John Deere participated in the DB2 9 for z/OS Early Support Program back in 2007. Since that time we have been migrating DB2 9 to our Classic/Legacy and SAP environments. This updated presentation will include the items we tested as part of the ESP and include new material from what we have learned during the migrations and while running DB2 9. This presentation will provide an introduction to some of the new DB2 9 features including universal tablespaces, clone tables, partition by growth, index compression, not logged tables, roles and trusted context, several utility enhancements, reordered row format and DB2 9 spatial object exploitation. Just for fun, I’ll compare some of the DB2 9 features with the equipment we make at John Deere.
Agenda

- Migration
- Universal Tablespace
- Clone Tables
- Partition by Growth
- Index Compression
- Not Logged Tables
- Roles and Trusted Context
- Utilities
  - COPY
  - RECOVER
  - REBUILD INDEX
  - CATMAINT
Agenda

- Reordered Row Format (RRF)
- Think REBIND
- Performance apars of interest
- Spatial data
Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

The material in this presentation is based on our experience at John Deere. This may or may not be indicative of what you will experience in your shop.

Any apars mentioned in this presentation were current at the time this presentation was written. You should check the current maintenance recommendations for new apars, pe’s and pre-req’s.
We'll use equipment from my company to summarize the key features that we have learned about DB2 9.

See why I compare DB2 9 to an end loader, a motor grader, a combine, an excavator, a tractor and the Gator.
z/OS 1.8 is required for the volume based utility enhancements and for database roles and trusted contexts.

The temporary database used by declared global temporary tables and static scrollable cursors goes away and these functions use the workfile database, DSNDB07. You may want to increase the size of DSNDB07. Do not drop the temporary database until you are sure that you won’t fallback to DB2 Version 8.

Migration

- Info Apar II14401 and II14464.
- DB2 V8 Tolerance Apar PK11129 (UK90008).
  - Must be on all DB2 Data Sharing members.
- Migrate only from DB2 V8 NFM.
- Expanded BSDS (DSNJCNVB).
- PDSE requirement for SDSNLOAD.
- DB2 Stored Procedures WLM Managed.
- z/OS 1.7 (some new functions require z/OS 1.8).
- Converged Temporary Space.
Compatibility Mode (CM) is the mode that DB2 is in when DB2 9 is started for the first time from DB2 V8 New Function Mode (NFM). Limited new function of DB2 9 is available in CM. This is the only mode that permits fallback to DB2 Version 8.

Enabling New Function Mode (ENFM) is the mode that is entered when CATENFM START is executed which is the first step of DSNTIJEN. ENFM can fall back to Compatibility Mode which will be identified as CM*.

New Function Mode (NFM) is the mode that DB2 enters once CATENFM COMPLETE is run. This is the only step of job DSNTIJNF. New functions can be used at this time. Fallback can occur to Enabling New Function Mode which will be identified as ENFM* or to Compatibility Mode which will be identified as CM*.

Enabling New Function Mode Star (ENFM*) is the same as ENFM but indicates that at one time DB2 was in NFM.

Compatibility Mode Star (CM*) is the same as CM but indicates that at one time DB2 was in either ENFM or NFM. You can not fallback to DB2 Version 8 from CM*.

In message DSN7100I the actual DB2 9 migration modes will be identified as C, C*, E, E*, or N.
### Migration/Fallback paths tested at John Deere

- Apply fallback SPE to DB2 V8.
- Migrate to CM. (CATMAINT, DSNTIJTC)
- Fallback to V8.
- Remigrate to CM.
- Migrate to ENFM. (DSNTIJEN)
- Fallback to CM*. (DSNTIJCS)
- Remigrate to ENFM. (DSNTIJEN)
- Migrate to NFM. (DSNTIJNF)
- Fallback to ENFM*. (DSNTIJES)
- Remigrate to NFM. (DSNTIJNF)

---

DSNTIJTC is used to migrate to DB2 9.

DSNTIJEN is used to move to ENFM.

If the real time statistics database, DSNRTSDB, does not exist then steps ENFM1103 and ENFM1103 of job DSNTIJEN will fail. These steps load data from the database to the catalog.

DSNTIJCS is used to fallback to CM* from ENFM, ENFM* or NFM.

DSNTJINF is used to move to NFM.
This message will be issued when starting DB2 if the DB2 Fallback SPE has not been applied.
The current DB2 9 mode is identified by message DSN7100I.
C – Compatibility Mode (CM)
C* - Compatibility Mode Star (CM*)
E - Enabling New Function Mode
E* - Enabling New Function Mode Star (ENFM*)
N – New Function Mode
Universal Tablespace

- A new DB2 9 tablespace.
- A partitioning tablespace with the features of a segmented tablespace.
- One table per tablespace.
- DB2 9 features that use the Universal tablespace.
  - Clone Tables
  - Partition by Range
  - Partition by Growth
- To convert existing tablespace to UTS in DB2 9 have to do a drop, create and reload data.

Simple tablespaces – multiple tables per tablespace, different tables share pages. Simple tablespaces are deprecated in DB2 9. You can use existing ones but can not create new ones.

Segmented – multiple tables per tablespace, different tables don’t share pages. Includes a space map page.

Partitioned – one table per tablespace. Needs a partitioning column.

Universal Tablespace – A partitioned segmented tablespace. Contains a space map.

