DB2 9 & 10 for z/OS
Overview

Frank Merino
Sr. IT Specialist
DB2 Advisor
East IMT Extended Technical Team
fmerino@us.ibm.com
Disclaimer and Trademarks

Information contained in this material has not been submitted to any formal IBM review and is distributed on "as is" basis without any warranty either expressed or implied. Measurements data have been obtained in laboratory environment. Information in this presentation about IBM's future plans reflect current thinking and is subject to change at IBM's business discretion. You should not rely on such information to make business plans. The use of this information is a customer responsibility.

IBM MAY HAVE PATENTS OR PENDING PATENT APPLICATIONS COVERING SUBJECT MATTER IN THIS DOCUMENT. THE FURNISHING OF THIS DOCUMENT DOES NOT IMPLY GIVING LICENSE TO THESE PATENTS.

TRADEMARKS: THE FOLLOWING TERMS ARE TRADEMARKS OR ® REGISTERED TRADEMARKS OF THE IBM CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES: AIX, DATABASE 2, DB2, Enterprise Storage Server, FICON, FlashCopy, Netfinity, RISC, RISC SYSTEM/6000, System i, System p, System x, System z, IBM, Lotus, NOTES, WebSphere, z/Architecture, z/OS, zSeries

The FOLLOWING TERMS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF THE MICROSOFT CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES: MICROSOFT, WINDOWS, WINDOWS NT, ODBC, WINDOWS 95

For additional information see ibm.com/legal/copytrade.phtml
Major Presentation Sections

- SQL Enhancements
- Native XML Support… pureXML
- Temporal Data – Time Travel Query
- LOB Enhancements
- Advanced Design Options
- Security & Administration for Today’s Business
- Optimization Evolution
- Index Enhancements

- Distributed Data Performance
- Utility Enhancements
- Virtual Storage Improvements
- Serviceability & Instrumentation
- DB2 for z/OS Tooling
- DB2 for z/OS Documentation
DB2 10 Sample Performance Improvements

- **DB2 10 CMx with REBIND**
  - Run time CPU reductions 5% - 10%
  - 1 MB page size 0% - 4% z10, z196
  - Page fix buffers 0% - 8% since V8
  - Release deallocate 0% - 15% short trans, batch
  - Virtual storage constraints 0% - 5% memory, latches
  - Data sharing fewer members 1% for each 2 members
  - Insert 0% - 40% high volume insert
  - Predicate evaluation 0% - 60% complex predicates
  - Increased use of zIIP 0% - 3% IO, RUNSTATS, parallelism
  - Utilities (from V8) 3% - 20%

- **DB2 10 NFM**
  - Improved dynamic SQL cache 0% - 20% literals
  - Access: hash, index include 0% - 5% access improved
DB2 10 Performance...

- A beta customer shared some numbers @ IOD 2010

<table>
<thead>
<tr>
<th></th>
<th>DB2 10 vs DB2 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chargeable Application + DB2 CPU</td>
<td>-40 %</td>
</tr>
<tr>
<td>Chargeable DB2 CPU</td>
<td>-40 %</td>
</tr>
<tr>
<td>Total Application + DB2 CPU</td>
<td>-35 %</td>
</tr>
<tr>
<td>Total DB2 CPU</td>
<td>-36 %</td>
</tr>
<tr>
<td>Application + DB2 Elapsed Time</td>
<td>-26 %</td>
</tr>
<tr>
<td>DB2 Elapsed Time</td>
<td>-34 %</td>
</tr>
<tr>
<td>DB2 Suspension Time</td>
<td>-23 %</td>
</tr>
<tr>
<td>Lock / Latch Suspension Time</td>
<td>-88 %</td>
</tr>
<tr>
<td>Page Latch Suspension Time</td>
<td>-67 %</td>
</tr>
<tr>
<td>Log I/O Suspension Time</td>
<td>-77 %</td>
</tr>
</tbody>
</table>

- Another beta customer (V8 to V10) shared:
  - Average 5 – 9% CPU reduction
  - Some workloads achieved a 30% CPU reduction
  - REORGs elapsed times improved

