the session and SYSTOOLS.SESSION\_LONGOPS, to allow self identifying session and to track the execution time of long-running batch jobs.
- DBMS\_STATS package supports viewing and modify optimizer statistics collected
- DBMS\_LOCK package makes lock management services available to PL/SQL developers
- DBMS\_RANDOM package provides the capability to produce random numbers.

A number of new DBMS modules (aka packages) will be provided as built-ins over course of v11.5. These are provided for compatibility with Oracle.

Historically these have been provided as add-ons as part of the Database Conversion Workbench migration support

UTL\_RAW is a module that simplifies manipulating binary data.

DBMS\_APPLICATION\_INFO provides another means of accessing or setting the SET CLIENT INFO data used to tag application/user identification

DBMS\_STATS – Another mechanism to interact with statistics collection in Db2

DBMS\_LOCK - Ability to do ones own lock serialization within a SQL PL context.

DBMS\_RANDOM – Ability to generate random numbers

## DB2 Add-on Modules for Oracle Compatibility

- Following modules included in Version 2.0 and not yet built-in:
  - UTL\_URL
  - SQL PL Web Toolkit
    - HTP, HTF and MOD\_DB2 Apache module
  - UTL\_INADDR
  - UTL\_TCP
  - OWA\_UTIL

DB2 Add-on modules for Oracle Database Compatibility (or “DB2 Add-on Modules” for short) can facilitate migration of Oracle applications to DB2 for LUW by providing a set of SQL PL modules that have similar functionality to some of Oracle's PL/SQL built-in packages. Download DB2 Add-on Modules

Still required for a few of the non-core functions. UTL\_URL is one lined up to be tackled next.

Are there any in list frequently used ?

Db2 Oracle Compatibility 11.5.1 through 11.5.6

## WAIT <seconds> | NOWAIT

- Allows specifying lock timeout at the statement level
  - Overrides lock timeout special register
  - NOWAIT - Do not wait on any lock encountered immediately error out
  - WAIT <seconds> - Wait number of seconds on locks encountered
- Both return standard lock timeout error SQLCODE -911 rc 68

Introduce support for the SKIP LOCK clause.

Skip lock is provides increased concurrency on searched update / delete statements

Allow for implementing queries that access a table in queue form

## SKIP LOCKED (1 | 2)

- Allows a QUERY to silently skip rows locked by other transactions
- A qualifying row
  - Is NOT locked or LOCKED in non-conflicting mode then **the row is returned**
  - Is locked in conflicting mode then **the row is skipped**
  - Is locked in conflicting mode. However lock avoidance techniques, such as currently committed, are successfully applied to avoid the lock then **a row is returned**

Introduce support for the SKIP LOCK clause.

Skip lock is provides increased concurrency on searched update / delete statements

Allow for implementing queries that access a table in queue form

## SKIP LOCK (2|2)

- Can provide increased concurrency for certain application types
- Typical Usage
  - Message queuing where applications expect to skip over records which are in flight by other active transactions
  - Used most often in conjunction with “FOR UPDATE” or in a searched UPDATE / DELETE statement

## Scalar Functions (1 | 4)

- NCHR
  - Produce a UTF-8 character given a UTF-32 codepoint
- Example:
  - NCHR(65313) → 'A' (Full width A)
  - HEX(NCHR(65313)) → EFBCA1
  - NCHR(65) → 'A' (Half width A)
  - HEX(HCHR(65)) → 41

NCHR is a scalar function that produces a UTF-8 Character given a UTF-32 codepoint (integer value)

## Scalar Functions (2 | 4)

- **TO\_MULTI\_BYTE**
  - Convert any character in the input string to its multi-byte equivalent
- **TO\_SINGLE\_BYTE**
  - Convert any character in the input string to its single-byte equivalent
  - **Now supported in any database codepage**
- **Example (UTF-8):**
  - `HEX('A') → 41`
  - `HEX(TO_MULTI_BYTE('A')) → EFBCA1`
  - `HEX(TO_SINGLE_BYTE(NCHR(65313))) → 41`

TO\_MULTI\_BYTE function provides the ability to normalize a string into all full width characters

TO\_SINGLE\_BYTE can be used to normalize characters into the more typical single byte representation



## Scalar Functions (3 | 4)

- **ASCIISTR**
  - Returns an ASCII version of the string. Any non-ASCII characters are escaped as “\xxxx or \xxxx\yyyy” where xxxx and yyyy are UTF-16 codepoints
- **Example:**
  - `ASCIISTR('ABÄCDE')` → `'AB\00C4CDE'`

