

Cross Platform


Application Design

## SPs, UDFs, and UDTs Putting it all Together

Peter G Backlund  
Peter Backlund DB2-Konsult AB

Session: G12


**Wednesday, October 23rd, 16:15-17:15**



Recent versions of DB2 - both DB2 for z/OS and OS/390 and DB2 for Linux, UNIX, and Windows - have some very important additions,

- Stored Procedures
- User Defined Functions
- User Defined Distinct Types.


## Contents



1. Introduction
2. Procedures and Functions - Overview
3. Procedures - Some Details
4. Functions - Some Details
5. Distinct Types

Appendix

---



International DB2 Users Group

After a short introduction, we will take a look at basic aspects of procedures and functions; how they are invoked and what they return.

Thereafter we take a closer look at procedures and also compare DB2 for z/OS and OS/390 to DB2 for Linux, UNIX, and Windows.

We then do the same for functions.

The last point covers user defined distinct types.

Appendix A is a summary of all functions clarifying in what environment and from which version they can be used.

Appendix B is a summary of new functions in DB2 for LUW version 7.2.

## Detailed Contents



|   |   |
|---|---|
| 1 - Introduction                          | 23 - SPB - Sample Execution             |
| 2 - Procedures and Functions              | 24 - SPB - Sample Debugging             |
| 3 - Procedures and Functions...           | 25 - Functions - overview               |
| 4 - Procedures and Functions - Invocation | 26 - Column Functions                   |
| 5 - Invocation in Triggers                | 27 - Scalar Functions - Just a Sample   |
| 6 - Procedures and Functions - Result     | 28 - Scalar Functions - Examples        |
| 7 - Procedures and Functions - Definition | 29 - ORDER BY with Expression           |
| 8 - SP and UDF - Some Options             | 30 - LAST_DAY and WEEK                  |
| 9 - Procedures and Functions - Execution  | 31 - External UDF - C                   |
| 10 - Procedures - Languages               | 32 - SQL Scalar UDF                     |
| 11 - Functions - Languages                | 33 - Last_Day for DB2 LUW               |
| 12 - Procedures and Functions - Coding    | 34 - UDT - User Defined Distinct Types  |
| 13 - External Procedures and Functions -  | 35 - Sample UDT Definition              |
| 14 - SQL Procedure                        | 36 - Sample UDT Definition...           |
| 15 - SQL Procedure - Example              | 37 - Sample UDT Definition...           |
| 16 - Language Elements - Common           | 38 - What is the '+'-function?          |
| 17 - Language Elements - LUW Only         | 39 - Sample UDT Definition...           |
| 18 - SQL/PSM - Implementation 390         | 40 - UDTs and UDFs                      |
| 19 - SQL/PSM - Implementation LUW         | 41 - Summary                            |
| 20 - Stored Procedure Builder             | Appendix A - Scalar Functions           |
| 21 - SPB - Programming Languages          | Appendix B - New DB2 LUW V7.2 Functions |
| 22 - SPB - Sample                         |   |

---



International DB2 Users Group

## Disclaimer



- I work as an independent DB2 consultant
- Who is independent?
- The opinions and statements made in this presentation are entirely my responsibility



International DB2 Users Group

I worked for IBM 1968-1988 and for a consulting company 1988-1984. Since then I am running my own one-man consulting company. My main customer is IBM where I do a lot of IBM teaching. I have worked actively with DB2 since 1981.

The question is not whether I am independent, but rather that the opinions and viewpoints presented here are my own and that I am completely responsible for the content of this presentation.

If you use some of my advice, and they don't work out right; please don't sue me!

## Thanks



- Don Chamberlin

A Complete Guide to DB2 Universal Database  
(ISBN 1-55860-482-0)

- Graeme Birchall

DB2 UDB V7.2 - SQL Cookbook  
([http://ourworld.compuserve.com/homepages/Graeme\\_Birchall](http://ourworld.compuserve.com/homepages/Graeme_Birchall))

- Lennart Henäng



International DB2 Users Group

We all know that Dr. Ted Codd is the father of relational.

Don Chamberlin is one of the fathers of SQL. His book is the absolute best. Unfortunately there are two limitations, it covers DB2 LUW only and it is written for V5.2. Apart from that, the book is absolutely marvelous.

Graeme Birchall of New York is a great SQL enthusiast. His SQL Cookbook (which is freely available on the net) is excellent and he also keeps it well updated.

Finally I want to thank Lennart Henäng, formerly of IBM Sweden. Lennart is one of the great when it comes to understanding and also explaining all aspects of DB2 Besides telling me the difference between apples, oranges, and pears, he is also a very good friend of mine.

## 1 - Introduction



- **SP** - stands for Stored Procedure
- **UDF** - stands for User Defined Function
- **UDT** - stands for User Defined Distinct Type
  
- **DB2 390** (sometimes **390**) stands for DB2 Universal Database for z/OS and OS/390 Versions 6 and 7
  
- **DB2 LUW** (sometimes **LUW**) stands for DB2 Universal Database for Linux, UNIX, and Windows Versions 6, 7, and 7.2



International DB2 Users Group

IBM is very particular about terminology. Unfortunately the terminology changes from time to time.

DB2 is a registered trademark and has to be used properly.

I have taken the liberty to use the abbreviations "DB2 390" and "DB2 LUW" hoping that IBM won't complain to much.

We will start with procedures and functions and do some comparisons and pinpoint differences

## 2 - Procedures and Functions



- Procedures and Functions
  - have many similarities
  - but also some differences



International DB2 Users Group

## 3 - Procedures and Functions...



- Procedures (Stored Procedures) are new from versions 4 and 5
- Functions (Built-in Functions) have existed long time, some even from the beginning of DB2
  - Column Functions (Count, Sum, Avg, Min, and Max)
  - Scalar Functions (Coalesce, Date, Length, Substr, ...)
  - User Defined Functions are relatively new



International DB2 Users Group

Procedures (or rather Stored Procedures) were first introduced in DB2 390 in Version 4 back in 1994. This was a preliminary implementation and important improvements were made in V5 and V6. In DB2 LUW Stored Procedures have been available at least since Version 5.