Benefits of Universal Tablespace (UTS) include improved table scan performance, improved mass delete performance and improved ALTER TABLE ROTATE performance.
Clone Tables

- Fast replacement of one table with another.
- IBM’s solution to the request for online load replace.
- Application is unaware of the clone table, no application changes are necessary.
- Can not clone a table that has RI.
- Can not make schema changes to a table that has a clone.
- No clone tables on MQTs.
- Can not clone the DB2 Catalog tables.
- Must be a Universal Tablespace, DB2 managed and the only table in the tablespace.

If the tablespace is not a Universal Tablespace receive SQL error:
DSNT408I SQLCODE = -148, ERROR: THE SOURCE TABLE DVDB2SP.USER_GROUP_CONNECT CANNOT BE ALTERED. REASON 05
DSNT418I SQLSTATE = 42809 SQLSTATE RETURN CODE
DSNT415I SQLERRP = DSNXICLN SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD = 17 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION

You can not make schema changes:
ALTER TABLE DVDB2SP.USER_GROUP_CONNECT2
ADD TEST_COL CHAR(1);
DSNT408I SQLCODE = -148, ERROR: THE SOURCE TABLE DVDB2SP.USER_GROUP_CONNECT2 CANNOT BE ALTERED. REASON 11
DSNT418I SQLSTATE = 42809 SQLSTATE RETURN CODE

Additional columns added to DB2 Catalog tables:
SYSTABLESPACE
  Instance
  Clone
SYSCOPY
  Instance
Clone Tables

- Clone table is identical to the base table.
  - Created in the same tablespace.
  - Created with the same indexes.
- Clone table created as empty.
  - Table must be populated.
- Grant access on the clone table.
  - Security separate from base table.
- Utilities process against the clone table.
  - Can not run RUNSTATS on a clone table.
  - Consider running RUNSTATS after EXCHANGE DATA if there is significant differences in the data between the two tables.

The clone table is identical to the base table in every way including columns, data types and check constraints. It is even created with the same indexes and before triggers. The clone table can be created with a different schema (owner) than the base table.
Clone Tables

- `ALTER TABLE DVDB2SP.USER_GROUP_CONNEC2 ADD CLONE USER_GROUP_CONNECT_CLN;`

- Table is created in the same table space as the base table.

- Base table:
  - DA296.DSNDBC.VRADB002.VRATS901.I0001.A001
  - DA296.DSNDBD.VRADB002.VRATS901.I0001.A001

- Clone table:
  - DA296.DSNDBC.VRADB002.VRATS901.I0002.A001
  - DA296.DSNDBD.VRADB002.VRATS901.I0002.A001

Same database, same table space, difference only in the instance number.