ASCIISTR scalar function provides the ability to take a character string which includes national language (ie. non-ASCII) characters and produce an ASCII representation of that string through escaping into UTF-16 codepoints any non-ASCII characters

## Scalar Functions (4 | 4)

- **UNISTR**
  - Returns the Unicode string in UTF-8
  - Processes an ASCII string converting “\xxxx or \xxxx\yyyy” UTF-16 codepoints into appropriate UTF-8 or UTF-16 characters.
- **Example:**
  - `UNISTR('abc\00e5\00f1\00f6')` → `'abcåñö'`

UNISTR produces the national language string given a pure ASCII string with non-ASCII characters escaped as UTF-16 codepoints .ie undoes what ACSISTR produces.

## Db2's Netezza compatibility - Recap

Features	
Data Types	Boolean, Interval (sort of), Synonym names
SQL Dialect	CTAS, JOIN USING, UPDATE FROM, :: Cast, etc.
Scalar Functions	Many SQL Toolkit built-in, Common PostgreSQL
Operators	% modulo, Bitwise ( , &, !, #), ^ exponential
NZPLSQL	Native support for most NZPLSQL syntax
External Tables	Netezza Style External Tables for LOAD / UNLOAD
And more ...	

Db2 has been extended in many ways to support Netezza compatibility and indirectly PostgreSQL compatibility

Data types have been added such as Boolean type, handling of INTERVAL literals and casts, synonyms added so alternate names can be used for existing types (eg. INT8 instead of BIGINT)

SQL syntax has been enhanced to provide for improved Create Table As , alternate JOIN and UPDATE syntax.

Many new scalar functions were added to support either common PostgreSQL functions like DATE\_PART or to provide native support for the most common SQL Toolkit functions like HASH

Modulo, Bitwise and exponential Operators added

NZPLSQL syntax parser added to support native NZPLSQL procedures. As well as extension to support as functions

External Tables added

## Netezza Compatibility

- `SQL_COMPAT='NPS'`
  - Session level variable introduced to control Db2 vs Netezza semantic behavior
  - Can be toggled ON/OFF at session level as required
  - JDBC “specialRegisters” option or C-Client `db2dsdriver.cfg` “sessionglobalvariables” can be used to configure for a connection
- `CONNECT_PROC`
  - An option as well – but be careful not to impact all connections as console, system connections expect Db2 semantics

27

## Netezza Compatibility

Netezza compatibility is available by toggling on / off at the session level. Unlike our Oracle compatibility there is no database wide switch which means both Db2 and Netezza SQL expectations can co-exist in the same database.

One can though influence how the session defaults by explicitly configure the clients to specify `SQL_COMPAT='NPS'`  
Or by creating a `CONNECT_PROC` to set `SQL_COMPAT='NPS'`

Be careful though since many packaged applications will expect Db2 syntax as do many Db2 toolsng (console / DSM, etc.)

Netezza Compatibility: 11.5.1 through 11.5.6

## >1012 table columns

- Increase maximum number of columns allowed to 2048 from 1012
  - Supported in 32K page sizes
  - For row tables columns widths must fit page size as before
  - Overall row width still subject to the 1M extended row size
- Actually exceeds Netezza 1600 limit but aligns with Teradata

29

A number of select list improvements also delivered.

You can now use \* plus additional columns or even \*,\* if you really wan too 😊

You can reference aliases defined in the select list in other context.

Within the select list itself, group by, having and where clauses.

WHERE clause support is being investigated

Of course you need to specify the SQL\_COMPAT='NPS' to enable syntax as "alias" resolution can interfere with Db2s correlation name resolution.

## SELECT LIST improvements

- **SELECT \*, C1+C2, C2+5 FROM T1**
  - Can select all columns "\*" and augment select list with additional columns
- **SELECT LIST ALIAS** eg. **SELECT C1+C2 AS C3 FROM T1**
  - Can be leveraged for more than just naming the output column
  - Reference an expressions later in SELECT list
    - **SELECT C1+C1 AS C3, C3 + 5 AS C4 ..... WHERE C3 = C4**
  - Reference in ORDER BY, GROUP BY, HAVING or WHERE clauses
  - Requires `SQL_COMPAT='NPS'` be set on the session

A number of select list improvements also delivered.