20 concurrent LOADs
10M rows per job
SHRLEVEL NONE
SQL Enhancements (New Function Mode)

- MERGE
- SELECT FROM
  - UPDATE
  - DELETE
  - MERGE
- FETCH FIRST / ORDER BY in Subselect
- ORDER OF
- INTERSECT & EXCEPT
- TRUNCATE
- Native SQL Stored Procedures
- Extended SQL Procedure Language (SQL PL)
  - SQL Scalar Functions
  - SQL Table Functions
- SKIP LOCKED DATA
- Access Currently Committed
- Extended Indicator Variables
- OLAP Built-in Functions
- More OLAP Specifications
  - Moving Sums and Averages
- SOUNDEX & DIFFERENCE
- Other Built-in Functions
MERGE

“Upsert”
- A combination of insert and update, MERGE is a single SQL operation
- Single values or arrays
- INSERT or UPDATE a row in a table or view based on the rows existence

If a row matches ON specified columns
- it is updated,

If it does not match
- it is inserted.
MERGE

- Array MERGE operation
- Targets OLTP applications like SAP
- Maintains trigger integrity
- **NOT ATOMIC CONTINUE ON SQL EXCEPTION** is optional, but must be specified if inserting multiple rows

```sql
MERGE INTO account AS T
USING VALUES (:hv_id, :hv_amt) FOR 5 ROWS AS S(id, amt)
ON T.id = S.id
WHEN MATCHED THEN
  UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
  INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQL EXCEPTION;
```
Select from Update, Delete, and Merge

- **Benefits**
  - Builds on SELECT from INSERT delivered in V8
  - Enables user to efficiently determine values updated into or deleted from tables.
  - Cuts down on network cost by reducing the number of SQL statements used to accomplish the same thing.

- **SELECT from MERGE**
  - Returns all rows that were either updated or inserted, including generated column values.

- **INCLUDE column specification**
  - A list of columns to be included in the result table of a Delete/Insert/Update/Merge statement.
  - Only available if the statement is nested in the FROM clause of a SELECT or SELECT INTO statement.
Select from Update, Delete, and Merge

- Builds on SELECT FROM INSERT delivered in V8
- Like the INSERT statement, the FROM clause of a SELECT statement will now allow an UPDATE, DELETE, or MERGE statement:

```
FROM
  single-table
  nested-table-expression
  table-function-reference
  data-change-table-reference
  joined-table

FINAL TABLE
  ( INSERT statement )
  ( searched UPDATE statement )
  ( searched DELETE statement )
  ( MERGE statement )
```

FINAL TABLE
  TABLE
  OLD TABLE
  FINAL TABLE

FINAL TABLE
  TABLE
  OLD TABLE
  FINAL TABLE

OLD TABLE
  TABLE
  FINAL TABLE

© 2011 IBM Corporation
Select from Update, Delete, and Merge

- First example shows SELECT returning data from the nested UPDATE.
- Second example shows SELECT returning data from the nested MERGE with an INCLUDE column.
  - This include column returns the compare column from the ON clause and an indicator of whether the row was Updated or Inserted.
FETCH FIRST/ ORDER BY in Subselect

- FETCH FIRST and ORDER BY clauses are now allowed in subselects.
- In this example both clauses are used in the subselect as a way to select the top 10 rows from table T1 as ordered by columns C2 and C1.

```sql
INSERT INTO T2
(SELECT C1, C2, C3
  FROM T1
  FETCH FIRST 10 ROWS ONLY
  ORDER BY C2, C1);
```
ORDER BY ORDER OF

- Preserving the order of a derived table
- The outer most fullselect can inherent the ordering of a nested subselect with the ORDER OF clause of ORDER BY

Inline View

TOP_MONTHLY_TIME _CHARGES contains an ORDER BY

The outer SELECT then uses a new DB2 9 OLAP function to assign row numbers.

The outer SELECT inherits the ordering of the subquery
Intersect & Except

- **Intersect** – Returns the records that two result sets have in common
  - Frequently used in ‘find the duplicates query’
  - Rows from either result set that do not match are excluded.