'`ALTER TABLE base-table-name DROP CLONE`’ will drop the clone table.
`'DROP TABLE’` will drop both the base table and the clone.
Clone Tables

```sql
-DIS DB(VRADB002) SPACE(*)
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>PART</th>
<th>STATUS</th>
<th>(Details)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRATS901 TSB1</td>
<td>0001</td>
<td>RW</td>
<td></td>
<td>(Base Table - I0001.A001 dataset)</td>
</tr>
<tr>
<td>VRATS901 TSC2</td>
<td>0001</td>
<td>RW</td>
<td></td>
<td>(Clone Table - I0002.A001 dataset)</td>
</tr>
<tr>
<td>VRAI901 IXB1</td>
<td>0001</td>
<td>RW</td>
<td></td>
<td>(Base index - I0001.A001 dataset)</td>
</tr>
<tr>
<td>VRAI901 IXC2</td>
<td>0001</td>
<td>RW</td>
<td></td>
<td>(Clone index - I0002.A001 dataset)</td>
</tr>
</tbody>
</table>

In this example, the base table and it’s associated base index have a dataset suffix of I0001.A001. The clone table and it’s associated index have a dataset suffix of I0002.A001.
**Clone Tables**

**EXCHANGE DATA BETWEEN TABLE**
DVDB2SP.USER_GROUP_CONNEC2 AND
DVDB2SP.USER_GROUP_CONNECT_CLN;

--DIS DB(VRADB002) SPACE(*)

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>PART</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRATS901</td>
<td>TSB2</td>
<td>0001</td>
<td>RW (Base table, i0002.A001 dataset)</td>
</tr>
<tr>
<td>VRATS901</td>
<td>TSC1</td>
<td>0001</td>
<td>RW (Clone table, i0001.A001 dataset)</td>
</tr>
<tr>
<td>VRAI901</td>
<td>IXB2</td>
<td>0001</td>
<td>RW (Base index, i0002.A001 dataset)</td>
</tr>
<tr>
<td>VRAI901</td>
<td>IXC1</td>
<td>0001</td>
<td>RW (Clone index, i0001.A001 dataset)</td>
</tr>
</tbody>
</table>

*No data is moved.*

The **EXCHANGE DATA** command will switch the underlying datasets associated with the base table and the clone table. Only the data instances will change, no data will be copied. This is transparent to the application and does not require any application changes.

On **EXCHANGE DATA**, the clone assumes the base objects’ statistics. This allows bound static SQL to function without a rebind.

**EXCHANGE DATA** will put an entry in **SYSCOPY** with ICTYPE=’A’ and STYPE=’E’.

**EXCHANGE DATA** requires one of the following:

1) Ownership of both tables.
2) Insert and delete privileges for both tables.
3) DBADM authority on the database.
4) SYSADM authority.
We have a security application table that is refreshed every evening. Originally this was done using the DB2 LOAD utility with the REPLACE option. This resulted in the table being unavailable for 15-30 minutes. We switched to emulating an online load using a copy of the table using REPAIR utilities and IDCAMS renames of underlying datasets. This cut the outage time down to one minute. (We learned about this process at an IDUG conference.) Even with this methodology we still had some issues. The tablespace still had to be taken offline. The tablespace may not be stopped immediately (STOPP) which impacts the application. By implementing Clone Tables for this application we will be able to refresh the data nightly without impacting the application.
Clone Tables are like a Loader

- You can load new data into them without affecting your production table.
Partitioning Features in DB2 9 NFM

- **Partition by Range.**
  - UTS: one table per tablespace.
  - Partitioning column required.
  - CREATE TABLE...SEGSIZE...NUMPARTS.
  - SYSTABLESPACE has TYPE column of ‘R’.

- **Partition by Growth.**
  - UTS: one table per tablespace.
  - No partitioning column required.
  - Non-partitioning indexes only.
  - CREATE TABLE...SEGSIZE...MAXPARTITIONS.
  - SYSTABLESPACE has TYPE column of ‘G’.

Partition by Range still requires a partitioning column but the Universal Tablespace provides advantages of the Partitioned tablespace.

Partition by Range advantages:

- Better space management, fewer reorgs.
- Improved mass delete.
- Improved ALTER TABLE ROTATE performance.
Specifying keywords SEG SIZE and MAX PARTITIONS will create a Universal Tablespace as Partition by Growth. Specifying MAX PARTITIONS without SEG SIZE will create a UTS as Partition by Growth with a default segsize of 4K. No partitioning index is defined.

Specifying keywords SEG SIZE and NUM PARTS will create a Universal Tablespace as Partition by Range. A partitioning index is required.

Specifying only the SEG SIZE keyword creates a segmented tablespace.

Specifying only the NUM PARTS keyword creates a partitioned tablespace.

If none of the keywords are specified the default table space is a Partition by Growth Universal Tablespace.

<table>
<thead>
<tr>
<th>SEG SIZE</th>
<th>MAX PARTITIONS</th>
<th>NUM PARTS</th>
<th>TABLESPACE TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEG SIZE</td>
<td>MAX PARTITIONS</td>
<td></td>
<td>PARTITION BY GROWTH (UTS)</td>
</tr>
<tr>
<td></td>
<td>MAX PARTITIONS</td>
<td></td>
<td>PARTITION BY GROWTH (UTS) Segsize 4K</td>
</tr>
<tr>
<td>SEG SIZE</td>
<td>NUM PARTS</td>
<td></td>
<td>PARTITION BY RANGE (UTS)</td>
</tr>
<tr>
<td>SEG SIZE</td>
<td>NUM PARTS</td>
<td></td>
<td>SEGMENTED</td>
</tr>
<tr>
<td></td>
<td>(blank)</td>
<td>(blank)</td>
<td>PARTITION BY GROWTH (UTS) – previously Simple TS</td>
</tr>
</tbody>
</table>
Partition by Growth

- Tablespace divided into separate partitions and managed by DB2.
- Allocate partition when needed.
- Compression dictionary and other attributes copied from previous partition to new partition.
- Beneficial -
  - When there is no obvious partitioning column.
  - For segmented tablespaces exceeding 64GB.
  - For ERP applications such as SAP.

Partition by Growth partitions according to space needs and not column values.

A Partition by Growth tablespace can be explicitly created with
CREATE TABLESPACE … MAXPARTITIONS n

Or implicitly created with
CREATE TABLE PARTITION BY SIZE EVERY nG
Partition by Growth

- A partition by growth tablespace can grow up to 128 TBs. Maximum size is determined by MAXPARTITIONS, DSIZE and PAGE SIZE.
- MAXPARTITIONS starts with one partition and then adds other partitions as needed up to the value specified for MAXPARTITIONS.
- REORG does not delete existing partitions even if they are no longer needed.
- Incompatible with ADD PARTITION and ROTATE PARTITION.

You can not specify NUMPARTS with MAXPARTITIONS.
Changes to DSIZE and SEGSIZE require a DROP to change, there is no ALTER option.

The number of partitions will change dynamically.
This is an example of results we saw at John Deere during the DB2 9 ESP for a tablespace defined as Partition by Growth. These results may or may not be indicative of what you will see at your shop based on your data.
One of the major benefits of a partition by growth tablespace is that you do not define a partitioning key.

Running reorg after loading the partitions may result in additional compression.
Partition by Growth is like a Motor Grader

- DB2 will spread your data across the partitions.
Index Compression

- Index must be in 8K, 16K, or 32K buffer pool.
  - Indexes in DB2 V8 are 4K BP.
  - SQL code -676 if incorrect page size.
- ALTER INDEX COMPRESS YES.
  - Index placed in RBDP (rebuild pending).
- New column in SYSIBM.SYSINDEXES.
  - COMPRESS ... Y/N.
- DSN1COMP utility can be run against the index .
- No compression dictionary for index compression.

‘ALTER INDEX SAPR3.CKIT10 COMPRESS YES’ returns SQLcode -676 if index page size is not 8K, 16K or 32K.
SQLCODE: -676, ERROR: THE PAGE SIZE OF THE BUFFERPOOL FOR THE INDEX
IS NOT VALID FOR THE SPECIFIED INDEX
Index Compression

- Convert 4K index to 8K, 16K, or 32K.
- Turn compression on.
- Process:
  - STOP INDEX
  - ALTER INDEX ...BP8K
  - ALTER INDEX...COMPRESS YES
  - REBUILD INDEX

(Sample jcl is in the notes section of the presentation.)

Altering the Index to a different buffer pool puts the index in Rebuild Pending (RBDP) status. Altering the index to turn on compression also puts the Index in RBDP. However both alter statements can be executed in the same job as shown below:

```
//DB2CMD   EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSTSIN  DD *  

DSN SYSTEM(X96A)
-STOP DB(A000XAAL) SP(TECSH0)  
  BP change requires the IX Space to be stopped
RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2) LIB('DB2X96.X96A.RUNLIB.LOAD')
-STA DB(A000XAAL) SP(TECSH0)
  Restart IX Space
END

//SYSIN   DD *
ALTER INDEX SAPR3."TECS~0" BUFFERPOOL BP16K1;  
  Move to 16K BP
ALTER INDEX SAPR3."TECS~0" COMPRESS YES;  
  Turn compression on
```
Index Compression - DSN1COMP

8 K Page Buffer Size yields a

51 % Reduction in Index Leaf Page Space
The Resulting Index would have approximately
49 % of the original index's Leaf Page Space
No Bufferpool Space would be unused

16 K Page Buffer Size yields a

71 % Reduction in Index Leaf Page Space
The Resulting Index would have approximately
29 % of the original index's Leaf Page Space
13 % of Bufferpool Space would be unused to

ensure keys fit into compressed buffers

Use DSN1COMP before you decide on an index page size. There may be a trade off between disk space savings and wasted buffer pool buffers.
This is an example of results we saw at John Deere during the DB2 9 ESP. These results may or may not be indicative of what you will see at your shop based on your data.

Before compression four instances of the tablespace dataset were needed (A001 through A004).

With 8K buffer pool and index compression only two instances were needed and with 16K buffer pool and index compression all the data fit into a single dataset.
This is an example from our testing of index compression. In this example there was minimal impact to CPU time with index compression.

This is an example of results we saw at John Deere during the DB2 9 ESP. These results may or may not be indicative of what you will see at your shop based on your data. Index page size used 8K. The CPU time is Class 2.

<table>
<thead>
<tr>
<th>Access</th>
<th>Number of Rows</th>
<th>CPU Before Compression</th>
<th>CPU After Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Count, Index Only (Unique index)</td>
<td>88 million</td>
<td>1min 19.52sec</td>
<td>1min 16.09sec</td>
</tr>
<tr>
<td>Select Count, Index Only (Non-unique IX)</td>
<td>88 million</td>
<td>45.01 sec</td>
<td>45.11 sec</td>
</tr>
<tr>
<td>Random access, Index only (1 index)</td>
<td>76 million</td>
<td>79min 57.38 sec</td>
<td>81min 15.61 sec</td>
</tr>
<tr>
<td>Random Inserts (5 indexes)</td>
<td>3011</td>
<td>21.06 sec</td>
<td>20.08 sec</td>
</tr>
<tr>
<td>Sequential Inserts (5 indexes)</td>
<td>3011</td>
<td>16.40 sec</td>
<td>17.35 sec</td>
</tr>
</tbody>
</table>

Index page size used 8K. CPU time is Class 2.
Index Compression is like a Combine

• You will be able to harvest dast savings.
Not Logged Tablespace

- DB2 9 New Function Mode only.
- Default is logged.
- DBADM authority required.
- Undo/Redo log records are not maintained.
- Indexes inherit the logging attribute from the base tablespace.
- You need to think this one through. Don’t use it just because you can.

Not logged has a very small gain in cpu time. Elapsed time improvements are small other than in cases where the log has become a bottleneck, such as in cases where 20 concurrent batch insert jobs are inserting millions of rows each. With other options, such as fast disks, log striping and DB2 9 log improvements, you can log very quickly. The small gain may cost more in recoverability than it gains.
For a single task inserting it is very difficult to see significant differences in performance. If you are running 10 or 20 concurrent tasks then the log could be a bottleneck. Scalability for logging is improved in DB2 9.

Not Logged Tablespace

• Possible uses:
  ▪ D/W for loading summarized tables (MQTs).
  ▪ Loading data into a table and making changes to the data with SQL before using the table.
  ▪ Changing a lot of data in a controlled process such as for year-end processing.

• If anything goes wrong during the ‘not logged period’ then you need to recover from the previous image copy or reload the data.

• Do not sacrifice data recoverability for performance. In most cases impact on performance is not noticeable for not logged tablespaces.
Not Logged Tablespace

- CREATE TABLESPACE...NOT LOGGED
- ALTER TABLESPACE...NOT LOGGED
- SYSIBM.SYSCOPY
  - ICTYPE = A
  - STYPE = O, logging attribute altered to NOT LOGGED
  - STYPE = L, logging attribute altered to LOGGED
- SYSIBM.SYSTABLESPACE
  - LOG = YES, logging attribute is Logged
  - LOG = NO, logging attribute is Not Logged
- Consider auditing SYSTABLESPACE.

DB2ADM authority can turn logging off.
Consider auditing SYSTABLESPACE for not logged tables using the following SQL:

SELECT NAME, DBNAME, LOG FROM SYSIBM.SYSTABLESPACE
WHERE LOG = 'N';
Not Logged Tablespace

- Recoverable Points for a Not Logged Tablespace.
  - To the point of ALTER TABLESPACE NOT LOGGED.
  - Image Copy SHRLEVEL REFERENCE.
- LPL/RECP Status:
  - Deadlock
  - Timeout
  - Any abend
- Automatic LPL recovery is not initiated for not logged tablespace.
- -START DATABASE has no effect on the LPL status of a not logged tablespace.
After turning logging back on the tablespace will be in COPY pending status. Avoid ROLLBACK in applications as this will put the not logged tablespace in LPL status and RECP (recover pending).

Recoverable points for a not logged tablespace are either the point at which ALTER TABLESPACE NOT LOGGED occurred or an Image Copy SHRLEVEL REFERENCE.
ALTER TABLESPACE NOT LOGGED places an entry into SYSCOPY and switches the LOG column of SYSTABLESPACE.

When logging is turned back on for a tablespace, SYSCOPY will be updated with ICTYPE=A, STYPE=L, LOGGED=Y.
Not Logged Tablespace

The first update to the tablespace after setting the ‘Not Logged’ attribute places the tablespace in ICOPY status.

VRATS903 TS RW,ICOPY

Rollback or timeout results in the tablespace being placed in RECP status.

VRATS903 TS RW,RECP,LPL,ICOPY

Subsequent access to the table will result in SQLCODE -904 and the table will need to be recovered to a ‘recoverable point’.
Not Logged Tablespaces are like an Excavator

- You can dig yourself a really big hole.
Database Roles

- Groups together one or more privileges that can be assigned to users.
- A virtual authid.
- User assigned only to one role.
- Roles can own objects.
- Establish through a trusted context.

The following DB2 catalog tables contain information about roles and trusted contexts:
SYSROLES, SYSCONTEXT, SYSOBJROLEDEP, SYSCTXTRUSTATTRS, SYSCONTEXTAUTHIDS
The following DB2 catalog tables contain information about roles and trusted contexts:
SYSROLES, SYSCONTEXT, SYSOBJROLEDEP, SYSCTXTRUSTATTRS, SYSCONTEXTAUTHIDS

The relationship between a connection and a trusted context is established when a connection to the server is first created.
Roles and Trusted Context

- The issue at John Deere is to replace a shared userid that has SYSADM authority with individual userids that have SYSADM roles.
- Create the database role:
  CREATE ROLE SAPADM_ROLE;
  GRANT SYSADM to ROLE SAPADM_ROLE;

In this example a new role called SAPADM_ROLE is being defined. The SAPADM_ROLE is being assigned SYSADM authority on this DB2 subsystem.
Audit needs to be more detailed now for individual accountability. Roles help.
A trusted context can only be created if the role exists otherwise the create command fails with a SQL error -204.

You can use the ALTER command to add attributes to a trusted context.

In this example a trusted context is being defined for the BP51498 primary authid. The connection attributes are for a batch job called X96SELECT and for a tso userid BP51498 that will be used for SPUFI. The default role for BP51498 is SAPADM_ROLE and the role can be the object owner for DB2 objects.
Roles and Trusted Context

- If jobname or tso user id is not an attribute on the trusted context, you will get SQL error -551.
- Original design did not allow for wild carding of jobnames in the trusted context.
- SAPADM_ROLE will be the ‘grantor’ when granting authorities.
  - Audit trace Class 2 will show detailed information for grant and revoke.
- SAPADM_ROLE will own the created objects.

There are many other possible uses for database roles and trusted contexts, such as for an application server. This was the only scenario that John Deere tested as part of the DB2 9 ESP.
DB2 9 has introduced a new column, OWNER into the catalog tables (SYSIBM.SYSTABLES, SYSIBM.SYSINDEXES, etc.). This column along with column CREATEDBY are for documentation only. The CREATOR column is the column that is used as the high level qualifier when an unqualified object is accessed.
Roles and Trusted Context

- Role cannot be dropped if a trusted context depends on it. (SQL -478)
- Role can not be dropped if DB2 objects exist that are owned by the role. (SQL -478)
- Revoking a user’s access to a role does not cause cascade delete.
- Redbook SG24-6480, ‘Securing DB2 and Implementing MLS on z/OS’
Able to pull a lot of different equipment and tools.

Roles and Trusted Contexts have much more depth and capabilities than just the one example that I have demonstrated here.
CPU reductions are very substantial, with some customers experiencing 20% to 30% savings. More online utilities are coming! Online rebuild builds an index in place after setting RBDP. Dynamic SQL will avoid the index until available. This works well for non-unique indexes. Online reorg solves the outage during the BUILD2 phase when reorging a partition at a time. Online Check Data and Check Lob are the same as existing utilities, only with SHRLEVEL CHANGE. Check has the SHRLEVEL CHANGE technique extended to SHRLEVEL REFERENCE.
COPY using Template Switching

- Provides the ability to direct DB2 image copy datasets to either disk or tape based on size.
- At John Deere, it will replace an internal usermod that puts ‘large’ image copies to tape instead of disk.
In template DB2CPYDK, the LIMIT(20 CYL,DB2CPYTP) parameter instructs the DB2 COPY utility to image copy a tablespace with a size up to and including 20 cylinders to disk. A table space with a size over 20 cylinders will use template DB2CPYTP, which will put the image copy dataset to tape.

**CALCULATION:**

Using DA296 tablespace VRADB002.VRATS901 as tablespace to backup. The corresponding tablespace dataset occupies 49 cylinders per a listcat of the dataset:

**VOLUME**

<table>
<thead>
<tr>
<th>VOLSER</th>
<th>PHYREC-SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SH9D50</td>
<td>4096</td>
</tr>
</tbody>
</table>

**ALLOCATION**

<table>
<thead>
<tr>
<th>SPACE-TYPE</th>
<th>HI-A-RBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYLINDER</td>
<td>36126720</td>
</tr>
<tr>
<td>49</td>
<td>2457600</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Per IBM doc:

- **Cyl**: HFPunit / 184
- **MB**: HFPunit / 256
- **GB**: HFPunit / 262144

Convert the H-U-RBA into 4k pages: 2457600 / 4096 = 600.

600 / 184 = 3.2 Cyl - rounded down to 3 CYL

600 / 256 = 2.3 MB – rounded down to 2 MB

**Note** *** all calculations are truncated not rounded. So, for example a value of 3.8 cylinders would truncate to 3 cylinders.
COPY using Template Switching

//DA96UTIL.SYSTEMPL DD
 DSN=DXX.D945.DB2.TEMPLATE.CNTL(D945V9),
 DISP=SHR
//DA96UTIL.SYSIN DD *

COPY TABLESPACE DSNDB06.SYSDBASE
COPYDDN(DB2CPYDK)
SHRLEVEL REFERENCE

JCL will use the DB2CPYDK template specified in COPYDDN to copy the tablespace to disk. Based on the template, if the image copy exceeds the limit in DB2CPYDK then the image copy will use the DB2CPYTP template and be put to tape.
Template Switching is like a Gator

- A utility vehicle.
**RECOVER...RESTOREBEFORE**

- Used to bypass an image copy as the recovery base. (Available in CM.)

**SYSCOPIY:**

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>ICDATE</th>
<th>START_RBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRADB003</td>
<td>VRATS903</td>
<td>070104</td>
<td>00008090E9C5</td>
</tr>
<tr>
<td>VRADB003</td>
<td>VRATS903</td>
<td>070104</td>
<td>0000808E4AF7</td>
</tr>
</tbody>
</table>

RECOVER TABLESPACE VRADB003.VRATS903
RESTOREBEFORE X'00008090E9C5'

---

Allows you to recover an object but avoid using a specific image copy as the base. This is an alternative to restoring to an older image copy using DSN1COPY and then running RECOVER LOGONLY.

Example SYSOUT:

THE IMAGE COPY DATA SET
DXX.D945.@DA96ALL.VRADB003.VRATS903.G0006V00

RECOVER UTILITY LOG APPLY RANGE IS
RBA 0000808F21FA LRSN 0000808F21FA TO
   RBA 0000808F4F63 LRSN 0000808F4F63
RECOVER UTILITY LOG APPLY RANGE IS RBA 000080918878 LRSN 000080918878 TO
   RBA 000080BFE62E LRSN 000080BFE62E
RECOVER TORBA/TOLOGPOINT with data consistency

- RECOVER utility with TORBA or TOLOGPOINT is to a consistent point.
  - Uncommitted transactions are rolled back.
  - Don’t need to use the QUIESCE utility

SYSCOPY:

```
+---------------------------------------------+
<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>ICDATE</th>
<th>ICTIME</th>
<th>ICTYPE</th>
<th>STYPE</th>
<th>SHRLEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRADB003</td>
<td>VRATS903</td>
<td>070105</td>
<td>141616</td>
<td>P</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>VRADB003</td>
<td>VRATS903</td>
<td>070105</td>
<td>140909</td>
<td>P</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>VRADB003</td>
<td>VRATS903</td>
<td>070105</td>
<td>134340</td>
<td>P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

RECOVER TORBA or TOLOGPOINT in DB2 9 will be to a consistent point.
RECOVER TOCOPY, TOLASTCOPY and TOLASTFULLCOPY using SHRLEVEL CHANGE will continue to work as they did in DB2 Version 8.
REBUILD INDEX SHRLEVEL CHANGE

- Online index rebuild.
- Allows DML concurrency on a table with REBUILD utility executing against indexes.
- Index being rebuilt is a non-unique index.
  - Unique index can result in -904’s.
- Useful in a SAP environment for adding indexes.

Restriction:
SHRLEVEL CHANGE is not well suited for unique indexes and concurrent DML because the index is placed in RBDP while being built. Inserts and updates of the index will fail with a resource unavailable (-904) because uniqueness checking cannot be done while the index is in RBDP. SHRLEVEL CHANGE is not allowed for not logged tables, XML indexes, or spatial indexes.
CATMAINT – UPDATE VCAT

- VCAT name can be switched to a different name using a new option of the CATMAINT utility.
- CATMAINT UPDATE VCAT SWITCH(DB2X96, DB2X92)
- VCAT names for all table spaces, indexes and stogroups were switched in 9 minutes.
  - SAP system with 60,000 objects.
- Useful for cloning SAP systems.
  - We will replace an internally developed process that took over two hours to execute.

CATMAINT should be run under Install Sysadm user and DB2 should be in maintenance mode.