Column and Scalar Functions were part of the first version of DB2. Many functions have been added in different versions and most notably in DB2 390 V6. In DB2 LUW most functions have been available at least since Version 5.

User Defined Functions are new to DB2 390 V6 and have been available at least since Version 5 in DB2 LUW.

## 4 - Procedures and Functions - Invocation



- A Procedure is **invoked** through the SQL CALL

```
Exec SQL  
  Call Procname ( Parm1, Parm2 );
```

- A Function is **used** in an expression

```
Exec SQL  
  Select Myfunc ( Parm1, Parm2 )  
    Into :hostvar  
  ...
```



International DB2 Users Group

The SQL CALL statement is used for invoking a stored procedure. The name can be a one-, two-, or three-part name and can also be a hostvariable. Optionally parameters can be passed. They can be literals or hostvariables and are of three types, IN (for passing info to the proc), OUT (for receiving info from the proc, or INOUT (for both passing and receiving). If there are no parameters, the parenthesis can be omitted.

A function is invoked by being used in an expression. The name can be a one- or two-part name. Optionally arguments (which only can be of input type; literals or hostvariables) can be passed to the function. Even if no arguments are passed, the parenthesis are mandatory.

## 5 - Invocation in Triggers



- A special case is the invocation of a function in a trigger; the VALUES statement is used

```
...  
VALUES ( Myfunc( Parm1, Parm2 ) )  
...
```

- For a procedure, the CALL is still used
  - but remember; only in DB2 390
- In DB2 LUW 7.2, a trigger can execute a dynamic compound statement with a subset of the SQL Procedure Language



International DB2 Users Group

A special situation is the use of procedures and functions in the body of triggers.

Procedures (which currently only can be used in DB2 390) are invoked using the CALL statement.

Functions can be used in expressions and can also be invoked "stand-alone" using the VALUES SQL clause.

A function in DB2 LUW can not contain SQL statements.

However, in DB2 LUW V7.2 it is possible to have "compound statements" (a begin-end block which can contain SQL/PL - more later) in user defined functions and triggers.

## 6 - Procedures and Functions - Result



- From a Procedure the result is returned as

- zero, one, or more parameters
- and
- zero, one, or more result sets

- From a Function the result is returned as

- a value
- or
- a table
- or
- a row



International DB2 Users Group

The way result is returned is different between procedures and functions.

For a procedure the result can be sent back in parameters and/or in result sets (an "internal" result table which can be processed using a cursor in the calling program.)

For a function, the result is normally returned as a value. It is also possible to return a table (which is used in the from clause) or (for DB2 LUW) a row.

## 7 - Procedures and Functions - Definition



- Procedures and Functions are defined (or rather registered) to DB2 using the SQL statements

- CREATE PROCEDURE  
procedure-name (parameters)  
option-list

- CREATE FUNCTION  
function-name (parameters)  
RETURNS data-type  
option-list



International DB2 Users Group

Both procedures and functions are DB2 objects and as such are defined (or registered) to DB2 through the DDL CREATE statement. As part of the function definition its data type has to be specified.

We can also use ALTER to change some of the attributes and DROP to remove a procedure or function.

## 8 - SP and UDF - Some Options

|                      | DB2 390 |     | DB2 LUW |     |
|----------------------|---------|-----|---------|-----|
|                      | SP      | UDF | SP      | UDF |
| Language             | X       | X   | X       | X   |
| Result Set           | X       | --  | X       | --  |
| No SQL               | X       | X   | X       | X   |
| Reads/Modifies SQL   | X       | X   | X       | -Y- |
| External Name        | X       | X   | X       | X   |
| No WLM environment   | X       | --  | --      | --  |
| WLM Environment Name | X       | X   | --      | --  |
| Fenced               | --      | --  | X       | X   |

-Y- indicates changes in V7 and V7.2

International DB2 Users Group

This slide illustrates some of the options available. The Language clause will be discussed later. The Result Set clause specifies the maximum number of result sets a procedure can return and is invalid for functions.

A UDF in DB2 LUW can not contain SQL-statements; but in V7 a return with select is possible and in V7.2 a compound statement can be used.

The WLM clause is only used with DB2 390 and is only applicable to procedures and is used to specify whether the procedure should run in the DB2SPAS or in a WLM-managed Address Space. DB2 390 external functions are always executed in a WLM address space. All DB2 390 procedures and functions are always "fenced" as they run in a separate address space.

## 9 - Procedures and Functions - Execution

- For DB2 390,
  - a procedure is executed
    - in the DB2 SPAS
    - in a WLM Address Space
  - a function is executed
    - in a WLM Address Space
    - internally (Built-in)
- For DB2 LUW,
  - a procedure or function is executed
    - in a separate process (Fenced)
    - in the DBMS process (Not Fenced)
    - internally (Built-in Functions)

International DB2 Users Group

This slide gives some details on the execution environment.

For DB2 390 procedures and functions always run in separate address spaces (except built-in functions.) For procedures this can be the DB2SPAS or a WLM address space.

For DB2 LUW procedures and functions execute in a separate process or in the DBMS process.

## 10 - Procedures - Languages

|          | DB2 390 |    | DB2 LUW |    |
|----------|---------|----|---------|----|
|          | V6      | V7 | V6      | V7 |
| ASSEMBLE | X       | X  |         |    |
| C        | X       | X  | X       | X  |
| COBOL    | X       | X  | X       | X  |
| COMPJAVA | +       | X  |         |    |
| JAVA     |         | X  | X       | X  |
| OLE      |         |    |         | X  |
| PLI      | X       | X  |         |    |
| REXX     | +       | X  |         |    |
| SQL      | +       | X  |         | X  |

International DB2 Users Group

This is an overview of available programming languages for writing procedures.