- **Except** – Returns the records that are members of one result set that are not members of the other

- **Requirement**
  - Both result sets must have the same column structure.

For 2 given result sets…

RS1  RS2

Intersect

RS1  RS2

Except
# INTERSECT & EXCEPT Comparisons

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPT</td>
<td>DEPT2</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>DEPTNO</td>
</tr>
<tr>
<td>UNION ALL</td>
<td>INTERSECT ALL</td>
</tr>
<tr>
<td>UNION DISTINCT</td>
<td>INTERSECT DISTINCT</td>
</tr>
<tr>
<td>INNER JOIN</td>
<td>EXCEPT ALL</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>DEPTNO</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>DEPTNO</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>DEPTNO</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>DEPTNO</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>DEPTNO</td>
</tr>
<tr>
<td>DEPTNO</td>
<td>DEPTNO</td>
</tr>
</tbody>
</table>

- **A00**
  - A00
  - A00
  - A00
  - A00
  - A00
  - 1 record(s) selected
- **A00**
  - A00
  - A00
  - 1 record(s) selected
- **A00**
  - B01
  - B01
  - D11
  - B11
  - 2 record(s) selected.
- **B01**
  - A00
  - D01
  - 2 record(s) selected.
- **C01**
  - E01
  - D11
  - 2 record(s) selected.
- **D01**
  - E01
  - D11
  - 2 record(s) selected.
- **D01**
  - F22
  - D21
  - 2 record(s) selected.
- **E01**
  - C01
  - E21
  - 2 record(s) selected.
- **E11**
  - G22
  - H22
  - 2 record(s) selected.
- **F22**
  - I22
  - J22
  - 2 record(s) selected.
- **G22**
  - I22
  - J22
  - 2 record(s) selected.
TRUNCATE

- Allows fast delete of all rows in base tables or declared global temporary tables
  - Simple, segmented, partitioned, universal table spaces or tables
  - If table contains LOB or XML columns the corresponding table spaces and indexes are also truncated.
  - IMMEDIATE option – operation cannot be rolled back
    • Allows reusing the allocated space immediately for subsequent insert operations in the same unit of work without committing.
    • Not allowed if the table has in-flight modifications

- Deletes rows without firing DELETE triggers

- Option to REUSE or DROP STORAGE
  - Useful for nightly refresh of summary tables, warehouses, etc.
Native SQL Stored Procedures

- Eliminates generated C code and compilation
- Fully integrated into the DB2 engine
  - Any SQL procedure created without the FENCED or EXTERNAL keywords are native SQL procedures
- zIIP enabled for DRDA clients
- Extensive support for versioning:
  - VERSION keyword on CREATE PROCEDURE
  - CURRENT ROUTINE VERSION special register
  - ALTER ADD VERSION
  - ALTER REPLACE VERSION
  - ALTER ACTIVATE VERSION
- BIND PACKAGE with new DEPLOY keyword
Extended SQL Procedure Language (SQL PL)

- Enhanced Native SQL Procedures capabilities:
  - Can define an SQL parameter or SQL variable as a distinct type or XML data type
  - Makes it easier to port applications from other DBMS
  - DB2 10 NFM
Extended SQL Procedure Language (SQL PL)...

- **SQL Scalar Function:**
  - DB2 10 NFM introduces two types of SQL Scalar Functions: Inline and Non-inline
  - Inline SQL scalar functions are similar in structure to SQL scalar functions prior to DB2 10 with a single RETURN statement, and can use the XML data type in DB2 10 NFM.
  - Non-inline SQL scalar functions have added benefits:
    - Can contain logic using SQL statements and SQL control statements like those used in native SQL procedures. Will create a package.
    - Allow for debugging using the Unified Debugger, like what is used for debugging native SQL procedures.
    - Can replace the use of external scalar functions by:
      - Simplifying development
      - Address performance considerations
Extended SQL Procedure Language (SQL PL)...