CATMAINT can also be used for updating SCHEMA and changing OWNER to a database role.
CATMAINT – UPDATE SCHEMA

- Change schema names using CATMAINT.
- CATMAINT UPDATE SCHEMA SWITCH(SAPMDS,SAPR3)

CATMAINT should be run under Install Sysadm user and DB2 should be in maintenance mode.

CATMAINT can also be used for updating SCHEMA and changing OWNER to a database role.
The new stuff, things we have found or done since the DB2 9 ESP

- Reordered Row Format (RRF)
- Think REBIND
- Performance Apars of interest
- Spatial data
Reordered Row Format

- BRF – Basic Row Format
  - Currently existing row format prior to DB2 9 NFM.
  - VARCHAR columns intermingled within the row.

- RRF – Reordered Row Format
  - Must be in DB2 9 NFM.
  - VARCHAR columns are placed at the end of the row.
  - Subsequent REORGs convert BRF to RRF automatically.
  - Controlled by a hidden zparm.
  - UTS (Universal Table Spaces) created as RRF.
  - Most likely will not get as good a compression ratio with RRF compared to BRF. However, RRF provides improved CPU consumption.
### BRF & RRF

**Basic Row Format**

<table>
<thead>
<tr>
<th>Row Header</th>
<th>CHAR (4)</th>
<th>L</th>
<th>VARCHAR (20)</th>
<th>CHAR (25)</th>
<th>L</th>
<th>VARCHAR (10)</th>
</tr>
</thead>
</table>

**Reordered Row Format** *(same length as BRF)*

<table>
<thead>
<tr>
<th>Row Header</th>
<th>CHAR (4)</th>
<th>CHAR (25)</th>
<th>D1</th>
<th>D2</th>
<th>VARCHAR (20)</th>
<th>VARCHAR (10)</th>
</tr>
</thead>
</table>
RRF (Reordered Row Format) Apars

- **PK79127** (UK44075)
  - 00C90101 ERQUAL500F BRF TO RRF CONVERSION REORG LOG PHASE TO AVOID THE UNSUPPORTED CONVERSION PATH IN DM.
    - REORG with KEEPDICTIOANRY specified

- **PK78958** (UK45353) and **PK85985** (UK47287)
  - DISABLE RRF CONVERSION FOR COMPRESSED PAGESETS
    - Uncompressed pagesets use RRF with zparm enabled.
    - Compressed pagesets use BRF with zparm enabled.

- **PK78959** (UK45881)
  - REORG SERVICEABILITY ENHANCEMENT. (Get you back to BRF from RRF, used with IBM supervision.)

- **PK83585** -- THIS APAR HAS BEEN CANCELLED.
  - ENSURE UTS TABLESPACES ARE CREATED IN RRF FORMAT.
RRF (Reored Row Format) Apars

- PK87348 (UK50412)
  - Enable Basic Row Format for Universal Tablespaces
  - Hidden DSNZPARM SPRMRRF converted to opaque.
  - Default remains SPRMRRF=ENABLE

- PK85881 (UK50413)
  - LOAD/REORG ROWFORMAT SUPPORT
  - This APAR will enable the new ROWFORMAT option for LOAD and REORG TABLESPACE utility
  - Could convert individual page sets.
  - Typically you would use the zparm.
RRF (Reordered Row Format)

- If you are not experiencing any issues with RRF and compression ratios then IBM’s recommendation is to leave RRF enabled.
- With zparm enabled and PK87348 applied, compressed pagesets will be converted to RRF.
Think REBIND

- SQL Code -501
  - The cursor identified in a FETCH or CLOSE statement is already open.

- SQL Code -502
  - The cursor identified in an OPEN statement is already open.

- Try REBIND
Performance Apars of Interest

- PK84244 (UK47911)
  - IF SPARSE INDEX ACCESS ON A TABLE EXPRESSION OR VIEW, RESULT SET COULD BE INCORRECT OR ABEND04E RC00C90101 AT DSN$IIMS$1:5008

- PK79832 (UK45641)
  - DISCOURAGE NESTED LOOP JOIN WITH TABLESPACE SCAN OF THE INNER TABLE WHEN THE INNER TABLE IS A WORKFILE

- PK79771 (UK44494)
  - UNDERESTIMATED MATCHING FILTER FACTOR FOR INDEX COLUMNS WITH 0 MATCHING MULTI COLUMN FREQUENCY AND CARDINALITY STATS

- PK81852 (UK47375)
  - More rows found when FETCH FIRST N ROWS in subselect with parallelism, incorrect output.
Performance Apars of Interest

- PK88774 (UK48569)
  - INDEX COST UNDERESTIMATED FOR A DPSI OR PI WHEN ALL PARTITIONS QUALIFY

- PK61277 (UK39140)
  - Provide an option to adjust the CPU/IO cost balance for use by the optimizer (OPTIOWGT).

