sqladria

SQLADRIA SEMINAR

Zagreb, 28 September 2012

Sponsored by

bmcsoftware

SQLAdria Seminar

10.00 Opening

10.15 The natives are getting restless; an overview of DB2 stored procedures

Peter Plevka

BMC Software

11.15 Recent changes in DB2 for z/OS logging

Steve Thomas

BMC Software

12.15 Coffee Break

12.30 A new catalog and directory structure in DB2 for z/OS

Steve Thomas

BMC Software

13.30 Lunch

14.30 MDM, DG, RDM, DM, IG …

Goran Bavčar

NLB d.d.

15.00 Open forum

Sponsored by:

Zagreb, 28th September 2012

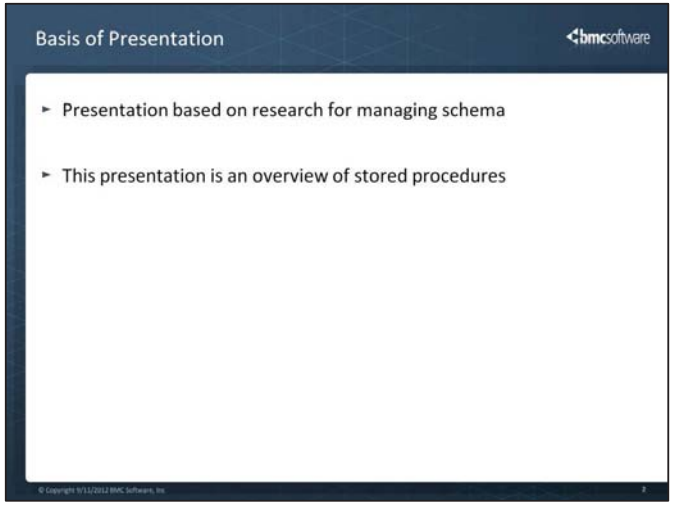
Hotel Palace

SQLADRIA SEMINAR
Zagreb, 28 September 2012

The natives are getting restless; an overview of DB2 stored procedures

Peter Plevka
BMC Software

Stored Procedures were first added to DB2 for z/OS back in V5. Most every release of DB2 since has added additional functionality or type of stored procedures. The different types of stored procedures have strengths and weaknesses. This presentation will provide a brief history of Stored Procedures in DB2 for z/OS, along with an overview of the schema used to create stored procedures. It will discuss the different types of stored procedures, External, External SQL and Native SQL touching on some of the advantages and disadvantages of the different types.



This presentation is the result of years of research of stored procedures. However, the research was biased toward schema management. We did not use stored procedures in applications. Therefore, the perspective of this presentation may be different from one given by DBA's who are concerned with using stored procedures as part of a business application.

We were concerned with the different options available for stored procedures and how those options may interrelate with other options. We did not do a great deal of research on exactly how those options may affect the execution of a stored procedure.

Agenda

- History of Stored Procedures supported by DB2
- Anatomy of Stored Procedures
- The 3 types of Stored Procedures
- Advantages of one type over another
- Native SQL: Deployment and Schema Management

History of Stored Procedures in DB2

- Stored Procedures introduced in DB2v5
  - Required manual insertion into catalog
  - Originally the DB2 'scripting' language was C, COBOL, etc.
- 'CREATE PROCEDURE' DDL syntax introduced in DB2v6
- External SQL procedures introduced in DB2v7
- Native SQL procedures introduced in DB2v9

Each type of procedure will be described in a later slide. We just wanted you to have an idea of the progression over the years.

We will discuss later that DB2's 'scripting language' is SPL or SQL Procedural Language.

When stored procedures were originally being introduced to DBMS's, Oracle, Sybase, etc. they created 'scripting languages' for their stored procedures. But IBM believed it already had excellent 'scripting languages' in COBOL, C, PL/I etc. Hence the birth of the External SQL procedure which will be discussed in more detail later.
But it COBOL etc... Was too weak for procedural language. Therefore SPL was introduced by IBM.
Native SQL are also based on SPL, but MUCH easier to maintain.

Agenda

- History of Stored Procedures supported by DB2
- Anatomy of Stored Procedures
- The 3 types of Stored Procedures
- Advantages of one type over another
- Native SQL: Deployment and Schema Management

Anatomy of a Procedure

```
CREATE PROCEDURE MG017.SPSQN006
(IN  EMPLOYEE_NUMBER  CHAR(8),
 IN  RATE             DECIMAL(6,2))
LANGUAGE SQL
MODIFIES SQL DATA
DYNAMIC RESULT SETS 0
DISABLE DEBUG MODE
PACKAGE OWNER  RDAMCG
ASUTIME NO LIMIT
COMMIT ON RETURN YES
```

id

parms (SYSPARMS)

options (SYSROUTINES)

Each procedure is identified by name. The schema part of the name may be built as unqualified but will be stored in the catalog. When we get into native procedures, we will see that Version also plays a role in identifying the procedure.

Procedures may or may not have parms passing to and from it. The parms are described in the Parm Section as to datatype, length, ccsid, subtype, etc. They are also identified as going into the procedure, coming from or out of the procedure or both. The parms are stored in SYSIBM.SYSPARMS

There are numerous options which a procedure can have. The options are stored in SYSIBM.SYSROUTINES (and possibly in SYSIBM.SYSENVIRONMENT)

Anatomy of a Procedure part 2

```
UPDATE MG017.TG01SS
SET SALARY = SALARY * RATE
WHERE EMPNO = EMPLOYEE_NUMBER;
```

procedure body

```
COMMENT ON PROCEDURE MG017.SPSQN006
IS 'THIS IS A COMMENT';
```

comment

```
GRANT EXECUTE ON PROCEDURE MG017.SPSQN006
TO  JOHNNY;
```

grant (SYSROUTINEAUTH)

Some procedures have the text or body of the procedure defined in the 'Create Procedure' syntax. The two SQL procedures have such text – External SQL and Native SQL.

A procedure may have a comment just as most other DB2 objects.

You can grant 'EXECUTE' authority on a procedure.

Agenda

- History of Stored Procedures supported by DB2
- Anatomy of Stored Procedures
- The 3 types of Stored Procedures
- Advantages of one type over another
- Native SQL: Deployment and Schema Management

Three types of Stored Procedures

- External
- External SQL
- Native SQL

These are the three types of procedures. Of course, each will be discussed in more detail.

One interesting fact to hang onto is this: For procedure types with 'External' in their name, the executable code is 'external' to DB2. For procedure types with SQL in their name, the scripting language used is DB2's SPL (SQL Procedural Language).

External Procedure – example

```
CREATE PROCEDURE ACMQA01.SPOP0001
     ( IN          CODE          CHAR (3)
      ,INOUT  DESCRIP     CHAR(10))
LANGUAGE C
EXTERNAL NAME SPOP1
PARAMETER STYLE DB2SQL
ASUTIME NO LIMIT
PROGRAM TYPE MAIN
COMMIT ON RETURN NO;
```

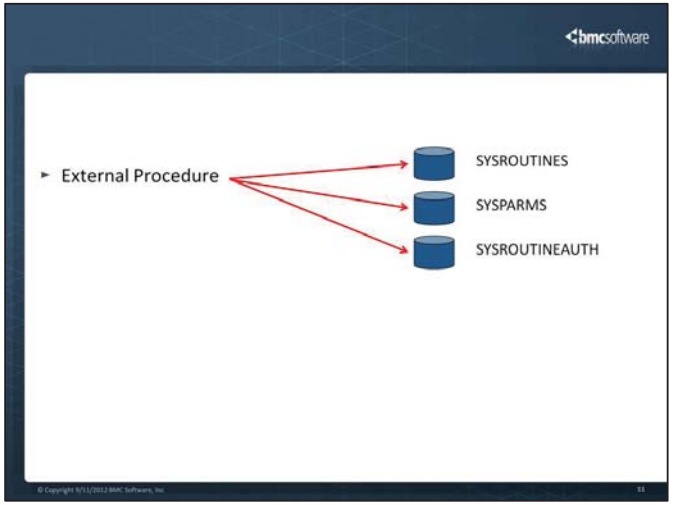
Language

Load Module Name

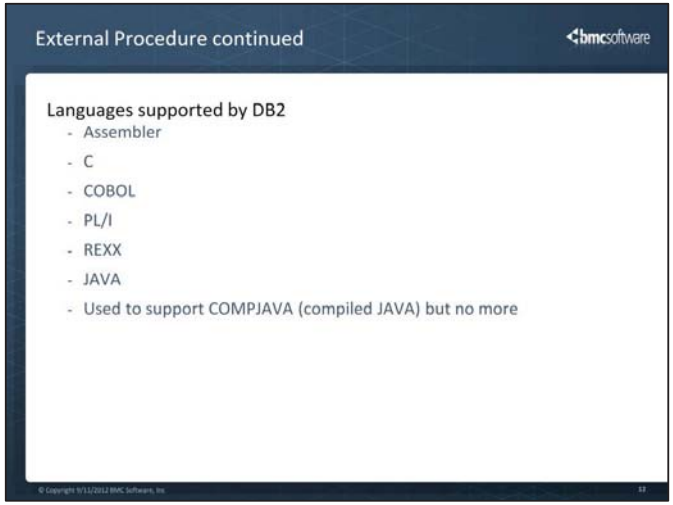
No procedure body

Note the Load Module name. This points directly to the Load Module of the C, COBOL (or whatever) program.

Note there is no procedure body or text in an External procedure. The procedure body is contained in an external library – hence the name – External.



When issuing the create procedure syntax for external procedures, rows are inserted into catalog tables SYSROUTINES and SYSPARMS. Also, a row is inserted into SYSROUTINEAUTH (grantor and grantee (in SYSROUTINEAUTH) are the same and match to owner (in SYSROUTINES). Grants will add rows into SYSROUTINEAUTH.



The majority of procedures that our customers have told us they are using are based on COBOL and JAVA.

How to Create an External Procedure

- Create a source module in a supported language
- Pre-compile, Compile and Link the source, creating a Load Module and DBRM member
- Bind the DBRM member into a package (if contains SQL)
- Execute 'Create Procedure' DDL

A stored procedure does not have to contain SQL. Some users have Stored procedures perform completely non-DB2 related tasks for them.

External Stored Procedures are often used to perform IO that cannot be performed via SQL.

Note: The executable load module called by an External Stored Procedure can contain multiple objects modules, it can also invoke other load modules directly.

External SQL Procedure - example

```
CREATE PROCEDURE ACMQAX9.SPIBM002
(IN  EMPNUM CHAR (6) FOR SBCS DATA CCSID EBCDIC
,OUT OMSG   CHAR (20) FOR SBCS DATA CCSID EBCDIC
)
      DYNAMIC RESULT SET 0
      LANGUAGE SQL
      EXTERNAL NAME 'SPIBM002'
      WLM ENVIRONMENT WLMENV1
```

Language

continued

The External SQL procedure has parms and options but must have the language of SQL.

Column EXTERNAL_NAME in SYSROUTINES is for external and external SQL procedures only.

External Name may be optional. If you don't have an external name, DB2 defaults the name and stores that name in EXTERNAL_NAME.

If external name option is not specified, the name of the procedure has to be 8 characters or less and that will be the external name.

Default for external procedure is procedure name; default for external SQL is DB2 controlled.

For external procedures the external name is the load module name.

Must be 8 characters or less (for external name)

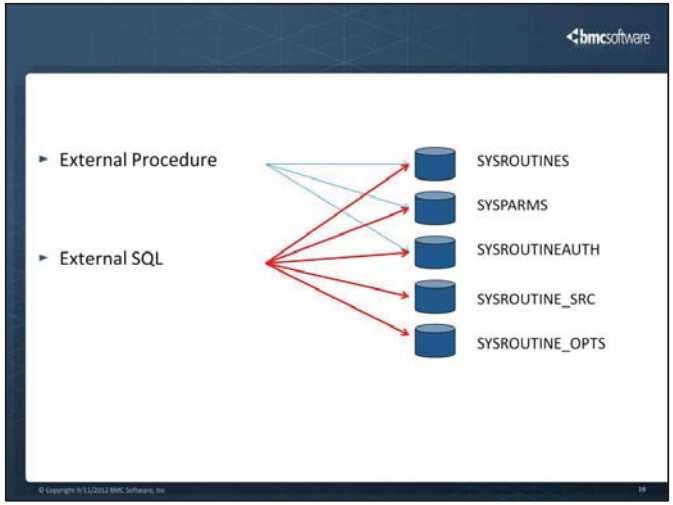
External SQL Procedure – example continued

```
BEGIN
  DECLARE NOT_FOUND CONDITION FOR SQLSTATE '02000';
  UPDATE DB2RES1.EMPLOYEE
  SET SALARY = SALARY * 1.03, BONUS = 0
  WHERE EMPNO = EMPNUM;
END
```

procedure body

One of the main differences between an External and an External SQL procedure is that the body or text of the procedure is contained within the Create Procedure DDL syntax.