- SQL Scalar Function... Examples:
  - **Inline**: Defines a scalar function that returns a duration in years.

```sql
create function years(x date)
returns smallint
language sql
contains sql
no external action
not deterministic
return year(current date - x) ;
```

```sql
select current date from sysibm.sysdummy1

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>LASTNAME</th>
<th>HIREDATE</th>
<th>SERVICE_LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>000010</td>
<td>HAAS</td>
<td>1965-01-01</td>
<td>46</td>
</tr>
<tr>
<td>000110</td>
<td>LUCCHESI</td>
<td>1958-05-16</td>
<td>52</td>
</tr>
<tr>
<td>000120</td>
<td>O'CONNELL</td>
<td>1963-12-05</td>
<td>47</td>
</tr>
<tr>
<td>200010</td>
<td>HEMMINGER</td>
<td>1965-01-01</td>
<td>46</td>
</tr>
<tr>
<td>200120</td>
<td>ORLANDO</td>
<td>1972-05-05</td>
<td>38</td>
</tr>
</tbody>
</table>
```

**Reference Date**: 2011-01-06
Extended SQL Procedure Language (SQL PL)...

- **SQL Scalar Function… Examples:**
  
  **Non-inline:** Defines a scalar function that reverses a string.

```sql
CREATE FUNCTION REVERSE (REVSTR VARCHAR(100))
RETURNS VARCHAR(100)
DETERMINISTIC
NO EXTERNAL ACTION
CONTAINS SQL
BEGIN
  DECLARE IN_STR CHAR(100) DEFAULT ' ';
  DECLARE LEN INT;
  IF IN_STR IS NULL THEN
    RETURN NULL;
  END IF;
  SET (IN_STR, LEN) = (REVSTR, LENGTH(IN_STR));
  WHILE LEN > 0 DO
    SET OUT_STR = SUBSTR(IN_STR, 1, 1) || OUT_STR;
    SET IN_STR = SUBSTR(IN_STR, 2);
    SET LEN = LEN - 1;
  END WHILE;
  RETURN OUT_STR;
END
```

```sql
SELECT LASTNAME ,EMPNO ,REVERSE(EMPNO) AS ONPME
FROM DSN81010.EMP
WHERE WORKDEPT = 'A00'
```

<table>
<thead>
<tr>
<th>LASTNAME</th>
<th>EMPNO</th>
<th>ONPME</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAAS</td>
<td>000010</td>
<td>010000</td>
</tr>
<tr>
<td>LUCCHESI</td>
<td>000110</td>
<td>011000</td>
</tr>
<tr>
<td>O'CONNELL</td>
<td>000120</td>
<td>021000</td>
</tr>
<tr>
<td>HEMMINGER</td>
<td>200010</td>
<td>010002</td>
</tr>
<tr>
<td>ORLANDO</td>
<td>200120</td>
<td>021002</td>
</tr>
</tbody>
</table>
Extended SQL Procedure Language (SQL PL)...

- **SQL Table Function:**
  - DB2 10 NFM introduces ability to create SQL table functions.
  - Written using SQL statements and includes a single RETURN statement that returns a single result table (a set of rows).
  - Parameters can be defined as:
    - Built-in or distinct data types
    - Can be defined to behave as parameterized views
Extended SQL Procedure Language (SQL PL)…

- SQL Table Function… Example:
  Defines a table function that returns a list of employees for a department.

```sql
CREATE FUNCTION DEPEMPLOYEES (DEPTNO CHAR(3))
RETURNS TABLE (EMPNO CHAR(6),
                 ,LASTNAME VARCHAR(15),
                 ,FIRSTNAME VARCHAR(12))

LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
DETERMINISTIC
RETURN

SELECT EMPNO ,LASTNAME ,FIRSTNAME
FROM DSN81010.EMP E
WHERE E.WORKDEPT = DEPEMPLOYEES.DEPTNO
;
```

```
SELECT * FROM TABLE (DEPEMPLOYEES ('A00')) AS T

EMPNO  LASTNAME  FIRSTNAME
-------  ----------  ---------
000010  HAAS      CHRISTINE
000110  LUCCHESI  VINCENZO
000120  O'CONNELL SEAN
200010  HEMMINGER DIAN
200120  ORLANDO   GREG
```
SKIP LOCKED DATA