- PK75643 (UK42565)
  - Change default of OPTIOWGT from DISABLE to ENABLE.
Performance Apars of Interest

- PK82356 (UK45908)
  - VERY LONG PREPARE TIME AND BAD ACCESS PATH FOR A STAR JOIN QUERY
    - Complex SQL statement gets -101
    - 5X virtual storage increase on the DB2 subsystem
Spatial Exploitation with DB2 9

- The following slides are taken from IBM IOD 2009 Session 1752, “Spatial Support on DB2 9 for z/OS – Customer Experiences at John Deere” presented by Sooraj Ali (John Deere) and Frank Butt (IBM).
CKC – Customer Knowledge Center
Data warehouse

- Customer (Person & Organization) centric data warehouse
- Primarily used by the marketing/sales division
- Database on DB2 z/OS V9, Size – 4TB+
- Customer table - 26 Million+ rows
- Contains all Persons & Organizations who we are connected with and their Business events

CKC is part of the DB2 spatial warehouse.
CKC Data warehouse is the Business Intelligence system used primarily by Sales & Marketing process of John Deere across all divisions in NA.
Why did John Deere decide to spatially enable its Customer Knowledge Center DW?

- Enables better understanding of a marketplace through rich spatial visualization of data and analytics
- Identification of underlying trends and opportunities leading to better planning, measuring and monitoring of customer, business, enterprise
- Allow disparate data sources to be taken into consideration, helps understand and discover underlying relationships.

GIS technology produces maps and we use to visualize the analysis
DB2 9 requirements for spatial enablement

- Install IBM Spatial Support – This is part of DB2 Accessories Suite
  - APARS (PK51020, PK00024, PK00137, PK50034, PK00324)

- z/OS Language environment should have PTF UK14217 or UK14218 applied

- DB2 should be at RSU0812 level and PK76265 applied

Let's examine the high level steps involved in the db2 zOS setup
ArcSDE setup

- Install DB2 Connect 9.1 on desktop
- Install ArcSDE 9.3 for DB2 z/OS on desktop
- Define SDE UDF and SDE databases on DB2 z/OS
- Define the ArcSDE geodatabase repository in DB2 z/OS

Here are the steps involved in setting up the ArcSDE
Spatial Tables and Views created by the install

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Full Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>DB2GSE.GEOMETRY_COLUMNS</td>
<td>View</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.GSE_COORDINATE_SYSTEMS</td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.GSE_COORDINATE_SYSTEMS_ID</td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.GSE_GEOGRAPHY_COLUMN</td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.GSE_GEOGRAPHY_COLUMN</td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.GSE_SPATIAL_REFERENCE_SYSTEMS</td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.GSE_UNITS_OF_MEASURE</td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.SPATIAL_REF_SYS</td>
<td>View</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.ST_COORDINATE_SYSTEMS</td>
<td>View</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.ST_GEOGRAPHY_COLUMN</td>
<td>View</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS</td>
<td>View</td>
</tr>
<tr>
<td></td>
<td>DB2GSE.ST_UNITS_OF_MEASURE</td>
<td>View</td>
</tr>
</tbody>
</table>
Objects created by SDE process in DB2
Create a spatial column in the database

- Add column, CUST_LOC
  - Data type as ST_point
  - Other supported data types – ST_lineString, ST_Polygon

- Update column, CUST_LOC with function
  - CUST_LOC = DB2GSE.ST_POINT(DOUBLE(LONITUDE), DOUBLE(LATITUDE), 1)

- Now you can display your customer location in a map and spatially analyze it!

Update CUST_LOC table takes xx hours
Definitions on geo spatial concepts.
Case Study – Sales Opportunity

• Business Problem
  • Identify sales opportunities for a product line

• Our Approach
  • Map sales data for product line by county
  • Mash up competitive sales for similar product by county
  • Identify the density of sales
Market activity - combined
**Best Practices for Spatial Enablement**

- Based on experiences at John Deere, other customer proof of concept activities, and internal testing in the IBM labs, IBM has put together a paper describing some of the setup and configuration recommendations for customers. This paper and other Spatial related presentations can be found in the following website:
  - [http://www-01.ibm.com/software/data/spatial/resources.html](http://www-01.ibm.com/software/data/spatial/resources.html)
  - Title: DB2 for z/OS Spatial Support And ESRI Enablement
I hope these pictures help you remember the key features.

✓ Clone Tables are like an End Loader. You can dump new data into them without affecting your production table.

✓ Partition by Growth is like a Motor Grader. DB2 will spread your data across the partitions.

✓ Index Compression is like a Combine. You will be able to harvest disk savings.

✓ Not Logged Tablespaces are like an Excavator. You can dig yourself a really big hole.

✓ Trusted Contexts are like a Tractor. Versatile, you are able to pull a lot of different equipment with them.

✓ Template Switching is like a Gator, a utility vehicle.
Bryan Paulsen is a technology architect for John Deere in Moline, IL. He specializes in DB2 for z/OS in the DB2 Classic/Legacy and DB2 SAP environments. His experiences include DB2 systems programming, project manager for the DB2 10, DB2 9 and DB2 V8 beta programs, DB2 disaster recovery technical lead and project manager for migrating John Deere’s SAP implementations to SAP on DB2 for z/OS. He is a founding member of the z/SAP Very Large Database (VLDB) Technical Exchange; an organization of companies running SAP on DB2 for z/OS.

Bryan has spoken at IDUG North America, IDUG Europe, IBM’s Information On Demand, SHARE, Central Canada DB2 User Group, Colorado DB2 User Group and several IDUG Regional Forums. He received the Best User Speaker Award for IDUG North America 2004, 2007, 2009 and IDUG Europe 2007. He is a member of the IDUG Speakers Hall of Fame. He was recognized as an IBM Information Management Champion for 2009 and 2010.

Bryan has served on the IDUG North America Conference Planning Committee since 2007. He is the IDUG North America 2011 Conference Chair.