The procedure body is SPL which contains SQL.



When issuing the create procedure syntax for external SQL procedures, rows are inserted into the same catalog tables as external procedures plus two additional tables. Again, once the external SQL procedure has been granted access, rows will be in SYSROUTINEAUTH.

SQL Procedural Language - SPL

- Both External SQL procedures and Native SQL procedures are written in SPL – DB2's scripting language.
- SPL is a topic beyond scope of this presentation
- SQL Reference Manual has an entire chapter devoted to it

The body of the External SQL procedure (which we just saw an example of) is written in SPL.

SQL Procedural Language - SPL

- SQL-control-statement: (from SQL Reference Manual)
  - *assignment-statement*
  - *CALL statement*
  - *CASE statement*
  - *compound-statement*
  - *FOR statement*
  - *GET DIAGNOSTICS statement*
  - *GOTO statement*
  - *IF statement*
  - *ITERATE statement*
  - *LEAVE statement*
  - *LOOP statement*
  - *REPEAT statement*
  - *RESIGNAL statement*
  - *RETURN statement*
  - *SIGNAL statement*
  - *WHILE statement*

As previously stated, we are not exploring SPL in this presentation. But we wanted to give you a taste of the functionality.

SQL-control-statement: (from SQL Reference Manual)
assignment-statement
CALL statement
CASE statement
compound-statement
FOR statement
GET DIAGNOSTICS statement
GOTO statement
IF statement
ITERATE statement
LEAVE statement
LOOP statement
REPEAT statement
RESIGNAL statement
RETURN statement
SIGNAL statement
WHILE statement

How to Create an External SQL Procedure

- Create your procedure using SPL
- SPL proc is input to IBM proc, SYSPROC.DSNTPSMP
  - DSNTPSMP is a IBM provided REXX external procedure
  - DSNTPSMP creates a 'C' program from the SPL
  - DSNTPSMP causes a precompile and compile of the C program
  - DSNTPSMP causes links and binds of the program
  - Options for precompile, compile etc need to be fed in

DSNTPSMP is a REXX program. You must be sure the REXX packages are bound on the ssid you are using which is part or installing DB2 REXX Language Support. See install guide for more information.

The C program is pretty much just a wrapper around your SPL procedure.

Feeding in the proper options for a compile, link, bind etc is an obvious requirement.

Following is some of the options used in external SQL procedures (in JCL form from the samples)

```
//LEOPTS   DD  *
//PCOPTS   DD  *
   SOURCE,XREF,MAR(1,80),STDSQL(NO)
//COPTS    DD  *
   SOURCE LIST MAR(1,80) NOSEQ LO RENT
//PLKDOPTS DD  *
//LKEDOPTS DD  *
   AMODE=31,RMODE=ANY,MAP,RENT
//BINDOPTS DD  *
   PACKAGE(DSN8ES61) MEMBER(DSN8ES2)
QUALIFIER(DSN8610) ACT(REP) ISO(CS)
```


How to Create an External SQL Procedure cont.

- DSNTPSMP creates and executes DDL to create the procedure in DB2
- 'Copy' of the source is stored in SYSROUTINES_SRC
  - Options for precompile, compile, link edit, bind get stored in SYSROUTINES_OPTS
- A WLM environment (Workload Manager) should be defined for the execution of DSNTPSMP.

Note that only a 'copy' of the source is stored in the DB2 catalog. The actual executable is stored in a LOAD library defined in the WLM environment.

The WLM environment has specific needs to execute DSNTPSMP. The source library, load library, dbrm library need to be defined at minimum.

Lots of problems with External SQL

- WLM environment difficult to build and maintain
- REXX packages had to be bound on ssid
- IBM would change DSNTPSMP causing JCL to stop working

- SOLUTION – Native SQL Procedures
- We believe most people are going to stop using External SQL procedures in favor of Native procedures.

WLM environments usually have to be built by SYSPROG's. Their requirements are not typically known to DBA's. The WLM application has to be active when DSNTPSMP is executed. Messages are written out to the WLM application outputs which DBA's may not typically be accustomed to reading.

The changes to DSNTPSMP might involve parm changes which caused the existing JCL set up to quit working. The changes occurred via a PTF, not just on DB2 release boundaries.

DB2 V9 introduces Native SQL procedures.

Agenda

- History of Stored Procedures supported by DB2
- Anatomy of Stored Procedures
- The 3 types of Stored Procedures
- Advantages of one type over another
- Native SQL: Deployment and Schema Management

Native SQL Procedures – advantages

- Use same SPL as External SQL procedures but MUCH easier to build
- No DSNTPSMP – hooray!
- No WLM environment – hooray!
- Not converted to C program; SPL text is executed interpretively
- Procedure body is stored in 2 mg CLOB column in SYSROUTINES

The SPL text is stored in SYSROUTINES and is more like an interpretive language vs. compiled program.

With External SPL, the text was stored in a separate catalog table and the build options used by DSNTPSMP were in a third table.

Now you can perform the build of the Native Procedure without having to have a IBM procedure defined and configured correctly to perform the actual build

Native SQL Procedures – MORE advantages

- Cost savings and performance advantages
  - Native procedures may use the ZIIP engine when invoked from the client
  - Native procedures run in DBM1 – especially good for data access
    - But not so good for procedural processes

The ZIIP offload is one of the driving factors behind the adoption of Native Procedures.

As more and more DB2 access is driven off the mainframe platform, being able to call a Native Stored Procedure from the client to access your DB2 data with the side benefit of having that CPU associated to the Native Proc running on a ZIIP engine.

There is no syntax to convert an External SQL proc to a Native proc but there is a third party vendor who can do it easily.

Native SQL Procedure - example

```
CREATE PROCEDURE MG017R.SPSQN047
(IN EMPLOYEE_NUMBER CHAR (8) FOR SBCS DATA
,OUT RATE DECIMAL (6,2)
)
      VERSION V1
        ...
BEGIN
  SELECT SALARY INTO RATE
  FROM MG017R.TG01SS
  WHERE EMPNO = EMPLOYEE_NUMBER;
END
```

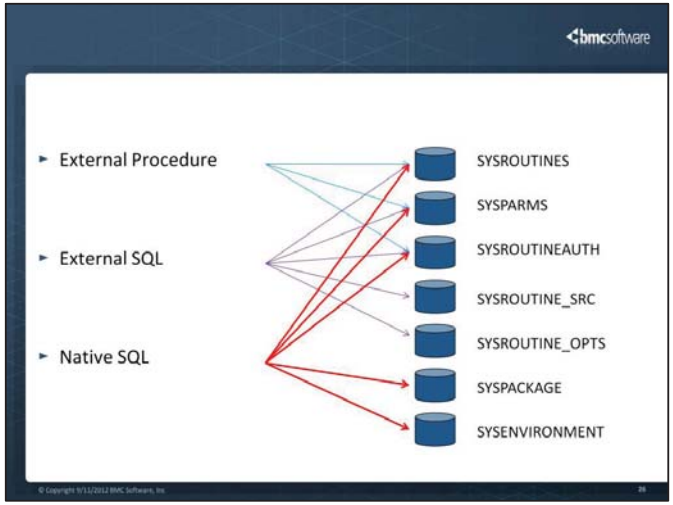
procedure body

The native procedure has an id, parms, options, and text or procedure body. The text is written in SPL. It can have an explicit version name. Versions will be discussed in the next slide.

VERSION is optional.

Looks very similar to External SQL procedure but much easier to build.

When the create procedure (Native SQL) is executed, a package will get build (tied to the SQL in the body).



When issuing the create procedure syntax for native SQL procedures, rows are inserted into the same catalog tables as external procedures plus two additional tables. Again, once the native SQL procedure has been granted access, rows will be in SYSROUTINEAUTH.

Agenda

- History of Stored Procedures supported by DB2
- Anatomy of Stored Procedures
- The 3 types of Stored Procedures
- Advantages of one type over another
- Native SQL: Deployment and Schema Management

Native SQL Procedures – Versions

- One procedure can have many versions.
- New versions are created using the ALTER statement

```
CREATE PROCEDURE Myproc …

ALTER PROCEDURE Myproc ADD
        VERSION V2 …

ALTER PROCEDURE Myproc ADD
        VERSION V3 …
```

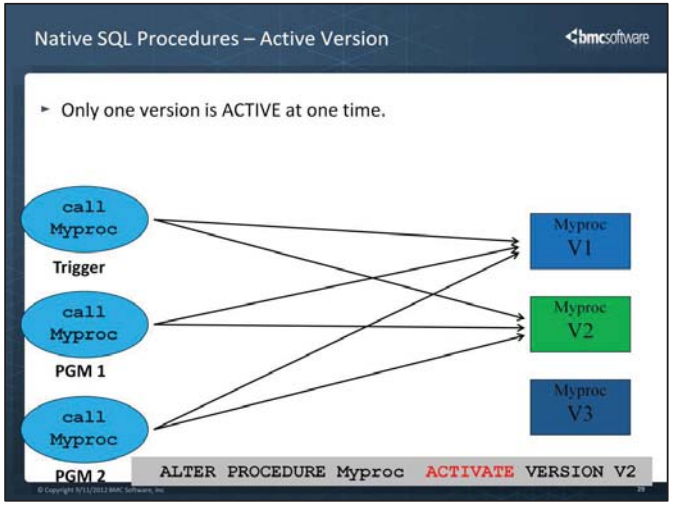
Myproc V1

Myproc V2

Myproc V3

You can name the first version or let it default to V1. Successive versions must be explicitly named.

You can theoretically maintain an unlimited number of versions for a Native Store Procedure. But only one can be active.



DB2 allows ALTERs to be directed against the ACTIVE version of a Native Stored Procedure, without having to know the active version name.

An Active Version can not be dropped via an ALTER DROP VERSION, either another version needs to be made active prior to the DROP VERSEION or the entire Native Procedure needs to be dropped via the DROP PROCEDURE.

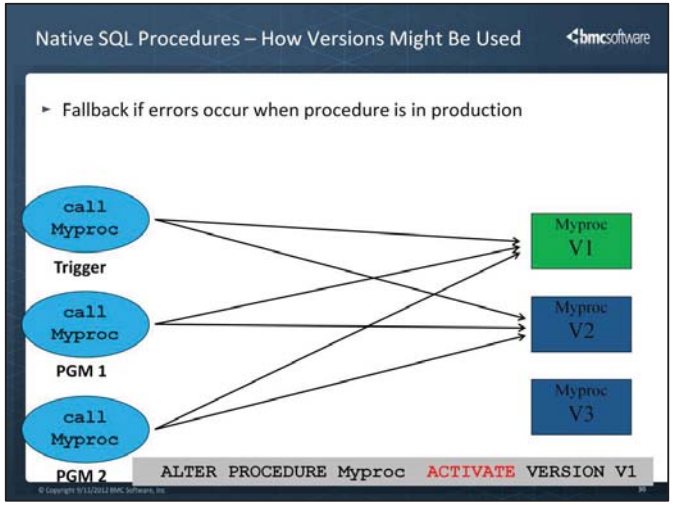
CREATE statement marks the original version as active and original stays active until another version is explicitly marked active
ALTER PROCEDURE Myproc ACTIVATE VERSION V2

Activating V2 causes V1 to be inactivate

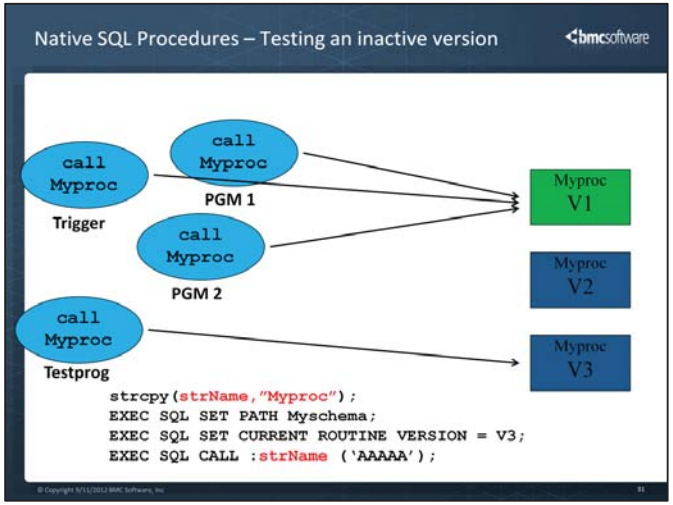
2 ALTER's needed (one to add version and one to activate version)

Note: the DROP PROCEDURE command will drop all versions of the Native Procedure.