- Rows with incompatible locks by other transactions are skipped

- Clause available
  - On SELECT INTO, PREPARE, searched UPDATE, searched DELETE, UNLOAD
  - Effective when CS or RS is in use
    - Otherwise it is ignored
  - Data is locked at the row or page

- QW0018SK – ROWS SKIPPED DUE TO INCOMPATIBLE LOCK HELD
  - Reported in IFCID 018

- Logic / Scenario
  - When a transaction needs to find work to do, regardless of order.
  - Messaging applications without strict ordering requirements expect to be able to skip over records that are locked by other transactions
Access Currently Committed…

- **SELECT not blocked by INSERT:**
  - Pre-DB2 10 SELECT waits and eventually a row returned or times out.
  - DB2 10 can see row being inserted is not committed, and would immediately skip row.

- **SELECT not blocked by DELETE:**
  - Pre-DB2 10 SELECT waits and eventually no row is returned or times out.
  - Currently committed row, including any LOB or XML data is returned, until DELETE is committed.
Access Currently Committed…

- **SELECT** bound with **ISOLATION(CS) or ISOLATION(RS) and CONCURRENTACCESSRESOLUTION(USECURRENTLYCOMMITTED)**

- **Syntax specifications:**
  - **BIND/REBIND PACKAGE and PLAN:**
    - Specify option **CONCURRENTACCESSRESOLUTION (USECURRENTLYCOMMITTED) or (WAITFOROUTCOME)**
  - **PREPARE attribute string:**
    - Specify **USE CURRENTLY COMMITTED or WAIT FOR OUTCOME**
  - **CREATE/ALTER PROCEDURE or FUNCTION**
    - Specify option **CONCURRENTACCESSRESOLUTION (USECURRENTLYCOMMITTED) or (WAITFOROUTCOME)**
    - Can be specified for new and existing Native SQL Procedures
  - **DRDA supports use of CONCURRENTACCESSRESOLUTION bind option, and can be used for SQLJ applications.**
Access Currently Committed…

- **DB2 10 Solution Restrictions:**
  - Support for uncommitted INSERT and DELETE
  - Reader will still wait for uncommitted UPDATE
  - Not applicable when:
    - LOCK TABLE IN EXCLUSIVE used
    - Lock holder is performing mass delete
    - If lock holder escalates

- **DB2 10 Solution Requirements:**
  - Universal Table Space
  - Row or Page-level locking
    - IRLM will remember first 8 locked rows when using page-level locking.
  - NFM
Access Currently Committed…

- **Consider as second generation lock avoidance.**
  - DB2 may revert to unconditional locking.
  - Alternative to DB2 9 “SKIP LOCKED DATA” option, which ignores row.
  - Alternative to ISOLATION(UR)

- **QMF global variable DSQEC_CON_ACC_RES can be used to support feature.**
Extended Indicator Variables

- **Pre-DB2 10 Problem/Requirements:**
  - No way to tell DB2 what value to use in an INSERT, UPDATE or MERGE when values for all the columns are not specified.
  - Need to define all possible SQL combinations… very tedious
  - Very low use of dynamic statement cache.

- **DB2 10 Solution:**
  - DB2 10 NFM introduces extended indicator variables to specify what value should be used for columns when a value is not provided in an INSERT, UPDATE or MERGE.
    - New values used for existing indicator variables.
  - Simplifies application development… no need to define all possible SQL combinations
  - Improves overall performance
Extended Indicator Variables

**Implementation:**

- EXTENDEDINDICATOR YES/NO option on BIND/REBIND PACKAGE
  - NO default for BIND.
  - Existing setting is default for REBIND
  - New EXTENDEDINDICATOR column added to SYSPACKAGE

- WITH EXTENDED INDICATORS or WITHOUT EXTENDED INDICATORS attribute string on PREPARE
  - Default - WITHOUT EXTENDED INDICATORS

- Various methods available for implementing support with JDBC, SQLJ and ODBC applications.