COMPJAVA (a special OS/390 "extension" to Java using a special compiler which translates Java byte code into machine code) and REXX and the SQL/PL are late additions to DB2 390 V6.

Common languages between DB2 390 and DB2 LUW are C (which includes C++) and COBOL. From version 7 JAVA and SQL/PL are also common.

## 11 - Functions - Languages



|          | DB2 390 |    | DB2 LUW |    |
|----------|---------|----|---------|----|
|          | V6      | V7 | V6      | V7 |
| ASSEMBLE | X       | X  |         |    |
| C        | X       | X  | X       | X  |
| COBOL    | X       | X  |         |    |
| JAVA     |         | X  | X       | X  |
| OLE      |         |    | X       | X  |
| OLEDB    |         |    | X       | X  |
| PLI      | X       | X  |         |    |
| SQL      |         | X  |         | Y  |

Y = In DB2 LUW 7.2, you can also use the SQL Procedure Language



International DB2 Users Group

This is an overview of available programming languages for writing external functions.

The only common language between DB2 390 and DB2 LUW is C (which includes C++).

From version 7 JAVA is also common.

For the SQL language, only a return statement can be used.

For DB2 LUW V7.2 a subset of SQL/PL can be used in a dynamic compound statement.

## 12 - Procedures and Functions - Coding



### • A Procedure can be

- External
- SQL

### • A Function can be

- (Built-in)
- External
- SQL
  
- Sourced



International DB2 Users Group

A procedure can be coded as an  
- external procedure (coded in COBOL, C, Java...  
- SQL procedure (which as we later will see results in a C procedure, so actually all procedures are external.)

A function can be  
- built-in (provided by IBM)  
- external (coded in C, Java...)  
- SQL (contained in a RETURN statement being an expression or a SELECT statement.) For DB2 LUW V7.2 this can be a "compound SQL-statement" using SQL/PL  
- sourced; copying an existing function

## 13 - External Procedures and Functions Programming



### • Without SQL-statements

- compile
- link

### • With SQL-statements

- precompile
- bind package
- compile
- link

➔ For a UDF; SQL is only allowed in DB2 390  
DB2 LUW V7.2 has compound statements



International DB2 Users Group

Programming of procedures and functions follows normal rules. If there are no SQL-statements, it is just a matter of compiling and link-editing.

If there are SQL-statements you have to precompile, bind package, compile, and link-edit.

The resulting loadmodule (or DLL) has to be placed in the appropriate library.

For functions you can only have SQL in DB2 390.

For DB2 LUW 7.2 functions you can use compound statements (SQL/PL) and that way use SQL.

## 14 - SQL Procedure



- The entire procedure is written in SQL/PSM
- SQL/PSM (Persistent Stored Modules) is an ANSI/ISO-standard for SQL-coding
- Other common procedure languages
  - Oracle: PL/SQL
  - Sybase: Transact SQL
  - SQL Server: Transact SQL
- We use the expression SQL Procedure and SQL Procedure Language (SQL/PL)



International DB2 Users Group

An SQL Procedure is written using the ANSI/ISO-standard for SQL/PSM.

This language is also used under other names in other relational DBMS's.

## 15 - SQL Procedure - Example



A simple example of an SQL procedure

```
CREATE PROCEDURE UPDATE_SALARY_1
  (IN EMPLOYEE_NUMBER CHAR(10),
  IN RATE DECIMAL(6,2))
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
  UPDATE EMP
  SET   SALARY = SALARY * RATE
  WHERE EMPNO = EMPLOYEE_NUMBER
END
```



International DB2 Users Group

This is a very simple example of an SQL Procedure.

The actual code is contained in a BEGIN - END block.

Two parameters are passed to the procedure and the EMP table is updated with a new salary.

## 16 - Language Elements - Common



- \* Compound Statements - BEGIN...END
- \* Declare Statements - Variables, Conditions ...
- \* SET - assignment
- \* IF, THEN, ELSE - controls conditional execution
- CASE - determines which execution path to follow
- \* LEAVE - terminates execution of labeled statements
- LOOP - repeats statements until LEAVE
- \* WHILE - repeats statements until termination test fails
- REPEAT - repeats statements until termination test is true
- CALL - calls a stored procedure
- \* GET DIAGNOSTICS - SQLERRD(3)

\*-marked can be used in DB2 LUW V7.2 triggers and SQL functions



International DB2 Users Group

Here are the language elements which can be used both in DB2 390 and DB2 LUW.

It is possible to code complete logic in the procedure. Local variables can be declared and used.

There are also (limited) possibilities for error handling.

## 17 - Language Elements - LUW Only



- Compound Statements can be nested
- `ALLOCATE CURSOR` - handling result sets
- `ASSOCIATE LOCATOR` - ditto
- \* `FOR` - looping through a table
- \* `ITERATE` - in a loop
- `RETURN` - terminate procedure
- `RESIGNAL` - error handling
- \* `SIGNAL` - error handling

\*-marked can be used in DB2 LUW V7.2  
triggers and SQL functions



International DB2 Users Group

DB2 LUW allows more options in the SQL Procedure.

Most notably is the ability to use nested compound statements (BEGIN-END blocks) and also handling of result sets from called procedures.

This illustrates one of the many differences between  
- DB2 for z/OS and OS/390  
- DB2 for Linux, UNIX, and Windows.

## 18 - SQL/PSM - Implementation 390



- The entire procedure is written in SQL/PL
- The procedure is registered to DB2, and then C-code is generated which is handled in the normal way (precompile, bind package, compile, and linkedit).

This can be done in three different ways:

1. Manually with JCL or CLIST
2. With the Stored Procedure DSNTPSMP
3. With the Stored Procedure Builder



International DB2 Users Group

Creating SQL Procedures for DB2 390 can be done manually through JCL or a CLIST.