You can now use \* plus additional columns or even \*,\* if you really want too 😊

You can reference aliases defined in the select list in other context.

Within the select list itself, group by, having and where clauses.

WHERE clause support is being investigated

Of course you need to specify the `SQL_COMPAT='NPS'` to enable syntax as "alias" resolution can interfere with Db2s correlation name resolution.

## Nested “WITH” (1 | 2)

WITH now included in “common-table-expression”

```
AS (--+ + --- fullselect --- )  
'---WITH - common-table-expression ---'
```

```
WITH T1(c1,c2) AS  
(  
    WITH T2(c3,c4) AS  
        (SELECT c5,c6 FROM T3)  
        SELECT c3,c4 FROM T2  
) SELECT c1,c2 FROM T1;
```

Common Table expression support extended to allow WITH in most places a sub-select allowed. Exceptions in predicate evaluation (ie. within WHERE)

You can now have WITH under a WITH statement or as the source of an INSERT statement.



## Nested "WITH" (2|2)

```
INSERT INTO t1
  WITH tw1(c5,c6) as
  (
    WITH tw2(c7,c8) as (SELECT * FROM t2)
    SELECT *FROM tw2
  ) SELECT * FROM tw1
```

## TRUNCATE ~~IMMEDIATE~~

- IMMEDIATE now optional on columnar tables
- No IMMEDIATE means:
  - Does not need to be the first statement
  - Can be aborted – **TRANSACTIONAL SCOPED**
- Similar to mass delete operation ( DELETE \* )

Transactional truncate support for columnar tables. No longer is the IMMEDIATE clause mandatory.

Transaction truncate statements can be any place within a transaction. Will be rolled back if the transaction aborts

Operate much like a mass delete of the table

## Create Table AS - Enhanced (1 | 3)

- Generated or Duplicate column names not valid in CTAS
- Db2 generates column names of form C# where # is 1...N
- Netezza generated name "?COLUMN?" - for complex expressions
- Netezza however will use function name as column name
- Db2 and Netezza both don't like duplicate names
- Netezza will allow generated but only 1

## Create Table AS - Enhanced (2 | 3)

```
CREATE TABLE CNTTBL AS  
SELECT COUNT(*), C1 FROM T1
```

Under SQL\_COMPAT='NPS' will create table:

CNTTBL	COUNT	C1
--------	-------	----

Additional enhancement to Create Table AS (CTAS) support under SQL\_COMPAT='NPS'

Now the function name will be used to generate column name when no explicit "AS <FUNCTION NAME>" exists.

## Create Table AS - Enhanced (2 | 3)

```
CREATE TABLE CNTTBL AS  
SELECT COUNT(*) + 10, C1 FROM T1
```

Under SQL\_COMPAT='NPS' will still fail with SQL0153N  
Would succeed in Netezza but unlikely to be seen commonly

Db2 will still not allow a CTAS statement to proceed if an ambiguous generated name is found in the statement. Netezza would allow this to proceed as long as only one exists.

Assumption is this is a rare case if seen at all. Expectation is that complex expressions will include a "AS <name>" clause to explicit name the column even in Netezza.

## Db2 Netezza User Defined Extensions (UDX) (1 | 3)

- UDX is Netezza term for UDF
- Language C++
  - Inherit from class defined in **#include "udxinc.h"**
  - All implement "instantiate" function
  - Scalar Function implement a "evaluate" function
  - Table Function implement "newInputRow" and "nextOutputRow" functions
  - Aggregate Function implement a "initializeState", "accumulate", "merge" and "finalResult" functions

Db2 has been extended to support Netezza User Defined function interface – previously available only in Db2 Warehouse.

The Netezza UDX interface is a C++ class based programming model. One inherits from the appropriate class in "udxinc.h" for the function type being implemented.

Mandatory class methods must be implemented for the Db2 engine to invoke at runtime.

Not currently available on windows.

## Db2 Netezza User Defined Extensions (UDX) (2 | 3)

- Analytic Executables (AE) Interface
  - AE is a concept used in Netezza to support fenced functions
  - Used extensively to support Languages other than C++
  - Db2 provides support for the AE interface
    - A push/pull mechanism for data exchange

Analytics Executable is a Netezza frame work and set of libraries for implementing fenced routines. Also leveraged in Netezza for providing language support beyond C++.

Db2 however provides the interface for supporting AE written C++ and Python User Defined functions.