There are some rebind implications of activate that will be discussed later.



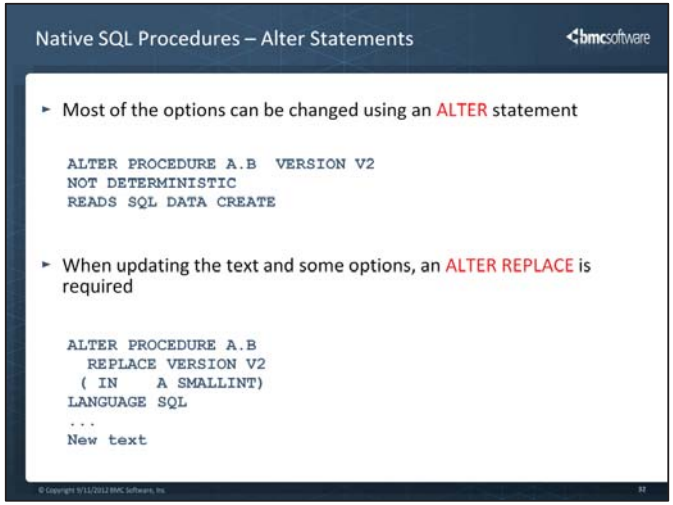
If you discover an error in the procedure when it is in production, all you need to do to fallback to the prior version is to activate the prior version (and you may need to rebind dependent packages – more discussion later).



There is no direct way via the CALL statement to explicitly execute a version of a Native Proc. But the SET ROUTINE VERSION can be used to achieve this via a special register. The schema must be set and the procedure name has to be in a host variable. Version could be hard coded or host variable.

This provides a fairly easy way to test out a new version of a Native Proc, without affecting any other users running against the Active version of the procedure on the subsystem.

We are not sure why you must use a host variable for the procedure name. Doing this was the only way we were able to get the SET CUIRRENT ROUTINE VERSION command to have any affect.



The first alter statement changes the value of the options Deterministic and SQL Data Access. It names V2 as the version to change. If version is left out, the Active version is changed. You may also specify ACTIVE VERSION in the syntax.

An Alter statement is legal for External and External SQL procedures as well as Native procedures.

The Alter Replace will replace all the parms, options and text. More discussion on this later. This is only valid for Native procedures.

Native SQL Procedures – Maintenance Considerations

- When you add a version (or alter replace a version), the owner is always the owner of the original version

```
CREATE PROCEDURE Myproc
ASULIMIT 1000 …
```

Myproc.V1
ASULIMIT 1000

```
ALTER PROCEDURE Myproc ADD
        VERSION V2 …
```

Myproc.V2
NO ASULIMIT

- When executing an Add Version or Alter Replace, the parms must be repeated in entirety.
- Challenges with Signature changes

If the options are not explicit in an Add Version or Alter Replace, the default for the option will be used. DB2 does not pick up the value from the existing version. It stores the default.

So if you created V1 of a Native proc with an ASULIMIT of 1000. Then you ALTER ADD Version V2, without explicitly stating an ASULIMIT, it will get the default of NO ASULIMIT. Not 1000 that the initial Native Proc was created with.

The last couple of slides have dealt with altering Native SQL procedures. Some attributes are not alterable. We will now discuss those options that are not alterable and what needs to be done to change them if they need to be changed.

Native SQL Procedures – Signature

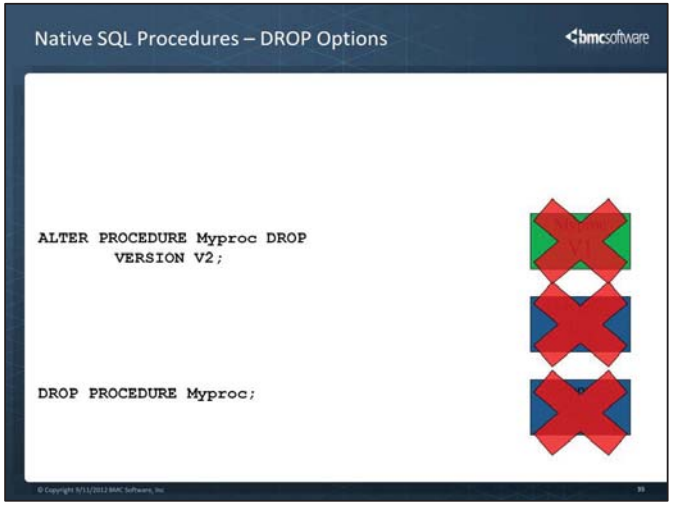
- What is a Signature? The following:
  - Schema name
  - Procedure name
  - Parms and their attributes (excluding name of parm)
  - Option, PARAMETER CCSID
- All versions of a native procedure have to have the same signature
- If you need to change part of the signature, you must change it in all versions. You must drop all versions (using drop procedure), then recreate all supported versions.

All versions of a procedure must have the same procedure signature. Therefore, each version of the procedure must have the same for the following items:
Schema and procedure name
Number of parms
Parm datatype/length/scale
Parm CCSID
parameter attributes for character data (FOR BIT/SBCS/MIXED DATA)
IN, OUT, INOUT options of parm
PARAMETER CCSID – which affects the CCSID of the parms -- more discussion on PARAMETER CCSID on next slide

To change a signature, you must drop all versions and rebuild them.

The name of a parm is not part of the signature. To change the name of a parm, you could issue an Alter Replace.

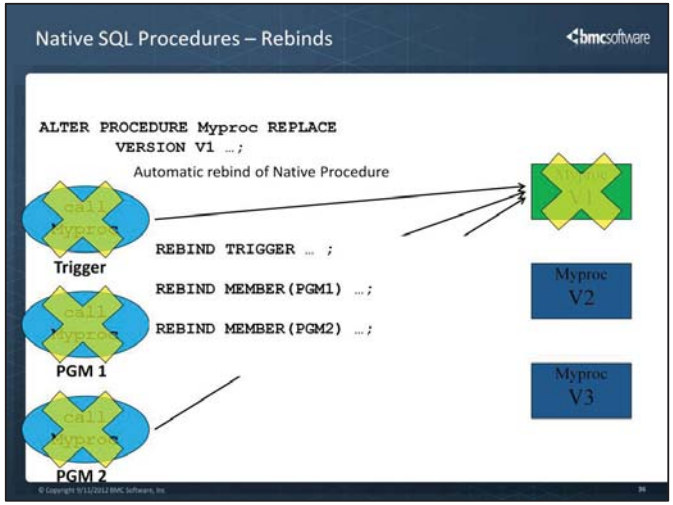
At some point, you may have many versions of a Native SQL procedure. If you need to change the signature, you will drop the procedure (which drops all versions). When you recreate the procedure, you most likely won't create all of the versions that you had before issuing the drop procedure (only create maybe the active version and any other "needed" versions).



Statement allows you to drop a single version of the procedure.

You cannot drop the version which is active

DROP PROCEDURE will drop all versions of a Native Procedure. Be carful if your intent is to only drop a single version.



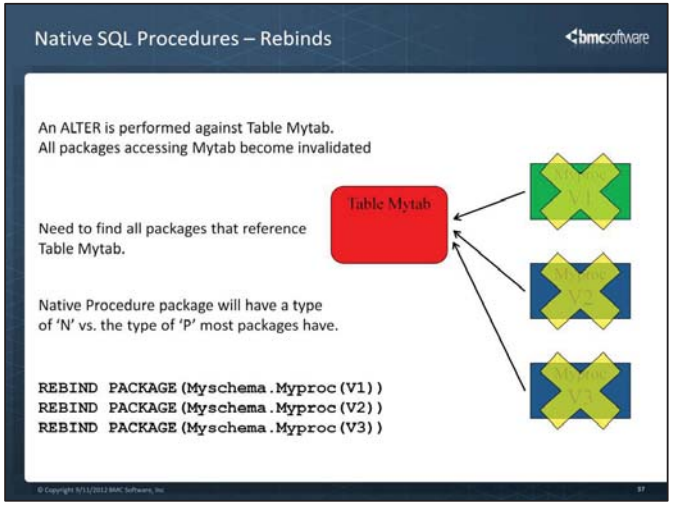
The yellow X's represent the package being invalidated.

If you alter the Active version of a procedure, you will probably have to rebind any dependent packages.

You can find packages that are dependent on your stored procedure by looking in SYSIBM.PACKDEP. BQUALIFER and BNAME will be the shema.name of the procedure and BTYPE will be 'O'.

The package for the native procedure will be automatically rebound if necessary.

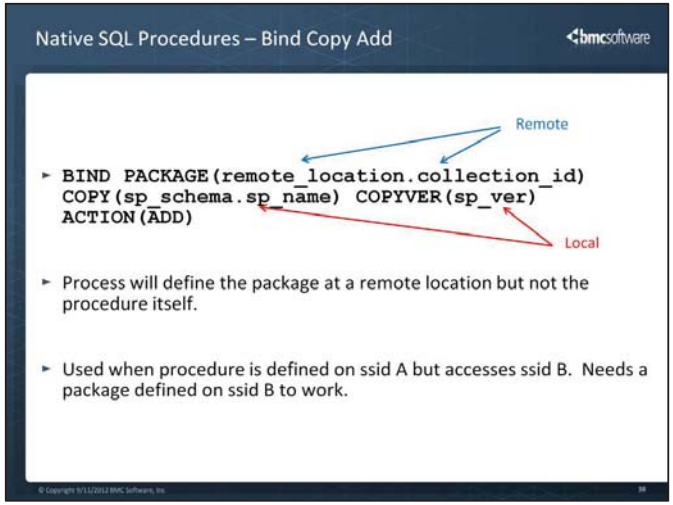
See SQL Reference, Alter Procedure (SQL – Native) table Create procedure and alter procedure options that result in rebind or regeneration when changed,



Your Native Stored Procedure package is just like any other package. If it references an object that is dropped, the package will be invalidated.

After you address the issues that invalidated the package, the procedure package will need to be rebound.

A second way to have rebinds is regenerate.



BIND PACKAGE(location.collectionid) COPY(sp_schema.sp_name) COPYVER(sp_ver) ACTION(ADD)

Note: The original package was of TYPE 'N', the new package will be a regular package. By looking at its definition in catalog you can't readily determine that the package is a copy of a Native Procedure package.

Native SQL Procedures – Bind Deploy

bmcsoftware

Remote

- **BIND PACKAGE (remote_location.new_schema)**

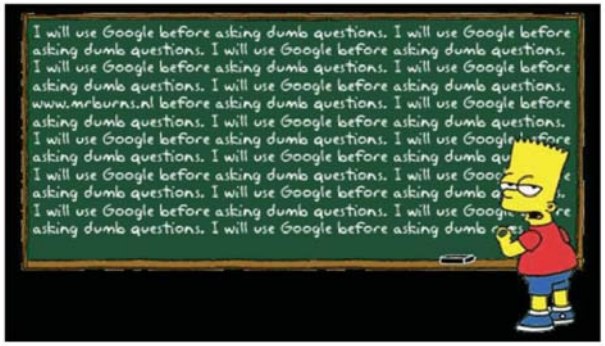
  DEPLOY (sp_schema.sp_name)
  COPYVER(sp_version)
  ACTION (ADD) QUALIFIER(new_qualifier)

Local

- The command will take a local procedure and define it on remote DDF location.
- Viable way to move procedures to production

Questions

bmcsoftware




bmcsoftware
BUSINESS RUNS ON I.T.
I.T. RUNS ON BMC™

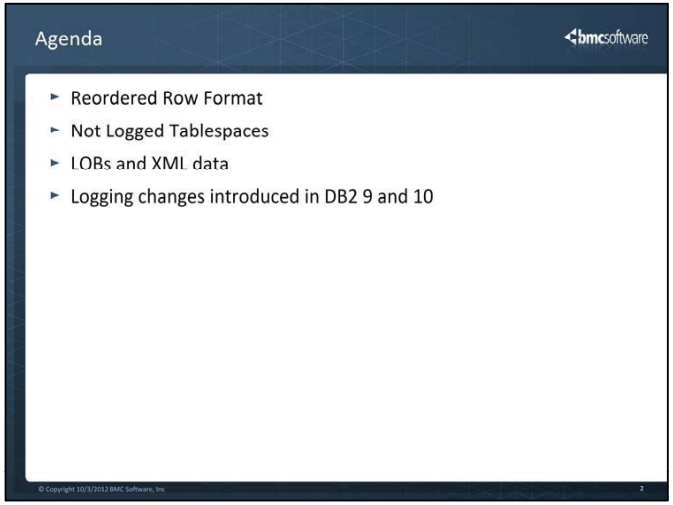
SQLADRIA SEMINAR
Zagreb, 28 September 2012

Recent changes in DB2 for z/OS logging

Steve Thomas
BMC Software

How well DB2 handles Log data can be a critical factor in maximizing performance, particularly in a high transaction rate environment. There have been a number of changes in the area of Logging over recent DB2 releases. Some of these have been obvious improvements such as the ability to use DFSMS Striping for Active Log datasets, but there have been Logging implications in some other changes such as the introduction of Reordered Row Format. However, many sites I visit have not really investigated these changes in any detail and the impact they might have on their operations. This presentation provides an update on changes to how DB2 manages logging updates and some useful hints and tips you will be able to take away and use to review your logging environment.