It is also possible to do it using the Stored Procedure DSNTPSMP (which is delivered as part of DB2 390 V6 and later and which has to be executed in a WLM address space.)

The most elegant way is to the SPB - more later.

## 19 - SQL/PSM - Implementation LUW



- The entire procedure is written in SQL/PL
- The procedure is registered to DB2, and then C-code is generated which is handled in the normal way (prepare, compile, and link into a DLL.)

This can be done in two different ways:

1. Interactively from the Command Line Processor (or from the Command Center)
2. With the Stored Procedure Builder

- For DB2 LUW V7.2, when SQL/PL is used in triggers or SQL functions, no C-code is generated



International DB2 Users Group

Also in the DB2 LUW world we can create SQL Procedures manually.

However, again the best way is to use the SPB.

As was the case for DB2 390, the SQL procedure will generate C-code.

This is not the situation when SQL/PL is used in triggers and SQL Functions (available in DB2 LUW V7.2.)

## 20 - Stored Procedure Builder



- A Windows NT-based tool used to manage Stored Procedures
  - develop
  - test
  - deploy
- Developing and testing is done locally using DB2 for Windows NT
- After testing, the SP can be generated for different environments, for instance for DB2 for z/OS and OS/390



International DB2 Users Group

The Stored Procedure Builder executes in an NT environment and can be used in all phases of procedure development and management in DB2 LUW.

Procedures can be deployed into a 390 environment.

Stored procedures in DB2 390 can be executed (and debugged) using the SPB.

## 21 - SPB - Programming Languages



- Stored Procedure Builder can use
  - Java (JDBC or SQLJ)
  - SQL/PSM
- Distributed Debugger (part of Visualage for Java) can be used with Java.
- An integrated debugger for SQL/PL became available in DB2 for Windows NT V7.2
- Remote debugging can be used for procedures executed in DB2 for z/OS and OS/390



International DB2 Users Group

In the SPB procedures can be developed either in Java (both JDBC and SQLJ can be used) or in SQL/PL.

A distributed debugger can be used for Java procedures.

As of DB2 for Windows NT V7.2 there is an integrated debugger which can be used with procedures in SQL/PL.

For procedures executing in DB2 390 a remote debugger can be used.

## 22 - SPB - Sample



The screenshot shows the IBM DB2 Stored Procedure Builder interface. The top menu bar includes File, Edit, Selected, Debug, and Help. The main window is divided into three panes:

- Left pane:** A project tree showing a project named 'SAMPLE' with a sub-project 'Stored Procedures'. Under 'Stored Procedures', there is a folder 'DB2ADMINIDUG\_NA2002' containing three files: 'SYSFUN.GET\_ROUTINE\_SAR', 'SYSFUN.PUT\_ROUTINE\_SAR', and 'SYSFUN.PUT\_ROUTINE\_SAR'.
- Right pane:** A source code editor showing the SQL code for a stored procedure. The code includes a 'CREATE PROCEDURE' statement, a 'LANGUAGE SQL' declaration, and a 'BEGIN' block with a cursor declaration and a 'SELECT' statement. The code ends with 'END P1'.
- Bottom pane:** A messages window showing the output of the procedure build process. The messages indicate that the stored procedure was created, changes were committed, a package was created, and the build was successful.

This slide illustrates the SPB with its five panes (of which three can be seen simultaneously.)

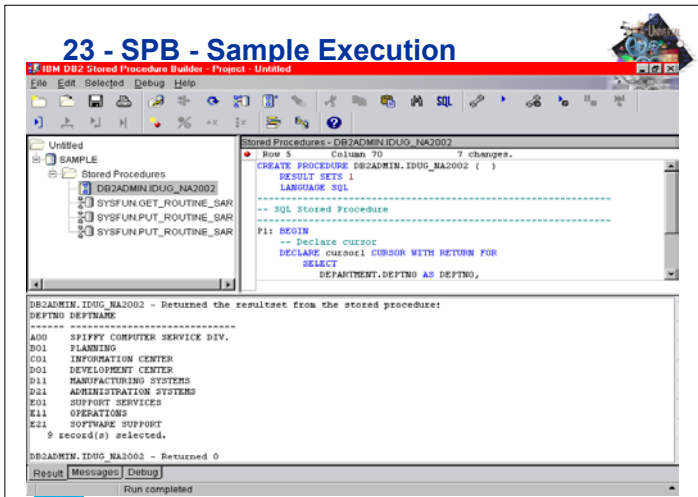
The left upper left pane is the three pane.

The upper right pane is the object pane with either procedure attributes or source code of the current procedure - which can be edited. The shown procedure is generated using a wizard which among other things contains an SQL Assist.

The bottom pane shows messages, in this case from building the procedure (registering, precompiling, compiling, linkediting, and bind package.)

This slide shows execution of the procedure.

The bottom pane now shows the result.

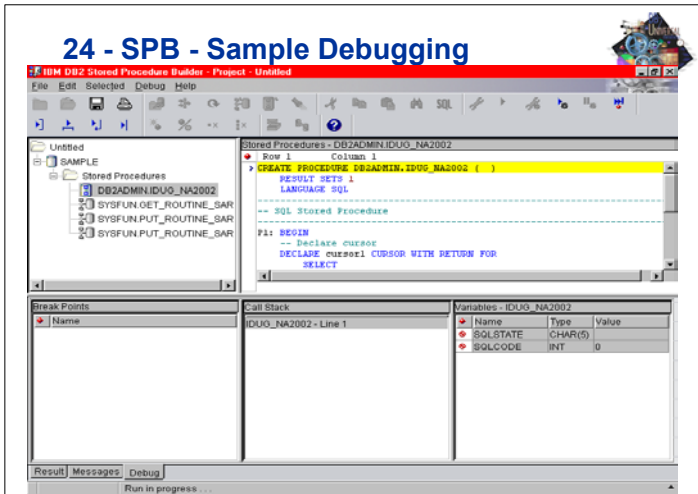


This slide shows execution of the procedure in debug mode.