## Db2 Netezza User Defined Extensions (UDX) (3 | 3)

- Supports Shaper and Sizer functions
- A shaper dynamically defines the returned row description
  - Used when a NPSGENERIC table function is defined as RETURN TABLE(ANY)
- A sizer dynamically defines the return length of a scalar
  - Used to provide sizing for RETURN VARCHAR(ANY) / CHAR(ANY)

An unique feature of the Netezza C++ interface is the ability to provide shaper and sizer functions that the SQL compiler can invoke at compilation time to dynamically determine lengths or table shape based on known inputs at compilation time.



## Python User Defined Functions

- Python 3 language support
- Supported User Functions:
  - Scalars
  - Table Functions
  - Aggregate Functions
- PYTHON\_PATH DBM CFG to specific location of python installation
- Leverages Netezza AE Interface for data exchange

Python User Defined functions now supported in Db2 leveraging the python 3 routine and language support

All variants of functions (scalars, table, aggregates) supported

Db2 does not provide / ship a python runtime. One must be installed a compatible runtime and update the DBM CFG PYTHON\_PATH to indicated were the runtime is located.

Leverage the Netezza AE pyton library to provide the data exchange between functions and Db2.

## Python UDF Example (1 | 2)

```
import nzae class
multiply(nzae.Ae):
    def _getFunctionResult(self, rows):
        x, y = rows
        if x is None:
            self.userError("first input column may not be null")
        return x * y
multiply.run()
```

Source example for a simple python scalar function. Showing how it leverages the “nzae” class.

## Python UDF Example (2 | 2)

```
CREATE FUNCTION multiply_scalar(integer, integer) returns integer
LANGUAGE PYTHON
PARAMETER STYLE NPSGENERIC
FENCED NOT THREADSAFE
NO FINAL CALL ALLOW PARALLEL NO DBINFO DETERMINISTIC
NO EXTERNAL ACTION RETURNS NULL ON NULL INPUT NO SQL
EXTERNAL NAME 'path_to_multiply.py'
```

Example of registering a python scalar function with unique or required options in red.

## R User Defined Functions

- R language support
- Supported User Functions:
  - Scalars
  - Table Functions
  - Aggregate Functions
- R\_PATH DBM CFG to specific location of R runtime installation
- Leverages Netezza AE Interface for data exchange

Python User Defined functions now supported in Db2 leveraging the python 3 routine and language support

All variants of functions (scalars, table, aggregates) supported

Db2 does not provide / ship a python runtime. One must be installed a compatible runtime and update the DBM CFG PYTHON\_PATH to indicated were the runtime is located.

Leverage the Netezza AE pyton library to provide the data exchange between functions and Db2.

## In-Database Analytics

- Stored procedures provide predictive analytics algorithms
- Clustering, Classification, Decision trees and other algorithms provided natively within the database
- Fast In-DB analytics when compared to executing these algorithms in applications
- Integrated with database to provide model management and security
- K-means clustering
- Naive Bayes
- Association rules
- Sequential patterns
- Linear regression
- Decision trees
- Regression trees
- KNN

Db2 now also provides access to many of the built-in in-database analytics procedures previously available only in Db2 Warehouse.

Some algorithms are only available on Linux x86 and PPCLE. With porting to other platforms being investigated.

## Scalar Functions

- **UNICHR (Alias NCHR)**
  - Unicode Character function produce a UTF-8 character given a UTF-32 codepoint
- **<FUNC>\_BETWEEN**
  - (DAYS\_BETWEEN, WEEK\_BETWEEN, MONTHS\_BETWEEN, HOURS\_BETWEEN, MINUTES\_BETWEEN, SECONDS\_BETWEEN)
  - Under SQL\_COMPAT='NPS' now produce absolute values

Scalar function improvements. The UNICHR function has been delivered which is just another name for NCHR discussed earlier.

The "BETWEEN" functions have been updated to produce the absolute value under "NPS" compat.