Steve is a Principal Consultant at BMC Software, and supports customers across the UK, Northern Europe, Greece, Turkey and Israel. He has been a DB2 specialist since 1989 and an IBM Information Champion since 2009. Steve has presented on a wide range of topics at events across the UK and in Europe. He helps organize the UK DB2 RUG and has been a member of the IDUG EMEA Conference Planning Committee for the past 7 years.



What data is logged during an Update?

- Assuming Data Capture Changes is not used
- Fixed Data - from first to last updated byte
  - Also true for Variable data if the length does not change
- Variable Data – from first updated byte to end of row
  - Bearing in mind the first updated byte will be the length field
- Leads to long standing good design practice
  - Locate Variable Columns at the end of the row
  - With the most updated ones towards the back end
  - Locate heavily updated fixed length columns together
- Usually only implemented when table first created
  - Rules tend to get relaxed as columns are added

Reordered Row Format (RRF)

- New Row format introduced in DB2 9 NFM
- Automatically implements some of these best practices
- Columns internally reordered on the data page
- All Variable columns stored at the end of the row
- No lengths – uses offsets into row instead
  - Both offsets and lengths are 2 bytes so row length identical
  - Row header is not included in the offset
- Logical Column location in the table unaffected
- No impact on Applications – not even a Rebind

Basic Row Format (BRF)

STAFF_NUMBER	INTEGER	NOT NULL
FIRST_NAME	VARCHAR(30)	NOT NULL
MIDDLE_INITIAL	CHAR(1)	NOT NULL
LAST_NAME	VARCHAR(30)	NOT NULL
DATE_JOINED	DATE	NOT NULL

001	05	STEVE	R	06	THOMAS	12/07/1999

2 byte Lengths (in hex)

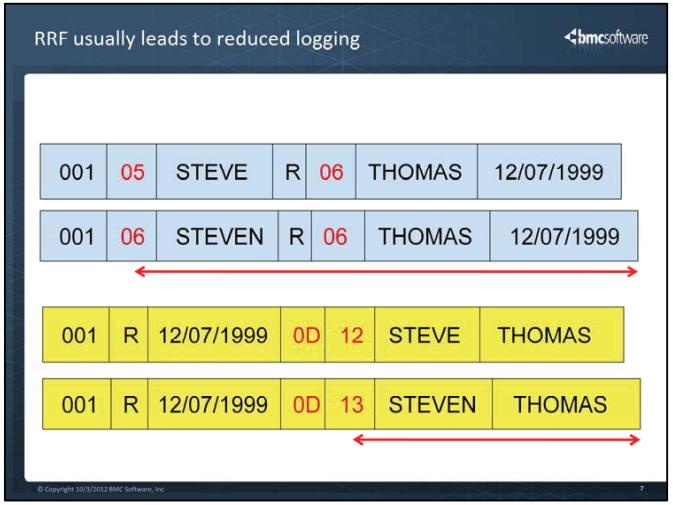
Let's look at our row in Reordered Row Format

STAFF_NUMBER	INTEGER	NOT NULL
FIRST_NAME	VARCHAR(30)	NOT NULL
MIDDLE_INITIAL	CHAR(1)	NOT NULL
LAST_NAME	VARCHAR(30)	NOT NULL
DATE_JOINED	DATE	NOT NULL

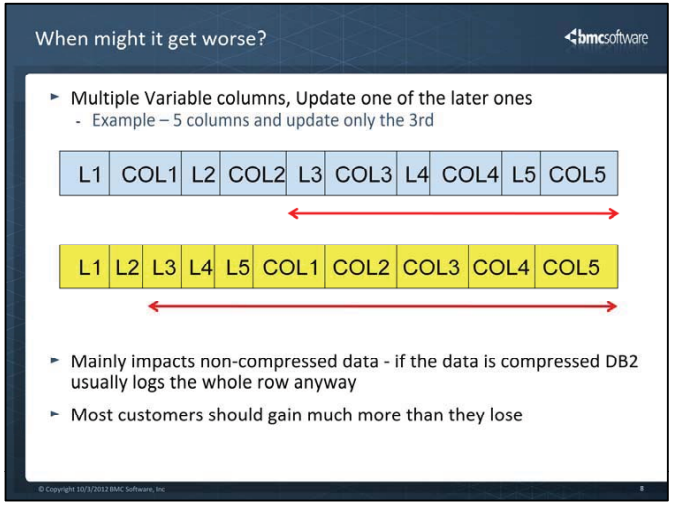
001	R	12/07/1999	0D	12	STEVE	THOMAS

2 byte Offsets (hex)

Note: Integer and Date columns both use 4 bytes of internal storage



In this update I've changed my first name from the shortened Steve to my full name of Steven



Test Results verify this

- What we did…
  - Created a table with 10 columns
    - 7 were VARCHAR of various sizes
  - Loaded 1,000 rows using both BRF and RRF
  - Updated the various columns on every row
  - Measured the results using a Log Analysis tool
- In most cases log volume between flat and 50% lower
  - Exception was updating the last columns
  - Worst case example we managed to double the log volume
    - Extreme example designed to cause maximum pain
- Conclusion: RRF should help in most cases
  - Beware situations with multiple (short) VARCHAR columns

Not Logged Tablespaces

- Added to provide an SQL equivalent to LOG NO utilities
  - Reduce Logging volume when data can be rebuilt
  - Never intended to be a performance option
- Prevents Logging of UNDO and REDO records
  - Control records are still logged (OPEN, CLOSE etc.)
  - So are Open URID records
    - Needed for Data Sharing and Long running URID messages
- Cannot be set for Catalog, XML or Workfiles
- Ideal applications include MQTs & Summary objects
- Default is always to Log updates
  - Except for Workfile tablespaces

Not Logged Tablespaces

- Incompatible with Data Capture Changes
- No SYSLGRNX entries created
- No SHRLEVEL CHANGE Copy or Reorg
  - Must use SHRLEVEL REFERENCE
- QUIESCE WRITE (YES)
  - Issues a Drain, Flushes the Bufferpool as normal
  - But no SYSCOPY row is created
- When you Update a NOT LOGGED object
  - Status is changed to ICOPY (non restrictive)

-CANCEL THREAD and Aborted URIDs

- Cancelling a thread on a NOT LOGGED object
  - Leaves the object in LPL
  - And in RECP, RBDP or AUXW depending on the type
  - Consider using –CANCEL THREAD..... NOBACKOUT
    - But remember the Logged objects as well!
- LPL can only be removed manually
  - No Automatic LPL recovery
  - Recover, Refresh (MQT), Load or Drop/Recreate
  - Can also issue DELETE without WHERE or TRUNCATE
- Same consideration applies to ROLLBACK
  - Except for LOB objects – see later
- Ensure no Duplicate Key or RI violations

Linking Objects

- NOT LOGGED usually managed via Base Tablespace
  - This is called being LINKED
- XML and Indexes are always Linked
- LOB data can become Unlinked
  - Under certain circumstances
  - See next slide for a graphical example
- Look for LOG column in SYSIBM.SYSTABLESPACE
  - Base Tablespace is set to 'Y' or 'N'
  - Linked Objects are set to 'Y' or 'X'
  - LOB objects can be 'N' if they are Unlinked

Let's see an example

TS	Y	N	Y	Y	N
IX	Y	X	Y	Y	X
LOB	Y	X	Y	N	N
XML	Y	X	Y	Y	X
		TS Not Logged	**TS Logged**	**LOB Not Logged**	**TS Not Logged**

Potential Impact on Data Sharing

- NOT LOGGED data is not protected by the Log
  - So it needs to be externalized from Buffers quickly
- Accomplished by treating them differently
  - Read Only Check times (PCLOSEN & PCLOSET) effectively set to 1
  - DB2 treats the object like it was Read Only
    - Buffers are Forced much faster
- Good for data integrity but may have DS impact
  - Objects switch GBP Dependency frequently if updated
- Another good reason to limit using this feature to objects that are genuinely read only

Large Objects (LOBs)

- LOB data is being used a lot more today
  - Including in the Catalog and Directory in DB2 10
  - Important that people understand how LOB logging works
- LOB data has always supported NOT LOGGED
  - Syntax changed from LOG YES/NO in DB2 9
  - Old syntax is still recognized and supported
- Old 1Gb restriction for LOB data logging lifted in DB2 9 for z/OS
- Some data is always logged
  - Even if LOG NO or NOT LOGGED used
  - System Pages Logged

How does LOB logging differ?

- Size of LOB data is the obvious consideration
- LOB Data in an Auxiliary TS is never updated in place
  - Old data marked as deleted and new data added
  - So the old data is still available if required
  - Known as Shadow Copy Recovery
- No Logging of UNDO records for LOB data
  - ROLLBACK recovery for LOB data not possible
- Also no Logging of LOB data for DELETEs
  - System Pages provide information to allow roll forward recovery
- Remember NOT LOGGED data is Forced at Commit
  - May impact Application response times

DB2 9 – Logging related changes

- Archive Log process now runs in 31 bit mode
  - This happens in the MSTR address space
  - Reduces risk of Storage related failures when running Archive Logs
  - Allows much larger Log Buffers to be used
  - Also attempts to fill next Buffer while processing previous one
- Active Log input Buffers increased from 15 to 120
  - Improves Fast Log Apply performance by up to 100%
- Archive Log input buffer increased
  - Now uses 10 tracks per Stripe

DB2 9 – More Logging related changes

bmcsoftware

- DASD Striping of Archive Logs
  - Changed from BDAM to BSAM access
  - Allows other features of DFSMS Extended Format (EF) datasets
  - For example Compression
    - May be a viable alternative to striping
  - See DB2 10 improvements which may also affect this decision
- Extended BSDS became compulsory
  - Supports 10,000 Archive and 98 Active Log pairs
- All LOB Tablespaces can be Logged
  - Limited to <1Gb LOBs in previous releases

DB2 9 – Extended Volume Support

bmcsoftware

- Extended Volumes are a z/OS feature
  - Introduced between z/OS 1.10 and 1.12
  - Supports Volumes > 65,520 Cylinders (Model 54's)
- Exploited by DB2 9 for z/OS
- Archive Logs can now use DSNTYPE=LARGE
  - Prevents multi-volume or tape based archives
  - These were required if you were using 4Gb Active Logs
- Extended Volumes split into Base & Extended Addressing space (EAS)
  - PK58292 allowed BSDS & Active Logs to reside in EAS
  - Note space in EAS is allocated in 21 cylinder chunks

DB2 9 and 10 - LRSN Spin Avoidance

- Before DB2 9 each Log record required a unique LRSN
  - Could result in DB2 waiting until new LRSN available
  - Each LRSN represents around 16 microseconds
  - High CPU overhead and Log Latch Contention
- DB2 9 NFM allowed duplicate LRSNs
  - Must be successive Log records from the same member
  - Must be for different pages (commonly data and index pages)
  - Consecutive Log records for same page require unique LRSN
- DB2 10 NFM extends this capability
  - Consecutive Inserts for the same data page can share LRSN
  - Does not apply to Updates, Deletes or same index page
- Very useful for multi-row Inserts with no/few indexes

DB2 10 – Latch Class 19

- Critical for data consistency
  - Used to Serialize updates
- Single latch per subsystem used before DB2 10:
  1. Latch obtained
  2. RBA range reserved
  3. Log record moved into the Log Buffer
  4. Latch released
- Now Latch is released once Buffer space reserved
  - Multiple log records can be written in parallel
  - Improves potential logging rates
  - Available in CM Mode

DB2 10 – Log Buffers

- Log Buffers are now fixed in Storage
  - DSNZARM OUTBUFF – now defaults to 4Mb (was 400Kb)
- 4Mb should be enough for most sites
  - IFCID 001 can indicate when your value may be too small
    - QJSTWTB – Log writes waits due to no available log buffers
  - Care required as all storage defined is now fixed
- Potential reasons to increase this include:
  - Improved ROLLBACK – hopefully not relevant!
  - CONSISTENT COPY
  - Products that use IFCID306 log reading interface
    - E.g. DB2 Data Propagator for z/OS

DB2 10 – Log I/O Enhancements

- More log writes are asynchronous and in parallel
  - In DB2 9 DB2 performs re-writes of each Log CI serially
    - Typically at COMMIT (Forced Writes)
    - Avoided risk of data loss in the event of dual log I/O failures
    - But led to more waits for Log Writes at Commit
  - DB2 10 allows these to be performed in Parallel
    - Change made possible by improvements in Disk technology
    - No loss of data integrity
- Hardware and Channel Technology improvements
  - DS8800 Disks
  - High Performance FICON (HPF) I/O protocols
    - Can be used for Log I/O >64Kb on a z196

DB2 10 I/O Performance

- Taken together these changes make a big difference
  - Redbook reports 50% reduction in Log suspend time for 3 page Synchronous Writes (e.g. at Commit)
    - Absolute difference maintained for more pages
    - The relative Percentage savings are lower
- Maximum Log throughput now reported at up to 180Mb/sec
  - Compared to around 100Mb/sec on a DS8300 using DB2 9
- May prevent the need to use Log Striping?