The bottom pane now shows the debugging pane which is split into three parts,

- break points
- call stack
- variables.

Break points can be set for program lines and/or when some variable changes its value.



## 25 - Functions - overview

- Built-in Functions
  - scalar or column
- External Functions
  - scalar or table
- SQL Functions
  - scalar or (for WS) table or row
- Sourced Functions - "copies" of
  - built-in (scalar or column)
  - external (scalar)
  - SQL (scalar)

Functions can be classified as

- built-in (all provided by IBM)
- external (invoking a separate program)
- SQL (only a RETURN statement; for DB2 LUW 7.2 compound statement)
- sourced (on a built-in, external, or SQL function.)

Another way of classification is based upon what is returned

- scalar (a single value)
- column (a single value based on a column, all provided by IBM)
- table (a table, can only be used in FROM)
- row (a single row.)



## 26 - Column Functions



|                      | DB2 390 |    |    | DB2 LUW |    |
|----------------------|---------|----|----|---------|----|
|                      | V5      | V6 | V7 | V6      | V7 |
| AVG                  | x       | x  | x  | x       | x  |
| CORRELATION          |         |    |    | x       | x  |
| COUNT                | x       | x  | x  | x       | x  |
| COUNT_BIG            |         | x  | x  | x       | x  |
| COVARIANCE           |         |    |    | x       | x  |
| GROUPING             |         |    |    | x       | x  |
| MAX                  | x       | x  | x  | x       | x  |
| MIN                  | x       | x  | x  | x       | x  |
| OLAP functions       |         |    |    |         | x  |
| REGRESSION functions |         |    |    | x       | x  |
| STDDEV               |         | x  | x  | x       | x  |
| SUM                  | x       | x  | x  | x       | x  |
| VARIANCE             |         | x  | x  | x       | x  |

International DB2 Users Group

The original five column functions in DB2 were COUNT, MIN, MAX, SUM, and AVG.

Version 6 added a few more.

CORRELATION, COVARIANCE, STDDEV, VARIANCE and the REGRESSION functions are mainly of interest for statisticians and mathematicians.

GROUPING is a function used with the CUBE and ROLLUP operations in DB2 LUW GROUP BY.

The OLAP functions (new in DB2 LUW V7) are DENSE\_RANK, RANK, and ROW\_NUMBER.

## 27 - Scalar Functions - Just a Sample



|                     | DB2 390 |    |    | DB2 LUW |    |
|---------------------|---------|----|----|---------|----|
|                     | V5      | V6 | V7 | V6      | V7 |
| COALESCE (or VALUE) | x       | x  | x  | x       | x  |
| SIN                 |         | x  | x  | x       | x  |
| IDENTITY_VAL_LOCAL  |         | +  | x  |         | x  |
| CHAR                | x       | x  | x  | x       | x  |
| POSSTR              |         | x  | x  | x       | x  |
| LOCATE              |         | x  | x  | x       | x  |
| INSERT              |         | x  | x  | x       | x  |
| REPLACE             |         | x  | x  | x       | x  |
| TRANSLATE           |         | x  | x  | x       | x  |
| LAST_DAY            |         |    | x  |         |    |
| WEEK                |         | x  | x  | x       | x  |
| WEEK_ISO            |         |    | x  |         | x  |

International DB2 Users Group

This slide shows a few of the many scalar functions provided with DB2. Most of them were added in V6.

Last\_Day is one of the few which is available only in DB2 390.

The next few slides discusses some functions.

## 28 - Scalar Functions - Examples



```
posstr('Peter','e')      ==> 2
locate('e','Peter')     ==> 2
locate('e','Peter',3)   ==> 4
insert('Backlund',2,3,'erg') ==> 'Berglund'
insert('Adam',2,0,'mster') ==> 'Amsterdam'
replace('Peter','e','zz') ==> 'Pzztzzr'
translate('IMS','B2D','MSI') ==> 'DB2'
```

International DB2 Users Group

The posstr and locate functions are both used to find the "needle in the haystack" and as you see they have different order of the arguments. Locate can have a starting position.

The insert function operates on the first argument, inserting the fourth argument at the position specified by the second argument after deleting the number of characters specified by the third argument.

The replace function operates on the first argument, replacing each occurrence of the second argument with the third argument.

The translate function operates on the first argument, replacing characters from the third argument with positionally corresponding characters in the second argument.

## 29 - ORDER BY with Expression



```
select name
from tab order by
name
able
ABLE
baker
BAKER

select name
from tab order by
translate(name,'abAB','ABab')
ABLE
able
BAKER
baker

select name
from tab order by
hex(translate(name,'abAB','ABab'))
able
baker
ABLE
BAKER
```



International DB2 Users Group

An interesting use of the translate function is the ability to modify the sort order. In V6 you have to include the translate in the select list, but from V7 it can be put directly into the order by.

The examples shown are for DB2 LUW and won't work exactly the same way in DB2 390.

An important use of this function is to get national characters (Nordic countries, Germany, and others) in the proper order.

## 30 - LAST\_DAY and WEEK



```
SET :END_OF_MONTH = LAST_DAY(CURRENT_DATE)

END_OF_MONTH will be '2002-05-31'

Week(Current_Date) ==> 18

Week('2000-01-01') ==> 1
Week_ISO('2000-01-01') ==> 52

Week('2000-12-31') ==> 54
Week_ISO('2000-12-31') ==> 52
```



International DB2 Users Group

The Last\_Day function (only in DB2 390) returns the last day of the month for the given date.

In the United States the week begins on a Sunday, and January 1st always is in week number 1.

In Europe the week begins on a Monday and week number 1 is the first week of the year which has at least 4 days within the current year (thus Jan 4th is always in week 1.)