## Db2 SQL Toolkit Extension (1 | 2)

Array		XML	Regular Expression
ARRAY	GET_VALUE_DATE	ISVALIDXML	REGEXP_EXTRACT_ALL
ADD_ELEMENT	GET_VALUE_DOUBLE	ISXML	REGEXP_EXTRACT_ALL_SP
ARRAY_COMBINE	GET_VALUE_INT	XMLATTRIBUTES	REGEXP_EXTRACT_SP
ARRAY_CONCAT	GET_VALUE_TIME	XMLCONCAT	REGEXP_VERSION
ARRAY_COUNT	GET_VALUE_TIMESTAMP	XMLELEMENT	
ARRAY_SPLIT	GET_VALUE_TIMETZ	XML EXISTS NODE	
ARRAY_TYPE	GET_VALUE_VARCHAR	XMLEXTRACT	
ELEMENT_TYPE	COLLECTION	XMLEXTRACT	
ELEMENT_NAME		XMLPARSE	
REPLACE_ELEMENT		XMLROOT	
DELETE_ELEMENT		XMLSERIALIZE	
		XMLUPDATE	

The less common or duplicate SQL Toolkit functions have been ported to Linux x86 and ppcle to simplify migrations and avoid needing to convert to other Db2 native mechanisms that may not be strictly compatible.

Download link:

<https://community.ibm.com/community/user/hybriddatamanagement/viewdocument/compatibility-and-migration-tools-s-1?CommunityKey=b5b60ace-5eb2-4444-a961-d9edb6f39bd8>

## Db2 SQL Toolkit Extension (2 | 2)

Text Analytics	Data Transformation	Hash
WORD_DIFF WORD_FIND WORD_KEY WORD_KEY_DIFF WORD_KEY_TOCHAR WORD_STEM	COMRPRESS DECOMPRESS DECRYPT ENCRYPT FPE_DECRYPT FPE_ENCRYPT UUENCODE UUDECODE CRYPTO_VERSION	HASH_NVARCHAR MT_RANDOM

<https://community.ibm.com/community/user/hybridatamanagement/viewdocument/db2-sql-extension-toolkit-release?CommunityKey=71ceaea3-db2c-451d-87d1-51f254454c6a&tab=librarydocuments>

Download link:

<https://community.ibm.com/community/user/hybridatamanagement/viewdocument/compatibility-and-migration-tools-s-1?CommunityKey=b5b60ace-5eb2-4444-a961-d9edb6f39bd8>



## Db2 Family Compatibility (1 | 3)

- **Large Objects in Columnar engine (CLOB, BLOB, DBCLOB, NCLOB)**
  - Can now avoid storing Large objects in row store tables.
  - Descriptor and in-lined data placed in Columnar data page
    - Large Objects stored in same buddy space as row tables
- **DEC\_ARITHMETIC**
  - Values: DEC.S, DEC15, DEC15.S, DEC31, DEC31.S
    - (S = Minimum Scale)
  - Control how precision and scale determined for results of decimal arithmetic
  - Provided for compatibility with Db2 for zOS
  - Db2 Warehouse defaults to DEC.6 for better Netezza compatibility

Large Object Support will be available in next release.

Allows for storing large string or binary values within the columnar table directly. No more need for side ROW tables.

LOB Storage itself is shared with row engine. Columnar data page stores the descriptor and inlined data but large values put out to the “buddy” system.

Columnar though in-lining can be up to approximately the page size regardless of other columns in table.

DEC\_ARITHMETIC DB CFG parameter allows configuring how Db2 calculates the precision and scale or a resulting decimal value.  
Replaces the old MIN\_DEC\_DIV\_3 and reg var that could be used to force 3 and 6 scale respectively.

DEC.S where S can be 1-9 specifies Db2 rules but with a minimum scale of S.

DEC15 provides compatibility with Db2 for zOS DECARTH DECP value of same (DEC15). Enforces precision to be at most 15.

DEC31 provides compatibility with Db2 for zOS DECARTH DECP value of same (DEC31).

You can also specify DEC15.S or DEC31.S to influence the minimum scale chosen. The decimal 15 and 31 settings also influence Db2 for zOS decimal arithmetic emulation.

## Db2 Family Compatibility (2 | 3)

- IFNULL
  - Identical to COALESCE but limited to two arguments
  - Returns first argument if not NULL otherwise second argument is returned
- Equivalent to CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END
- Example:
  - IFNULL('a','b') → 'a'
  - IFNULL(NULL,'b') → 'b'

IFNULL scalar function introduced to provide greater compatibility with Db2 for zOS