DB2 10 - Dynamically adding an Active Log

- Before DB2 10 had to use DSNJU003
  - Requires you to stop DB2
- New –SET LOG NEWLOG syntax added
  - Available in CM mode
  - Remember to add both Active log pairs
  - This is a permanent change
- When might you want to do this?
  - Archive Log fails or hangs and you're running out of active logs
    - DB2 will stop if there no Active Logs available
  - If Log fills and you need to shut down DB2
    - Active Logs required to perform shutdown
- Look for message DSNJ110E if archive log waits
  - -ARCHIVE LOG CANCEL OFFLOAD may be required
    - Cancels hanging log offload and starts the process again

DB2 10 - Improved Checkpoint Flexibility

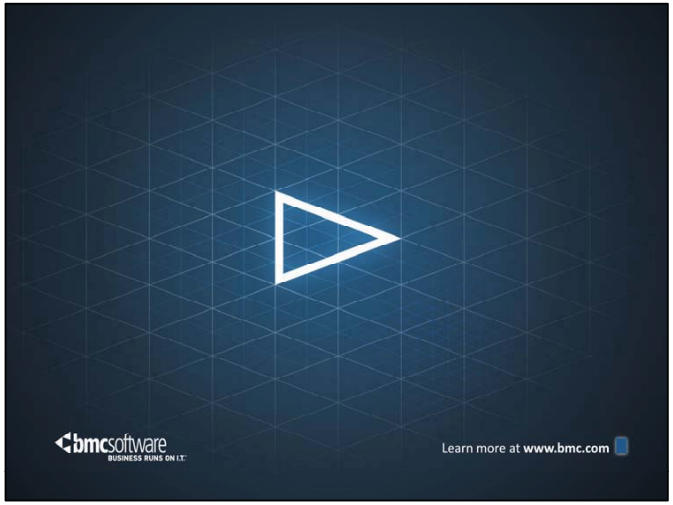
- DB2 9 checkpoints using Time or Number of log records
- Neither is ideal is all circumstances
- Time is usually best choice for most customers
  - But at busiest times there may be too many log records
  - This elongates subsystems restart time
- DB2 10 can checkpoint on either value
  - Set DSNZPARM CHKTYPE=BOTH
  - And provide values for both CHKFREQ and CHKLOGR
- Setting can be changed using –SET LOG command

Other DB2 10 changes

- Must use extended format BSDS
  - Supports 10,000 Archive and 98 Active Log pairs
  - Introduced in DB2 V8 but not compulsory
  - Will have been done if you're migrating from DB2 9
  - Install CLIST does this if it hasn't been done already
    - But doesn't resize the BSDS
  - If migrating from V8 then you need to check DSNJU004
    - Look for the DSNJCNVB job message
    - Better option to track down a V8 Install Guide
- Log Apply Storage default changed to 500Mb
  - DSNZPARM LOGAPSTG – used to be 100Mb
  - Introduced by PM31641 after GA date

Significant DB2 10 Logging related PTFs

- BACKOUT Recovery can struggle with Mass Deletes
  - Especially if no Data Capture Changes
  - PM30991 (Mar 2011) added SYSCOPY record to record this
    - Using TYPE=L STYPE=M
  - Customers started getting Lock Escalations
  - Hiper PM52724 (July 2012) corrected this situation
    - Now adds a Diagnostic Log record for UTS or Segmented TS



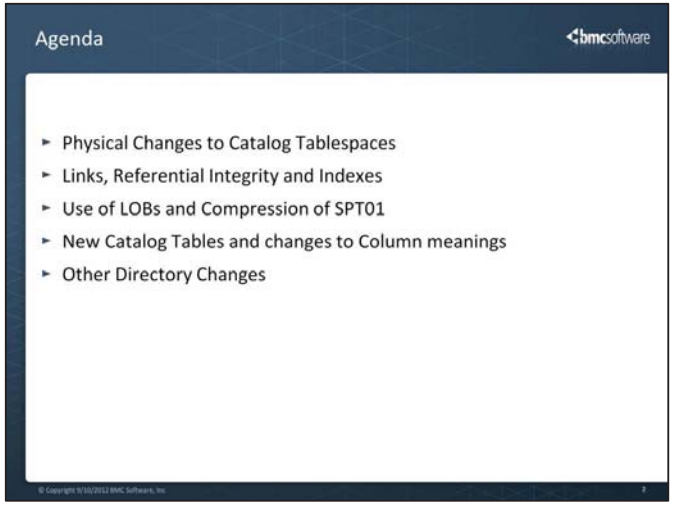
SQLADRIA SEMINAR
Zagreb, 28 September 2012

A new catalog and directory structure in DB2 for z/OS

Steve Thomas
BMC Software

For the second time in three releases of DB2 for z/OS a larger number of changes than normal have been made to the Catalog and Directory structure. At first sight the introduction of Unicode and Long Names along with DB2 V8 might appear to be more fundamental, but the improvements made in DB2 10 for z/OS are likely to yield equally important benefits. Chief among the changes are introduction of UTS objects and the removal of the internal links, but that's not all. We will soon be able to issue SQL against some Directory objects, and the introduction of many more LOBs, some of which can be Inline, will affect the way we need to manage the Catalog. These changes are very significant in their own right and taken together are probably the second most wide ranging (at least) since DB2 was first released back in the mid-1980's. This session describes all the changes that have been made, their benefits and how they will impact the way we need to manage the Catalog and Directory moving forwards.

Steve is a Principal Consultant at BMC Software, based in the UK. He has worked with DB2 since Version 1.3 in 1989 and has been an IBM Champion for Information Management since 2009. Steve is a well known speaker and has presented on a wide range of topics at events all across Europe. He has been a member of the IDUG EMEA Conference Planning Committee for the past 7 years and also helps organize the UK local DB2 Regional User Group.



1. The physical changes to the Catalog which include the conversion of several key tablespaces to single table UTS Partition by Growth objects.
2. The impact of removing the Internal links - hopefully most customers will benefit but there are also some things to be aware of from a performance perspective.
3. The increased use of LOB objects, particularly in the Directory, and what this means for customers who have been using Plan Stability and compressing the SPT01 tablespace in DB2 9 for z/OS. An APAR released since GA date provides several options here, so understanding what can and can't be done is important.
4. An overview of the new Catalog tables introduced to support new features such as Plan Stability and Pending Alters, along with the more significant changes in meanings for existing columns. I won't have time to cover all the changes here, so the main intention is to highlight some of the more important differences so you'll know where to look for more information.
5. Other Changes to the Directory in DB2 10 for z/OS in addition to the LOB and SPT01 ones mentioned earlier

Catalog gets more complicated every release

DB2 Version	Table spaces	Tables	Indexes	Columns	Table check constraints
V1	11	25	27	269	N/A
V3	11	43	44	584	N/A
V5	12	54	62	731	46
V6	15	65	93	987	59
V7	20	84	117	1212	105
V8	21	87	128	1286	105
DB2 9	28	106	151	1668	119
DB2 10	81 (88-7)	124	195	1701	119
Now	*95*	*134*	*233*	*1982*	*119*

- Excludes objects later added to Catalog such as RTS or XSR

Source: IBM Redbook – DB2 10 for z/OS Technical Overview SG24-7892

3

This table, taken from the Technical Overview Redbook shows how the Catalog has been expanding consistently with every release. The numbers exclude items such as RTS and XML Schema registration objects before they became part of the main Catalog. I remember when I would have had a fair chance of naming almost every table in the Catalog from memory, but those days have long gone now!

Before anyone lets me know the actual numbers on your own DB2 10 system are likely to be slightly different than those reported here which is why I added the row labelled "now" highlighted in green to show how the situation has changed even since the Redbook was published. This is another good reason why you should keep up to date with maintenance in order to obtain all the new features that continue to be added after GA date.

As you can see, although DB2 V8 provided some huge changes such as conversion to Unicode and Long names, the actual number of tables didn't increase very much, whereas Versions 9 and 10 have added around 20 and 30 respectively. What has changed a lot in DB2 10 is the number of tablespaces – in fact it's more than trebled, primarily due to the introduction of UTS objects and having more Auxiliary objects to store the new LOB columns which we'll discuss later.

Catalog Tablespace breakdown (Nov 2011)

- 55 Partition by Growth UTS
- 19 Segmented tablespaces
  - Of which 12 are still multi-table objects
    - Containing data about Trusted Contexts & Roles, Sequences, DDF, Stats History, Routines, RTS, Check Constraints, Java & XML
- 3 Simple tablespaces
  - SYSCOPY, SYSGPAUTH and SYSUSER
- 18 Auxiliary tablespaces (for LOBs)
- 1 uses ASCII (SYSTSASC)
- 2 use EDBCDIC (SYSCOPY and SYSEBCDC)

This slide provides a more detailed breakdown of the Tablespaces on my own DB2 10 subsystem. As you can see the majority of the objects are now Partition by Growth, many of which relate to constructs used in DDL and Bind operations, such as Databases, Tables, Indexes, Plans and Packages. There are also 18 Auxiliary Tablespaces used to store LOB columns. However, based on this data DB2 10 looks like an intermediate staging post in a longer term plan to convert the Catalog to more up to date tablespace technology. For example 12 of the remaining 19 Segmented Tablespaces still support multiple tables. There are also 3 simple tablespaces despite these being deprecated although the better news is that they all contain only one table. SYSCOPY fairly obviously contains SYSIBM.SYSCOPY, while SYSGPAUTH and SYSUSER contain SYSIBM.SYSRESAUTH and SYSIBM.SYSUSERAUTH respectively.
The vast majority of the objects now use Unicode, introduced in DB2 V8, but there are 2 EBCDIC and 1 ASCII objects. Two of these support the EBCDIC and ASCII versions of SYSDUMMY1, but the other one is SYSCOPY. This looks to be a likely candidate to be changed in a future release as it's also a Simple Tablespace and is often one of the larger Catalog Tables in terms of number of rows.

Physical Changes to Tablespaces

- Finally starting to lose the Simple Tablespaces ☺
- Many Catalog and Directory tablespaces now UTS
  - Partition By Growth (MAXPART 1 and SEGSIZE=32)
  - DSSIZE 64Gb
  - SMS Managed
  - Row level locking
  - Automatic Size Management (PRIQTY & SECQTY = -1)
  - RRF – Reordered Row Format
  - Page size varies but mostly 4Kb

The first thing we're going to discuss are the physical changes being made to the Catalog and Directory Tablespaces. With DB2 10 we're finally beginning to lose the simple tablespaces that have been used since DB2 Version 1.1 back in the mid-1980's. Almost all the new Tablespaces being added in this release are Universal Tablespace Partition by Growth objects. Most of these contain the tables that used to be stored in a number of key multi-table tablespaces in previous releases, primarily those handling DDL and Bind activity.