In order to handle this, the dayofweek\_iso and week\_iso functions have been added.

## 31 - External UDF - C



```
CREATE FUNCTION km_to_mi (km FLOAT)
RETURNS FLOAT
EXTERNAL NAME 'udfddl!kmtomi'
LANGUAGE C
PARAMETER STYLE DB2SQL
NO SQL
```

```
void SQL_API_FN kmtomi
(
    double *km,
    double *mi,
    ... )
{
    *mi = *km/1.609;
    return;
}
```



International DB2 Users Group

This slide shows a very simple external user defined function which is used to convert kilometers to miles.

The top half shows the function being defined to DB2; in this case LUW.

The bottom half shows the c-code which has to be compiled and linked as "member" kmtomi in the udfddl DDL.

## 32 - SQL Scalar UDF



```
create function mi_to_km(mi double)
returns double
language sql
return 1.609*mi
```



International DB2 Users Group

The function on the previous slide required a c-program.

This example gives the reverse, miles to kilometers, but implemented as an SQL scalar function.

This requires no compilation of an external program and is available as of Version 7.

The return statement in this example is a constant, but it can be any SQL select statement.

In DB2 LUW V7.2 the statement can be a dynamic compound statement, containing a subset of the SQL Procedure Language.

## 33 - Last\_Day for DB2 LUW



```
create function last_day(in_date date)
returns date
language sql
return date(substr(char(in_date),1,8) concat '01')
+ 1 month - 1 day

-- values (last_day(current date))          '2001-05-31'
-- values (last_day(date('2000-02-13')))    '2000-02-29'
-- values (last_day(date('2001-12-13')))    '2001-12-31'
```



International DB2 Users Group

The Last\_Day function is not available in DB2 LUW. This slide shows how easy it is to create your own SQL function. From the input date we find the first day of the month, add one month, and subtract one day.

The VALUES clause can be used to invoke any function.

This example was created using QMF for Windows, but the CLP (Command Line Processor) or the Command Center could also have been used.

## 34 - UDT - User Defined Distinct Types



- The goal is better quality of SQL statements
- Avoids "apples and oranges"
- Also known as "strong typing"
- Data types and allowed operations are checked at precompile and bind/prepare
- Does not affect execution performance



International DB2 Users Group

Data quality is extremely important. Check Constraints, Referential Constraints, and Triggers can achieve this.

Equally important is the quality of SQL statements. User Defined Distinct Types can be used to support this. UDTs are based upon existing DB2 data types and are "restored" to the base type at bind/prepare time. Thus performance is not affected by UDTs.

UDTs are verified at precompile and bind time to insure that only predefined comparisons and operations are done.

For example, it wouldn't make sense to add a shoe size to an amount, or to multiply two amounts.

### 35 - Sample UDT Definition



- Create distinct type km as integer with comparisons
- Create distinct type mi as integer with comparisons
- Create table us\_dist  
( from char(20)  
, till char(20)  
, dist\_mi mi  
, dist\_km km)



International DB2 Users Group

This and following slides shows an example of UDTs.

First we define two UDTs, km (for kilometer) and mi (for miles.) Both are defined as integers and at execution time will be treated as such. We also specify that two km columns can be compared.

We create a table, using the new types for two of the columns.

### 36 - Sample UDT Definition...



- You can't write...  
where dist\_mi > dist\_km
- You can't write...  
where dist\_mi = 20
- But you can write...  
where dist\_mi = mi(20)  
where integer(dist\_mi) = 20
- DB2 generates CAST functions  
- mi(integer)    - km(integer)  
- integer(mi)    - integer(km)



International DB2 Users Group

Let's try some operations with these columns.

dist\_mi and dist\_km are different data types and can not be compared. dist\_mi is of type mi and 20 is an integer, so they are not compatible.

We have to use cast functions to perform the comparisons. The cast functions "converts" one data type to another. When a UDT is defined, DB2 automatically generates two cast functions, between the UDT and its base type and vice versa.

### 37 - Sample UDT Definition...



- We have to write our own functions (UDF)  
- km\_to\_mi(km) returns mi  
- mi\_to\_km(mi) returns km
- Now we can write...  
where dist\_mi > km\_to\_mi(dist\_km)
- But not...  
select dist\_mi + km\_to\_mi(dist\_km)



International DB2 Users Group

If we want to compare mi (miles) to km (kilometers) we have to write conversion functions.

The last select statement is an addition of two expressions, both having the data type mi. Still this is not allowed, because we must first specify that addition of miles is a permitted operation. This way we can for instance specify that addition and subtraction is OK, but not multiplication or division.

How do we specify that addition is allowed?

## 38 - What is the '+'-function?



A + B

can be expressed as

+( A , B )



International DB2 Users Group

Well, addition of two terms is nothing else but the "plus"-function applied to the two terms.

So to allow addition, we have to define the plus function.

## 39 - Sample UDT Definition...



- We have to define allowed operations...

```
Create function "+" (mi,mi)
  returns mi
  source
  sysibm."+" (integer,integer)
```

- Now this is OK...

```
select dist_mi + km_to_mi(dist_km)
```



International DB2 Users Group

The plus function for miles can be "sourced" on the existing plus function for integers (as mi is defined as integer.)

The source for this plus function is the IBM-supplied function SYSIBM."+"(integer,integer)

The same technique is used for all needed functions.

After this definition we can add two mi (miles) terms.

## 40 - UDTs and UDFs



- Assume that we have defined data types
  - EURO
  - USD
- Functions for conversion between these datatypes will probably have to find the current conversion rate in some DB2 table.
- But remember...  
External UDFs in DB2 LUW can not contain SQL!
- However...  
From V7 we can use SQL Scalar UDFs  
and in DB2 LUW V7.2 Compound Statements (SQL/PL)



International DB2 Users Group

The functions for conversion between miles and kilometers are fairly simple to implement.