## Db2 Family Compatibility (3 | 3)

- Db2 Family naming for Oracle functions
  - ASCII\_STR (ASCIISTR)
    - Returns an ASCII version of the string. Any non-ASCII characters are escaped as “\xxxx or \xxxx\yyyy” where xxxx and yyyy are UTF-16 codepoints
  - UNICODE\_STR (UNISTR)
    - Second argument supported to determine if a UTF-8 or UTF-16 (VARGRAPHIC) string is returned
    - Processes an ASCII string converting “\xxxx or \xxxx\yyyy” UTF-16 codepoints into appropriate UTF-8 or UTF-16 characters.
- Db2 Family “SKIP LOCKED DATA” syntax supported as well

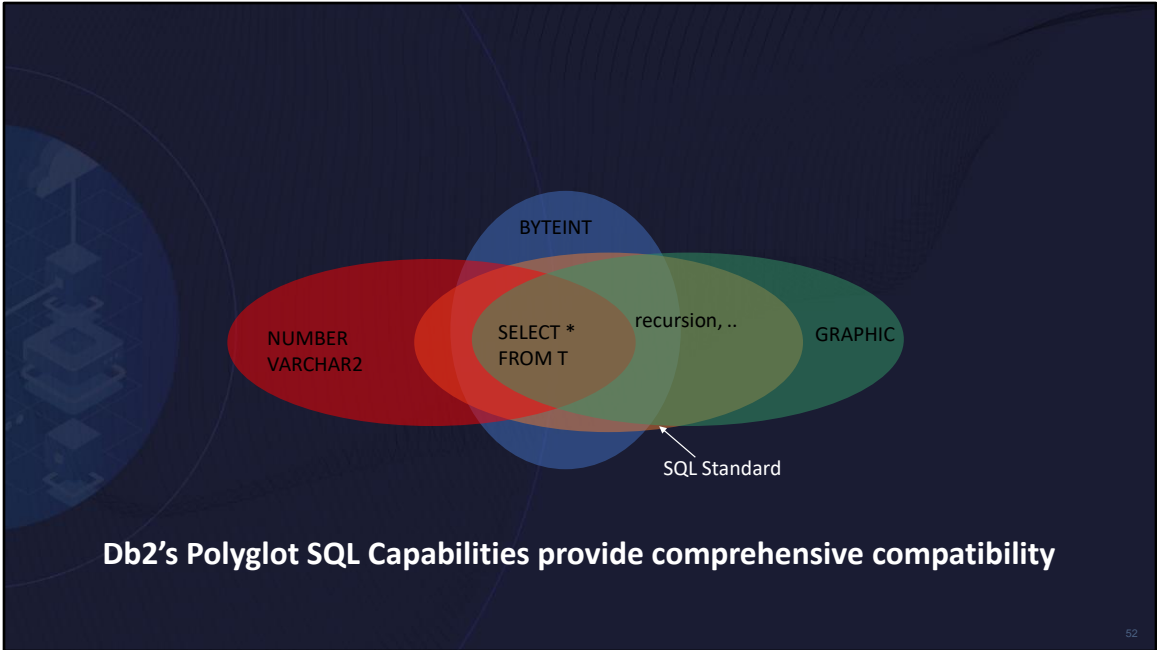
Db2 family variants of ASCIISTR and UNISTR have also been introduced including support for the additional second optional parameter on UNICODE\_STR that can be used to specify the resulting strings encoding (UTF-8 or UTF-16).

## Db2 REST APIs

*Build and maintain apps easily not worry about the **deployment specifics or scalability**  
Flexibility in using **technologies***

- REST services for IBM Db2 allows your web, mobile, and cloud application to interact with Db2 through a set of scalable RESTful APIs
- Provides the application developer APIs to **create, discover and execute** their own REST end-points, referred to as **services**, specific to their application needs
- Provides Db2 access without managing client drivers on end-user machines.
- Each **developer-defined service** is associated with a single SQL statement (SELECT, INSERT, UPDATE, DELETE initially)
- Services can be executed synchronously, or as a job that supports

While not directly compatible with DB2 for zOS's REST capabilities. Db2 has also been extended to support publishing REST endpoints to simplify modern application development.



**Db2's Polyglot SQL Capabilities provide comprehensive compatibility**

# Acknowledgements and Disclaimers

**Availability.** References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

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. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

© Copyright IBM Corporation 2021. All rights reserved.

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

IBM, the IBM logo, [ibm.com](http://ibm.com), and DB2 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 TM), 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](http://www.ibm.com/legal/copytrade.shtml)

\*Oracle is a trademark of Oracle Corporation

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



Speaker: Mike Springgay  
Company: IBM  
Email Address: [springga@caibm.com](mailto:springga@caibm.com)