The first significant points to note is that because the UTS tablespaces are Large or extended objects with a DSSIZE of 64Gb, they have to be SMS managed rather than being User defined. The UTS tablespaces also use Reordered Row Format and automatic size management, with both Primary and Secondary quantities set to -1. Quite apart from anything else this tells you that these features have now become mainstream as far as IBM are concerned, which should hopefully reduce any concern about using them for your own User data.

What objects now use UTS?

- Data from 7 existing multi-table tablespaces
  - Significantly those that impact DDL...
    - SYSDBASE, SYSDBAUT, SYSGROUP, SYSOBJ, SYSVIEWS
  - ...and Bind activity
    - SYSPKAGE, SYSPLAN
  - Comprising 43 tables
    - 2 have gone – SYSVTREE and SYSVLTREE
    - These existed in DB2 9 even though they weren't listed in Manuals
    - Data stored in a BLOB associated with SYSIBM.SYSVIEWS
- 12 new tables use PBG objects
  - Supporting new features such as Audit Policies, Autonomics, Column and Row Masking, Deferred Alter and Plan Stability
  - See later section for more details

From an existing user's perspective the most interesting data stored in these PBG objects is from 43 tables that were stored in 7 multi-table Tablespaces in previous releases. These tables store data related primarily to DDL and Bind activity and placing them in single-table PBG tablespaces along with Row level locking, along with the other changes we'll discuss should provide a significant degree of relief from the Catalog Contention that used to occur if a lot of DDL and/or Bind activity was executed in parallel. Note that 2 tables from these Tablespaces, SYSVTREE and SYSVLTREE no longer exist in DB2 10 which used to store internal data on Views. They still existed in DB2 9 even though they weren't listed in the Catalog list in Appendix A of the SQL reference Guide.

The remaining 12 tablespaces/tables store data related to various new functions introduced in DB2 10, as listed in the slide. It looks like the default for any new Catalog Tablespaces created from now onwards is going to be PBG objects.

SMS Management of Catalog/Directory

- No STOGROUP required for Catalog and Directory
- Controlled by 6 new DSNZPARMs
  - CATDDACL, CATDMGCL, CATDSTCL for Tablespaces
  - CATXDACL, CATXMGCL, CATXSTCL for Indexes
  - Required unless ACS routines allocate datasets as Extended
- No need to allocate new datasets manually
- Requires EA datasets if you haven't used them before
- User indexes don't need to be SMS-managed
  - Why would you not make the change at the same time?

The Catalog and Directory now being managed by DFSMS sounds simple but provides very useful benefits. There is still no STOGROUP defined for these objects. Instead, 6 new DSNZPARM settings provide Data, Management and Storage Classes for the Catalog Tablespaces and Indexes. What it does mean is that we no longer need to manually create new datasets ahead of time of a Catalog Object such as SPT01 is running out of space, or before you run a REORG against them. It's worth remembering that the Catalog now requires the use of EA or Extended datasets, so your Storage Team are going to have to be involved in the process, particularly as you'll no doubt need to ensure that the performance characteristics of these datasets are sufficient.
What is interesting is that there's no requirement for User Defined Indexes on the Catalog to be SMS Managed, but I can't see why you wouldn't choose to make this change at the same time for your own objects when the system is moving to use DFSMS.

When Conversion occurs

- DFSMS environment created by job DSNTIJSS
  - Run during Installation before you migrate to Compat Mode
- New objects are all created using SMS
  - 2 TS, 3 Tables, 2/3 columns and 3/4 indexes by DSNTIJTC
  - Remainder by DSNTIJEN
- DSNTIJEN uses REORG to convert the multi-table objects listed on Slide 6 to PBG
  - It also drops the old Tablespaces
- Review Backup & Recovery Strategy during Migration
- Remaining objects remain User Managed indefinitely
  - Converted as and when you REORG Catalog once in NFM

The DFSMS environment required to manage the Catalog and Directory is created by job DSNTIJSS which is run during installation but before the migration to Conversion Mode.
All the new Catalog objects created will be under the control of SMS. DSNTIJTC, which executes the CATMAINT migration to Conversion Mode, creates 2 new Tablespaces (SYSTSASC and SYSTSUNI) and 3 Tables. The additional table is SYSIBM.SYSDUMMYE, created in SYSEBCDC alongside the existing SYSDUMMY1 table which is retained for compatibility purposes. DSNTIJTC also creates around 2/3 of the new indexes and 3/4 of the new columns that are being added. The remaining new objects are created during the Migration to ENFM by job DSNTIJEN. This job also executes the REORGs required to convert the multi-table tablespaces referred to on Slide 6 into UTS objects with one table per tablespace, and then drops the old Tablespaces. These changes, together with the increased use of LOBs which we'll discuss shortly, mean that you may need to change your Catalog and Directory backup procedures several times during the DB2 10 migration process.
The remaining Catalog objects can remain User Managed for an indefinite period. They will be migrated to SMS Managed objects as and when you REORG the Catalog once you're in NFM.

Catalog Links

- The DB2 Catalog has always been slightly different!
  - Links or Pointers exist between Catalog Tables
- Originally for Performance & Integrity reasons?
  - No RI existed before DB2 Version 2.1
- For example, SYSIBM.SYSTABLES has links to:
  - SYSTABLES and SYSTABLESPACE (Ring)
  - SYSCOLUMNS
  - SYSINDEXES
  - SYSRELS
  - SYSSYNONYMS
  - SYSTABAUTH

Catalog Links have been with us since DB2 was first released. They are pointers between certain objects and I suspect were introduced for a mixture of Performance and Integrity reasons, particularly as DB2 didn't have RI until Version 2.1. I was working with it by then and can remember discussions over the next few years about not using RI for performance reasons in User data let alone in the Catalog. As a result the links became very embedded and difficult to remove, which is why they've been with us ever since.
The links are documented in the Diagnosis Guide which is a restricted Manual so I can't really list them all here, but as an example SYSIBM.SYSTABLES has forward and backward ring pointers (links) to it's parent Tablespace and the other tables contained within it, as well as links to the other child objects listed on the slide. This made finding these objects faster and it was also easier to ensure data integrity when adding or dropping objects.

Problems caused by the Links

- Primarily related to Lock Contention
  - Catalog Used Page Level locking
  - Multi-Table Tablespaces in Catalog exacerbated this
  - One Lock could and did cause significant issues
- Concurrent DDL Activity has always been a problem
  - Heavy BIND activity causes similar problems
  - Issues have increased as systems have become busier
- Also need to check the links are not broken
  - DSN1CHKR
  - DSNTESQ in SDSNSAMP

Removal of Links in DB2 10 for z/OS

- All Catalog and Directory Links have been removed
  - This occurs during migration to NFM
- Extra RI has been added to replace the Links
  - 25 new Referential constraints in all
  - Around 15 between existing objects
  - This also explains many of the 44 new indexes
- Indexes added to existing tables when converting to CM
- Definitely worth reviewing User Defined Catalog Indexes
  - Many were added because DB2 could use Links
  - But SQL and Third party tools could not

Performance Implications

- The old links were there for a reason
- More Indexes and RI to check and maintain
  - Several Tables now have 5 indexes
  - SYSTABLES involved in 15 RI Constraints, 14 as Parent
- The DB2 10 Performance Redbook quotes:
  - Class 2 CPU & Elapsed time both up by 20-30% for Full Bind
  - Single threaded DDL roughly equivalent in DB2 9 & DB2 10 CM9
  - In NFM Elapsed down by 4% and Class 2 CPU up by 10-40%
- Offset against this is much improved concurrency
  - 5 streams of DDL run in parallel executed in 30% of the elapsed time compared to running single threaded in DB2 9
- May need more Parallel BINDs to retain throughput

At most large sites I have worked with recently the main problem with the Catalog and Directory has been DDL and Bind concurrency, and the changes introduced in DB2 10 are aimed very much at this problem. However, there may be a price to pay in terms of CPU time because the links were there for a reason, even if they did cause concurrency problems. Updates to the DB2 Catalog now have more Indexes to maintain and RI to check, and the links were more efficient than Index based access. You can see the figures quoted in the Performance Redbook for CPU, but offset against this is an almost linear throughput benefit from running parallel DDL streams compared to single threaded DDL in DB2 9. Given that the CPU used by DDL and Bind doesn't typically tend to be the thing that's concerning customers the most then you have to think that DB2 10 has achieved it's aim in this regard.

LOB data in the DB2 Catalog

- 3 LOB columns in the Catalog in DB2 9
  - All where many would not encounter them very often
  - SYSROUTINES, SYSJAROBJECTS, SYSJARCONTENTS
- All that has changed in DB2 10
  - There are now 18 LOB columns in the Catalog ...
  - ... and several more in the Directory
- These include LOBs on some very common objects
  - SYSINDEXES, SYSVIEWS, SYSPACKSTMT
  - All have 2 LOB columns
  - Mixture of CLOB and LOB
- Several tables use INLINE LOBs
  - e.g. SYSPACKSTMT and SYSVIEWS

The bottom line here is that there is no escape this time!

What was the problem?

- Many Catalog columns need to store SQL Text
  - SYSINDEXES, SYSVIEWS, SYSTRIGGERS, SYSPACKSTMT
  - Package Text in SPT01
- Without LOBs the column length limit was <32Kb
  - SQL often broken up into chunks with Sequencing key
- SYSPACKSTMT can store both EBCDIC & Unicode data
  - Very hard to read or search
- Uses up a lot of space in the main Tablespace
  - Particularly SPT01 once Plan Stability was introduced
  - IBM allowed you to Compress SPT01 via a PTF in DB2 9
- Other Tables need to store large amounts of Binary data
  - DBD01 in the Directory being a prime example

Benefits of Using LOBS

- For the SQL Text using CLOBs:
  - Text can be merged back into a single column
    - The data is moved - old columns are retained but not used
  - EBCDIC and UNICODE translation handled automatically
  - Much easier to read and Search data
- For the Binary data using BLOBs:
  - Allows much more data to be stored

Inline LOBs in SPT01

- At GA BLOB/CLOB used in SPT01 with no compression
- This caused 2 problems
  - Space grew because Auxiliary TS could not be compressed
  - Performance suffered due to need to always read Auxiliary Space
- APARS PM27811 & PM27073 changed this
- COMPRESS_SPT01 becomes Opaque
- SPT01_INLINE_LENGTH sets max length
  - Defaults to 0
  - Setting large improves compression and fetch performance
  - Setting small allows more rows & reduces space problems
  - Online Alterable but care required if reducing (REORP)

Why do you care about DB2 using LOBS?

- Key implication is to your Backup & Recovery strategy
  - LOBs are created during Migration to NFM
  - But there are also implications due to Referential Integrity
- Recovery to PIT works differently on Catalog & Directory
  - CHKP and ACHKP are not set on these objects
  - If you don't recover the Base and Auxiliary Tablespaces together
  - Or if you don't Recover the complete RI set together
- Order of object Backup and especially Recovery critical
  - If in CM DSNTIJIC skips new or obsolete objects
    - Still get RC=0 with message DSNU1530I
  - See Utility Guide and Reference for new order once in NFM
  - If you stop halfway through ENFM it could get interesting!

New Catalog Tables (Slide 1 of 2)

- SYSPACKCOPY , SYSQUERY, SYSQUERYOPTS, SYSQUERYPLAN
  - Used to support Plan Stability feature
- SYSPENDINGDDL , SYSPENDINGOBJECTS
  - Support the new Deferred ALTER feature
- SYSDUMMYA, SYSDUMMYE, SYSDUMMYU
  - Equivalents of SYSDUMMY using ASCII, EBCDIC and Unicode
  - Used to be outside the Catalog in DB2 9
  - Note SYSDUMMY1 still exists for compatibility purposes
    - Stored in the SYSEBCDC Tablespace along with SYSDUMMYE

New Catalog Tables (Slide 2 of 2)

- SYSAUDITPOLICIES
  - Supports the new Audit Policies feature
- SYSAUTOALERTS, SYSAUTORUNS_HIST, SYSAUTOTIMEWINDOWS, SYSTABLES_PROFILES
  - Used by the Autonomic RUNSTATS feature
- SYSCONTROLS
  - Supports Row Permissions and Column Masks

These are some of the new Catalog Tables introduced in DB2 10 along with the features they support. I don't plan to go through each of these in any more detail. Many of them have been covered in much more depth during the Conference and the material is available on the IDUG website.