If we need conversions between other data types, we may have to reference DB2 tables and therefore need SQL statements. Before V7, this could only be done in DB2 390.

In V7, SQL functions can contain an SQL select in the return statement and this way data can be read from DB2 tables.

With DB2 LUW V7.2 it is possible to have a compound statement in an SQL function, thereby allowing business logic in the function. Such functions are of course not compatible with DB2 390.

## 41 - Summary



- Procedures, Functions, and Data Types are important extensions to the Relational Database Model
- Unfortunately, there are many differences between
  - DB2 for z/OS and OS/390
  - and
  - DB2 for Linux, UNIX, and Windows
- Stored Procedure Builder provides a very productive development environment



International DB2 Users Group

The purpose of this presentation was to show the importance of the new objects which have been added to DB2.

Procedures and Functions can be used to implement business logic and to improve data quality. In a client/server environment procedures can be important from a performance point of view.

Data Types are an important way of improving the quality of SQL statements.

The Stored Procedure Builder is an excellent tool for improving development productivity.

Session Title: SPs, UDFs, and UDTs - Putting it all Together  
Session #: G12



**Peter G Backlund**  
**Peter Backlund DB2-Konsult AB**  
**pbacklu@attglobal.net**



International DB2 Users Group

This first appendix is ten slides summarizing all built-in scalar functions provided with DB2.

For each function is shown in which version of DB2 390 or DB2 LUW it was made available.

## Appendix A

### Built-in Scalar Functions

#### A1 - Scalar Functions

|                 | DB2 390 |    |    | DB2 LUW |    |
|-----------------|---------|----|----|---------|----|
|                 | V5      | V6 | V7 | V6      | V7 |
| ABS or ABSVAL   | X       | X  |    | X       | X  |
| ACOS            | X       | X  |    | X       | X  |
| ADD_MONTHS      |         |    | X  |         |    |
| ASCII           |         |    |    | X       | X  |
| ASIN            | X       | X  |    | X       | X  |
| ATAN            | X       | X  |    | X       | X  |
| ATANH           | X       | X  |    |         |    |
| ATAN2           | X       | X  |    | X       | X  |
| BIGINT          |         |    |    | X       | X  |
| BLOB            | X       | X  |    | X       | X  |
| CCSID_ENCODING  |         |    | X  |         |    |
| CEIL or CEILING | X       | X  |    | X       | X  |

#### A2 - Scalar Functions

|                     | DB2 390 |    |    | DB2 LUW |    |
|---------------------|---------|----|----|---------|----|
|                     | V5      | V6 | V7 | V6      | V7 |
| CHAR                | X       | X  | X  | X       | X  |
| CLOB                |         | X  | X  | X       | X  |
| COALESCE            | X       | X  | X  | X       | X  |
| CONCAT              |         | X  | X  | X       | X  |
| COS                 |         | X  | X  | X       | X  |
| COSH                |         | X  | X  |         |    |
| COT                 |         | X  | X  | X       | X  |
| DATA LINK Functions |         |    |    | X       | X  |
| DATE                | X       | X  | X  | X       | X  |
| DAY                 | X       | X  | X  | X       | X  |
| DAYNAME             |         |    |    | X       | X  |
| DAYOFMONTH          |         | X  | X  |         |    |

### A3 - Scalar Functions

|                 | DB2 390 |    |    | DB2 LUW |    |
|-----------------|---------|----|----|---------|----|
|                 | V5      | V6 | V7 | V6      | V7 |
| DAYOFWEEK       |         | X  | X  | X       | X  |
| DAYOFWEEK_ISO   |         |    | X  |         | X  |
| DAYOFYEAR       |         | X  | X  | X       | X  |
| DAYS            | X       | X  | X  | X       | X  |
| DBCLOB          |         | X  | X  | X       | X  |
| DECIMAL         | X       | X  | X  | X       | X  |
| DEGREES         |         | X  | X  | X       | X  |
| DEREF           |         |    |    | X       | X  |
| DIFFERENCE      |         |    |    | X       | X  |
| DIGITS          | X       | X  | X  | X       | X  |
| DOUBLE          |         | X  | X  | X       | X  |
| EVENT_MON_STATE |         |    |    | X       | X  |

### A4 - Scalar Functions

|                    | DB2 390 |    |    | DB2 LUW |    |
|--------------------|---------|----|----|---------|----|
|                    | V5      | V6 | V7 | V6      | V7 |
| EXP                |         | X  | X  | X       | X  |
| FLOAT              | X       | X  | X  | X       | X  |
| FLOOR              |         | X  | X  | X       | X  |
| GENERATE_UNIQUE    |         |    |    | X       | X  |
| GRAPHIC            |         | X  | X  | X       | X  |
| HEX                | X       | X  | X  | X       | X  |
| HOUR               | X       | X  | X  | X       | X  |
| IDENTITY_VAL_LOCAL |         | X  | X  |         | X  |
| IFNULL             |         | X  | X  |         |    |
| INSERT             |         | X  | X  | X       | X  |
| INTEGER            | X       | X  | X  | X       | X  |
| JULIAN_DAY         |         | X  | X  | X       | X  |

### A5 - Scalar Functions

|                 | DB2 390 |    |    | DB2 LUW |    |
|-----------------|---------|----|----|---------|----|
|                 | V5      | V6 | V7 | V6      | V7 |
| LAST_DAY        |         |    | X  |         |    |
| LCASE or LOWER  |         | X  | X  | X       | X  |
| LEFT            |         | X  | X  | X       | X  |
| LENGTH          | X       | X  | X  | X       | X  |
| LN              |         | X  | X  | X       | X  |
| LOCATE          |         | X  | X  | X       | X  |
| LOG             |         | X  | X  | X       | X  |
| LOG10           |         | X  | X  | X       | X  |
| LONG_VARCHAR    |         | X  | X  |         |    |
| LONG_VARGRAPHIC |         | X  | X  |         |    |
| LTRIM           |         | X  | X  | X       | X  |
| MAX             |         |    | X  |         |    |