A few interesting New Columns

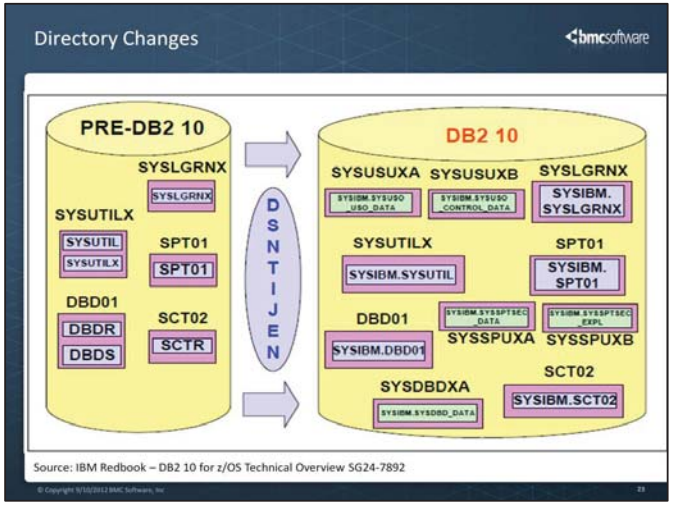
- PERIOD has been added to several tables
  - Used to indicate Start and End of Business or System Time
- LASTUSED in SYSPACKAGE and SYSPLAN
  - Does what it says on the tin!
- PLANMGMT in SYSPACKAGE
  - Tells you what Plan Stability option is being used for the package
- HASHSPACE in SYSTABLESPACE & SYSTABLEPART
  - Provides information about Hash Space used
  - Other new columns also used in various Tables
- MEMBER CLUSTER in SYSTABLESPACE
  - This used to stored in TYPE column as 'I' or 'K'

Interesting New Columns

- DRIVETYPE on SYSTABLESPACESTATS
  - Indicates whether the Table is stored on Solid State device
  - Also included in SYSINDEXSPACESTATS
- REORGSCAN_ACCESS in SYSTABLESPACESTATS
  - Number of times Table has been access since last REORG or LOAD REPLACE operation (or since creation)
  - Many more columns of this type are being added in every release
- HASHLASTUSED in SYSTABLESPACESTATS
  - Useful to determine whether Hash is used properly or not
  - Look at this in combination with the last column

Some Changes to Existing Columns

- SYSCOLUMNS
  - COLTYPE uses TIMESTZ for Timestamp with Timezone
  - SCALE used to record number of fractional digits for Timestamp
- SYSCOPY
  - STYPE has many new settings for OLS changes
- SYSTABLES
  - TYPE has an 'H' option for History Tables (Temporal data)
- SYSVIEWS
  - STATEMENT and PARSETREE contain the data that used to be stored in SYSIBM.SYSVTREE and SYSIBM.VLTREE
    - More for Internal purposes than for customer use



I took the liberty of copying this diagram from the IBM Technical Overview Redbook as it provides the clearest description of the Directory changes I've seen.

Directory Changes

bmcsoftware

- Implemented by DSNTIJEN
- Some Tables merged and are all more regular tables
  - For example they even have a Creator Name now!
  - But still not registered in the Catalog
- All 5 new objects are Auxiliary Objects for LOB data
  - 2 LOB columns on SYSUTIL
  - 2 LOB columns on SPT01
  - 1 LOB column on the DBD
- DBD01, SPT01 and SYSUTIL are now PBG objects
  - SPT01 now uses a 32Kb Pagesize
  - Links within DBD01 have been removed

Directory Changes

- Removal of the UTSERIAL lock
  - This was a special lock type used by Utilities
    - Used when updating SYSUTILX Directory Tablespace
  - Sometimes caused significant contention with multiple utilities
  - DB2 now takes a regular Page level lock
  - Should provide good benefits

SELECT from Directory

- Added well after GA date (November 2011)
  - See APARs PM35190 and PM42331
- Supported for SYSLGRNX and SYSUTIL
  - APAR quotes SYSUTILX but that has been removed?
  - *"Intended primarily for use by IBM Support"*
- No L-locks acquired regardless of Isolation Level
  - Can get -607 if SQL cannot be converted to WITH UR
- Enabled by a new execution of CATMAINT
  - Even for new installations
  - CATMAINT UPDATE UNLDDN PM35190
- Could be useful to identify such things as executing utilities, points in the log where datasets were open etc.


bmcsoftware
BUSINESS RUNS ON I.T.
I.T. RUNS ON BMC™

SQLADRIA SEMINAR
Zagreb, 28 September 2012

MDM, DG, RDM, DM, IG …

Goran Bavčar
NLB d.d.

NLB

www.nlb.si

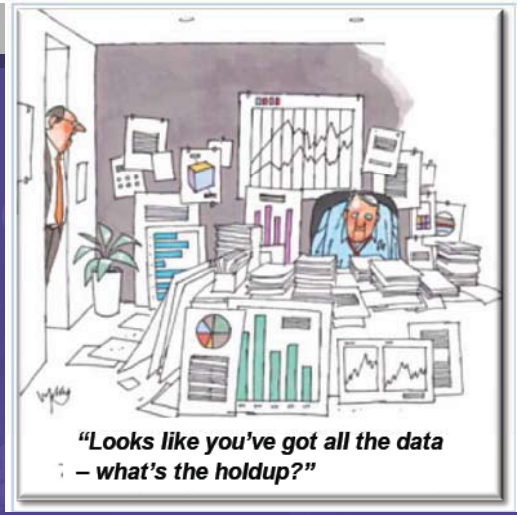
sqldria

MDM, DG, RDM, DM, IG …

za DBA, razvijalce in ostale tehnične ljudi

NLB

www.nlb.si



NLB

www.nlb.si

Reduce-Reuse-Recycle … Data?

- Reduce the amount of organizational data ROT
  - Redundant, obsolete, trivial
- Reuse the remainder
  - Fewer vocabulary items to resolve
  - Greater quality engineering leverage
- Integration is impossible without information architecture components (for mapping)
  - Maintenance of these components promotes greater reuse
  - Shared data is typified by organizational ability to use information as a strategic asset
- However, assets are useless without knowledge of the asset characteristics

NLB

www.nlb.si

MDM & its variants

- Customer Data Integration (CDI)
- Product Information Management (PIM)
- Reference Data Management (RDM)
- Geographic/organizational Data Management
- Master Data Governance (MDG)

MDM Working Definitions

Master Data Management (MDM)

Authoritative, reliable foundation for data used across many applications & constituencies with goal to provide single view of truth no matter where it lies.

Customer Data Integration

Processes & technologies for recognizing a customer & its relationships at any touch-point while aggregating, managing & harmonizing accurate, up-to-date knowledge about that customer to deliver it 'just in time' in an actionable form to touch-points.

Product Information Mgmt (PIM)

Processes & technologies for recognizing PRODUCT, SUPPLIER, & PRICING master data

CDI is mandatory first step for most organizations on journey to MDM

MDM Working Definitions - continued

Multi-Entity Master Data

An MDM solution to concurrently manage multiple, diverse master data domains (customers, accounts, products) across intra- and extra-enterprise business processes

Master Reference Data

Non-volatile reference data shared across the enterprise in context within applications (& possibly within the industry via "exchanges"); examples: units of measure; currency rates; calendars; other look-up tables

Large enterprises assume MDM = multi-entity MDM when evaluating software; RDM remains a key variable among the vendors as few vendors are enabled (yet)

MDM Working Sub-Definitions ("Use" Cases)

Operational MDM

Definition, creation, & synchronization of master data required for transactional systems & delivered via SOA to run the business; examples: near R/T customer hubs & securities masters

Analytical MDM

Definition, creation, & analysis of master data to measure the business; examples: counterparty risk mgmt apps & financial reporting consolidation

Collaborative MDM

Definition, creation, & synchronization of master reference data via workflow & check-in / check-out services;

For most G5000 enterprises, multiple (often all) variants will be needed to make MDM initiatives successful

Most Common MDM Topologies

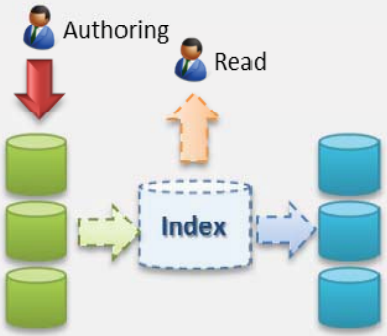
IMPLEMENTATION STYLE	DESCRIPTION
External (Service Provider)	• Database marketing providers • Data service providers • Service bureaus
Persistent (Database)	• Master customer information file/database • Operational data store/active data warehouse • Relational DBMS + Extract-Transform-Load (ETL) + Data Quality (DQ)
Registry (Virtual)	• Metadata layer + distributed query (enterprise information integration or EII) • Enterprise application integration (EAI) • Portal
Composite (Hybrid)	• Ability to fine-tune performance & availability by altering amount of master data persisted • XML, web services, service-oriented architecture (SOA)
"Chernobyl"	• Encapsulate legacy applications

Composite/Hybrid is majority architectural preference; Registry/Virtual 2nd choice

NLB

REGISTRY HUB

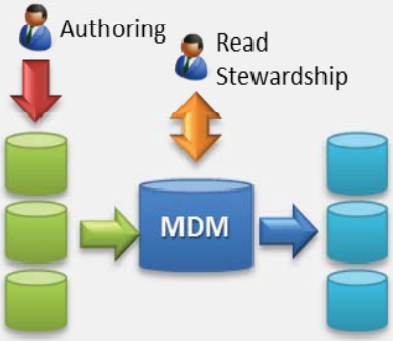
- Cross Ref Index
- No data consolidation
- Read access only
- Use of federated queries
- Non intrusive
- May not scale



NLB

CONSOLIDATION HUB

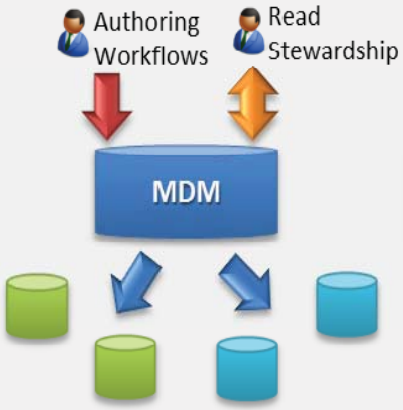
- Consume source data
- De-duplicate, enrich and consolidate
- Stewardship workflows
- Publish data to downstream apps
- Non intrusive



NLB

CENTRALIZED/TRANSACTIONAL HUB

- Direct data authoring in MDM
- Validation and Stewardship Workflows
- Source of truth for all applications
- Most intrusive - replaces existing processes



NLB

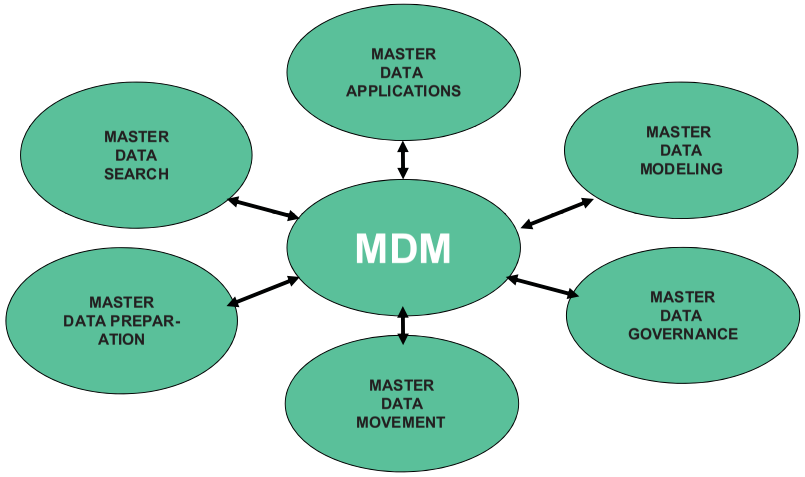
'Top 5' Business Drivers for MDM Initiatives

1. Synergies for cross-sell & up-sell
2. Compliance & regulatory reporting
3. "Once & done" economies & customer satisfaction
4. Legacy system integration & augmentation
5. Economies of scale for M&A

NLB

www.nlb.si

- • The business needs to take the lead and recognize the value of information.
  - Doing so can be a key differentiator for your company
  - Not engaging the business will mean you are more than likely to fail
- • Starting initiatives such as SOA can never succeed without Information being managed correctly.
- • You need to have an information strategy to set priorities
  - Master Data Management strategy
  - Information Services catalog
- • The information management people should all have a common goal:
  - Deliver accurate, reliable and fast information through the preferred interface of the consumer.



MDM - DG

- Data governance (DG) is vital to success of MDM projects – both initially & ongoing
- During 2011-12, Global 5000 enterprises will increasingly mandate that "*no MDM program be funded without the pre-requisite DG framework*"
- "Proactive DG" that includes entire master data lifecycle will increasingly be mandated as a core phase zero or phase one deliverable of most large-scale MDM projects
- Given substantial investment required for MDM programs, co-dependence/synergy of DG & MDM must be given close scrutiny – not only to contain costs, but also to insure success

Data Governance Working Definitions

Data Governance (DG)

Formal orchestration of people, process, & technology to enable an organization to leverage data as an enterprise asset across different lines-of-business and IT systems

Passive Data

Data steward consoles & other reactive data management capabilities focused on after-the-fact data compliance; often batch-like & not integrated with MDM

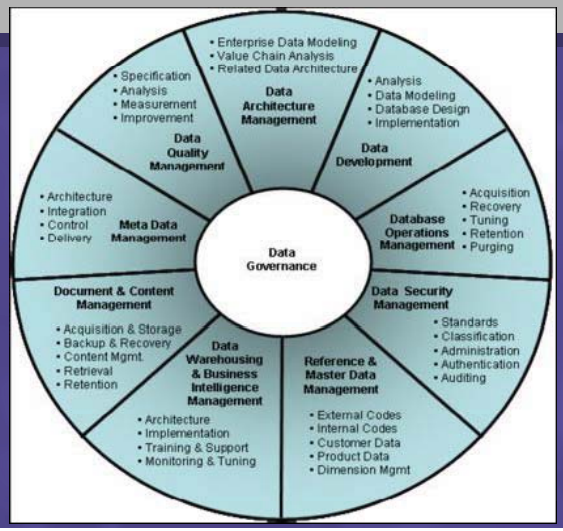
Active Data

Metrics-enabled, upstream data policy enablement; replaces manual data admin processes with role-based, real-time SME empowerment

Proactive Data

Metrics-driven, crowd-sourced capability for business users & IT to actively control their shared data across different lines of business & IT systems

Master Data Governance aligns IT-centric viewpoints & business-centric viewpoints regarding data quality; MDM & MDG are becoming natural way of aligning data with business processes



Master Data Governance Objectives

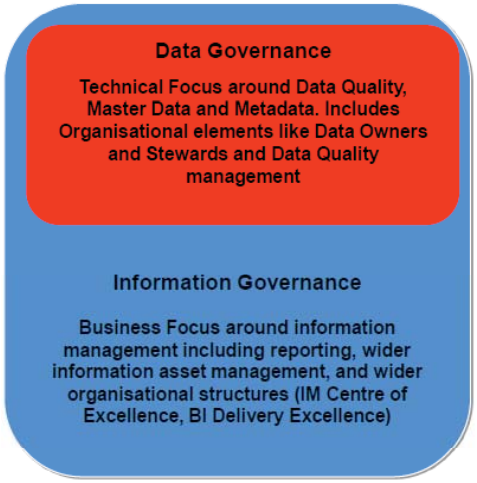
- Understand & manage strategic & tactical data, project ownership from data perspective, & priority setting for data projects
- Define day-to-day activities of creating, using & retiring data
- Describe how, when & by whom data was received, created, accessed, modified &/or formatted
- Determine whether data is fit for its intended use, including completeness & business-rule compliance
- Implement processes to cleanse, transform, integrate & enrich fresh data across subject areas
- Address security & privacy compliance across integrated subjects
- Manage master data by examining data assets & relationships that define enterprise operations

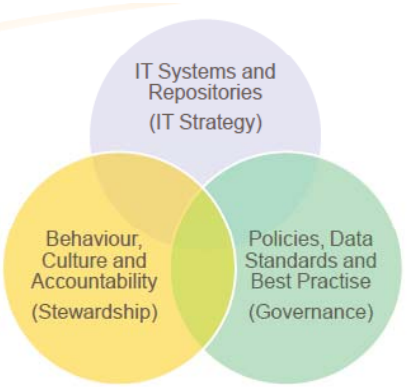
Source: Lance Miller/ Teradata
Teradata Magazine-March 2008

Data Governance is the collective effort to ensure we can trust data

Some definitions.....

- *Data governance is about how an organisation uses data to benefit and protect itself.* Source: http://www-01.ibm.com/software/
- *Data Governance (DG) refers to the overall management of the availability, usability, integrity and security of the data employed in an organisation:.*
  Source:http://searchdatamanagement.techtarget.com/definition/data-governance
- *Data governance is the practice of organizing and implementing policies, procedures and standards for the effective use of an organization's ..information assets.* *Source: www.sdn.sap.com*
- *Data Governance is the formal orchestration of people, process and technology to enable an organisation to leverage data as an enterprise asset.*
  Source:http://0036f23.netsolhost.com/data_governance.htm
- *Data Governance: The execution and enforcement of authority over the management of data assets and the performance of data functions.*
  Source:www.tdan.com/view-articles/5037



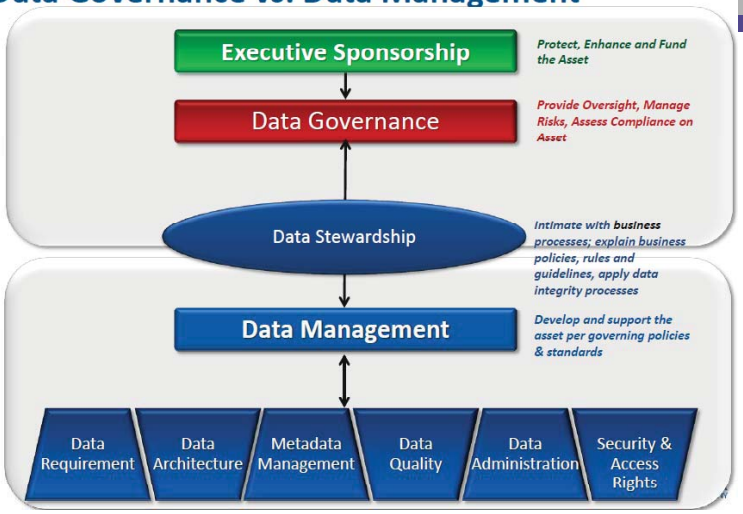


Partner with major IT initiatives to connect up our data; it's not just about technology, it has to be about consistent standards, process and behaviours too.

Embedding stewardship frameworks and culture into all departments.

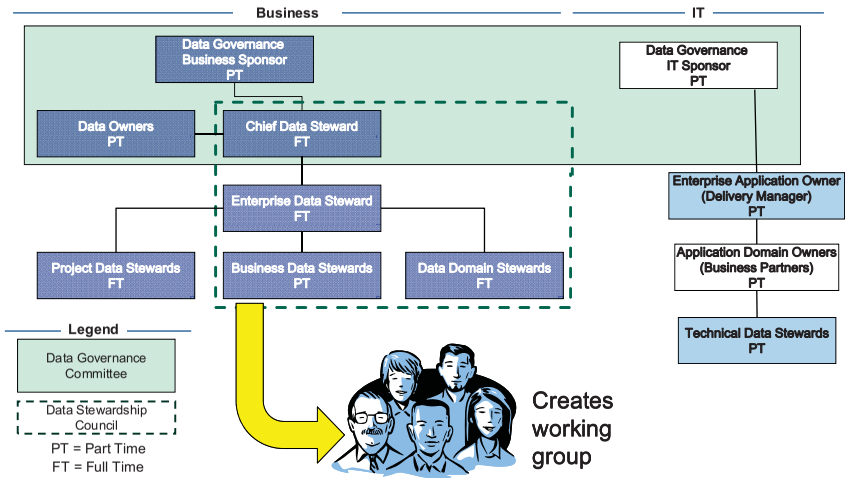
Working on key focus areas where improved stewardship can add value to the business.

Data Governance vs. Data Management



Data Governance Organization

Business and IT view of the Data Governance Organization:



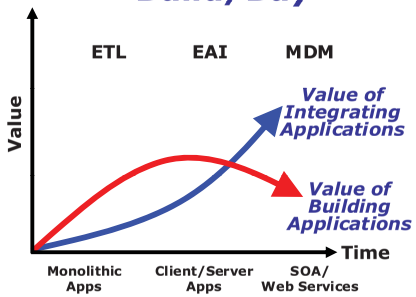
MDM Institute's Data Governance Maturity Model

- **"Anarchy" (basic)** – Application-centric approach; meets business needs only on project-specific basis
- **"Feudalism" (foundational)** – IT policy-driven standardization on technology & methods; common usage of tools & procedures across projects
- **"Monarchy" (advanced)** – business-driven, rationalized data with data & metadata actively shared in production across sources
- **"Federalism" (distinctive)** – SOA (modular components), integrated view of compliance requirements, formalized organization with defined roles & responsibilities, clearly defined metrics, iterative learning cycle

"The World Is Flat"

- Data structures & business processes must be supremely flexible
- Underlying IT infrastructure must enable new business models
- Policies/process flows must integrate in ways previously problematic

Value of Integration Exceeds Value of Build/Buy



EAI = enterprise application integration
ETL = extract-transform-load
MDM = master data management
SOA = service-oriented architecture

Global competition mandates a wide variety of new business styles

How Many Analysts Does It Take To Change a Light Bulb?

- **Gartner analyst**
  - "We feel that a new bulb is necessary & that the bulb will be replaced (0.99 probability) — we have a new service that addresses that issue"
- **Forrester/Giga analyst**
  - "In 5 years, the new illumination technologies will replace what you currently have ... Wait"
- **Ovum/Aberdeen analyst**
  - "We'll write about the old bulb for $25,000"
- **IDC analyst**
  - "There are 1,230,245 burnt-out bulbs in the world — for $2,500, we will tell you where they are ..."
- **Big Three consultant**
  - "It's time to re-engineer the sun ..."

NLB

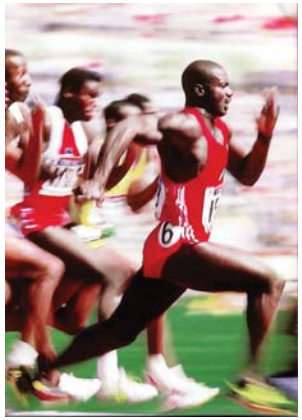
www.nlb.si

MDM myths

- • MDM is a product that I can buy
  - Can't just buy your way out. This requires a global project and conceptual approach.
- • MDM requires a big bang approach
  - Not quite. Perfectly possible to implement this is small increments.
- • This is an IT project
  - Not possible to do this without involvement from the business side.
- • This is a business project
  - It will require significant IT investment to put this on rails.

Remember: DG (&MDM) is a Program

- DG is a Program, Not a Project
- DG is a marathon, Not a sprint



NLB

Data Governance Lessons Learned

- It's hard to explain to our management
- ...so it's hard to enlist people to participate
- It can turn into an academic exercise
- Many companies are now on Round 2!
- Executives get nervous about the "G" word
- If you fail once, you may not get a second shot

NLB

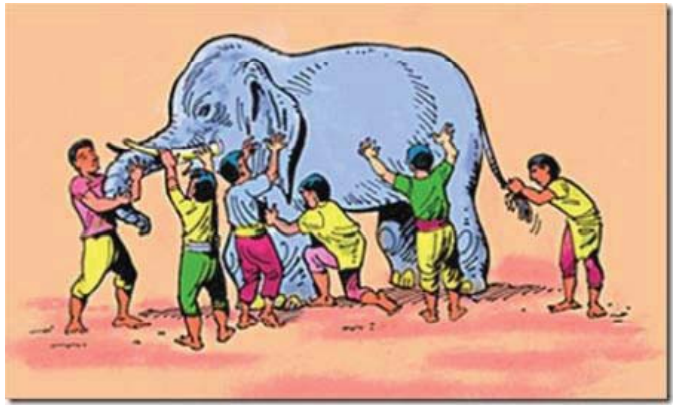
4 Things To Consider

- Understand how the data is used
  - To identify consumers who will care about the data
- Establish the impact of bad data
  - To establish business case
- Identify champions
  - To embed adoption
- Pick your battles
  - Understand that not all data is "created equal"

The art of convincing and keeping them convinced

1. Convincing :
   - Top-down : business case – here you get the budget
   - Bottom-up : "We want to work better, we want better quality" - here you get collaboration
2. Keeping them convinced :
   - Come back every 3-6 months with an added value
   - Define your approach so, that the project could be stopped every moment in time.

NLB



NLB

www.nlb.si

Thank you!