## A6 - Scalar Functions

|                  | DB2 390 |    |    | DB2 LUW |    |
|------------------|---------|----|----|---------|----|
|                  | V5      | V6 | V7 | V6      | V7 |
| MICROSECOND      | X       | X  | X  | X       | X  |
| MIDNIGHT_SECONDS |         | X  | X  | X       | X  |
| MIN              |         |    | X  |         |    |
| MINUTE           | X       | X  | X  | X       | X  |
| MOD              |         | X  | X  | X       | X  |
| MONTH            | X       | X  | X  | X       | X  |
| MONTHNAME        |         |    |    | X       | X  |
| MULTIPLY_ALT     |         |    | X  |         |    |
| NEXT_DAY         |         |    | X  |         |    |
| NODENUMBER       |         |    |    | X       | X  |
| NULLIF           | X       | X  | X  | X       | X  |
| PARTITION        |         |    |    | X       | X  |

## A7 - Scalar Functions

|                 | DB2 390 |    |    | DB2 LUW |    |
|-----------------|---------|----|----|---------|----|
|                 | V5      | V6 | V7 | V6      | V7 |
| POSSTR          |         | X  | X  | X       | X  |
| POWER           |         | X  | X  | X       | X  |
| QUARTER         |         | X  | X  | X       | X  |
| RADIANS         |         | X  | X  | X       | X  |
| RAISE_ERROR     |         | X  | X  | X       | X  |
| RAND            |         | X  | X  | X       | X  |
| REAL            |         | X  | X  | X       | X  |
| REPEAT          |         | X  | X  | X       | X  |
| REPLACE         |         | X  | X  | X       | X  |
| RIGHT           |         | X  | X  | X       | X  |
| ROUND           |         | X  | X  | X       | X  |
| ROUND_TIMESTAMP |         |    | X  |         |    |

## A8 - Scalar Functions

|          | DB2 390 |    |    | DB2 LUW |    |
|----------|---------|----|----|---------|----|
|          | V5      | V6 | V7 | V6      | V7 |
| ROWID    |         | X  | X  |         |    |
| RTRIM    |         | X  | X  | X       | X  |
| SECOND   | X       | X  | X  | X       | X  |
| SIGN     |         | X  | X  | X       | X  |
| SIN      |         | X  | X  | X       | X  |
| SINH     |         | X  | X  |         |    |
| SMALLINT |         | X  | X  | X       | X  |
| SOUNDEX  |         |    |    | X       | X  |
| SPACE    |         | X  | X  | X       | X  |
| SQRT     |         | X  | X  | X       | X  |
| STRIP    | X       | X  | X  |         |    |
| SUBSTR   | X       | X  | X  | X       | X  |

## A9 - Scalar Functions

|                   | DB2 390 |    |    | DB2 LUW |    |
|-------------------|---------|----|----|---------|----|
|                   | V5      | V6 | V7 | V6      | V7 |
| TABLE_NAME        |         |    |    | X       | X  |
| TABLE_SCHEMA      |         |    |    | X       | X  |
| TAN               |         | X  | X  | X       | X  |
| TANH              |         | X  | X  |         |    |
| TIME              | X       | X  | X  | X       | X  |
| TIMESTAMP         | X       | X  | X  | X       | X  |
| TIMESTAMPDIFF     |         |    |    | X       | X  |
| TIMESTAMP_FORMAT  |         |    | X  |         |    |
| TIMESTAMP_ISO     |         |    |    | X       | X  |
| TRANSLATE         |         | X  | X  | X       | X  |
| TRUNCATE or TRUNC |         | X  | X  | X       | X  |
| TRUNC_TIMESTAMP   |         |    | X  |         |    |

## A10 - Scalar Functions

|                | DB2 390 |    |    | DB2 LUW |    |
|----------------|---------|----|----|---------|----|
|                | V5      | V6 | V7 | V6      | V7 |
| TYPE_ID        |         |    |    | X       | X  |
| TYPE_NAME      |         |    |    | X       | X  |
| TYPE_SCHEMA    |         |    |    | X       | X  |
| UCASE or UPPER |         | X  | X  | X       | X  |
| VALUE          | X       | X  | X  | X       | X  |
| VARCHAR        |         | X  | X  | X       | X  |
| VARCHAR_FORMAT |         |    | X  |         |    |
| VARGRAPHIC     | X       | X  | X  | X       | X  |
| WEEK           |         | X  | X  | X       | X  |
| WEEK_ISO       |         |    | X  |         | X  |
| YEAR           | X       | X  | X  | X       | X  |

The final appendix has two slides showing new procedures and functions added to DB2 LUW Version 7.2

Besides of 12 scalar functions, 2 table functions are added.

The two procedures are provided as a tool for copying stored procedures between environments.

## Appendix B

New Functions and Procedures in  
DB2 for Linux, UNIX, and Windows V7.2

## B1 - New V7.2 Scalar Functions

---

- CHR
- DECRYPT\_BIN
- DECRYPT\_CHAR
- ENCRYPT
- GETHINT
- MQPUBLISH
- MQREAD
- MQRECEIVE
- MQSEND
- MQSUBSCRIBE
- MQUNSUBSCRIBE
- REC2XML

## B2 - New V7.2 Table Functions and Procedures

---

- Table Functions
  - MQREADALL
  - MQRECEIVEALL
  - SQLCACHE\_SNAPSHOT
- Procedures
  - GET\_ROUTINE\_SAR
  - PUT\_ROUTINE\_SAR

Session Title: SPs, UDFs, and UDTs - Putting it all Together  
Session #: G12



**Peter G Backlund**  
**Peter Backlund DB2-Konsult AB**  
**[pbacklu@attglobal.net](mailto:pbacklu@attglobal.net)**



International DB2 Users Group