- Extended indicator variable values:
  - -5 specifies the DEFAULT value – set column to the default value
  - -7 specifies the UNASSIGNED value - treat as column was not specified
Extended Indicator Variables … Examples

- Assume the following table definition:

```sql
CREATE TABLE T1 (
  C1 INT WITH DEFAULT 100,
  C2 INT,
  C3 VARCHAR(6));
```

- Request DB2 to use a `DEFAULT` value be used for an INSERT:

```sql
IND1 = -5;
EXEC SQL INSERT INTO T1 VALUES(:HV1 :IND1, 999, 'AAAA');
```

  - DB2 will use the default value for C1.

- Request DB2 to consider a value `UNASSIGNED` for an UPDATE:

```sql
IND1 = -7;
CHARHV1 = 'AAAA';
EXEC SQL UPDATE T1 SET C1=:HV1:IND1, C2=888
  WHERE C3=:CHARHV1;
```

  - DB2 will leave the value of C1 untouched and update C2.
OLAP Built-in Functions

- **RANK**
  - Generates a row number defining the rank as 1 plus the number of rows that precede the row.
  - If two or more rows are not distinct with respect to the ordering, there will be one or more gaps in the sequential rank numbering. Also referred to as Olympic Ranking.

- **DENSE_RANK**
  - Similar to RANK but ensures there are no gaps.

- **ROW_NUMBER**
  - Creates a sequential row number that is computed for each row starting with 1.
  - No number skipping.

- **Even though these functions are ordered, cursors remain updatable.**
OLAP Function Examples

ROW_NUMBER provides a sequential numbering of the rows.

RANK provides an “Olympic” style ranking of the results.

DENSE_RANK does not skip numbers like RANK does.
More OLAP Specifications

- **Moving Sum and Moving Average**
  - New DB2 10 NFM specifications (aggregate functions) that compute a single value for the current row based on some or all the rows in a defined group.
  - Supports cumulative sums and moving averages by using a window.
  - A window can specify three optional components:
    - Partitioning of the result table using PARTITION BY clause (in DB2 9).
    - Ordering of rows within a partition using ORDER BY clause (in DB2 9).
    - Aggregation group by using ROW or RANGE clause
    - Two mandatory components need to be specified for an aggregation group:
      - Keyword ROWS to indicate a physical group … or
      - Keyword RANGE to indicate a logical group … and
      - The starting and ending row of the aggregation group
  - Cannot be used within an XMLQUERY, XMLEXISTS, or as an argument of an aggregate function.
  - New -20117 SQL code to indicate an invalid window specification.
### More OLAP Specifications… Examples

- Assume a `SALES_HISTORY` table containing the following.
  - Each TERRITORY has 5 rows.

<table>
<thead>
<tr>
<th>TERRITORY</th>
<th>MONTH</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>200910</td>
<td>10.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200910</td>
<td>8.00</td>
</tr>
<tr>
<td>EAST</td>
<td>200911</td>
<td>4.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200911</td>
<td>12.00</td>
</tr>
<tr>
<td>EAST</td>
<td>200912</td>
<td>10.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200912</td>
<td>7.00</td>
</tr>
<tr>
<td>EAST</td>
<td>201001</td>
<td>7.00</td>
</tr>
<tr>
<td>WEST</td>
<td>201001</td>
<td>11.00</td>
</tr>
<tr>
<td>EAST</td>
<td>201002</td>
<td>9.00</td>
</tr>
<tr>
<td>WEST</td>
<td>201002</td>
<td>7.00</td>
</tr>
</tbody>
</table>
More OLAP Specifications… Examples

```
SELECT TERRITORY, MONTH, SALES, AVG(SALES) OVER(PARTITION BY TERRITORY ORDER BY MONTH ROWS 2 PRECEDING) AS MOVING_AVG
FROM SALES_HISTORY
```

<table>
<thead>
<tr>
<th>TERRITORY</th>
<th>MONTH</th>
<th>SALES</th>
<th>MOVING_AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>200910</td>
<td>10.00</td>
<td>10.00</td>
</tr>
<tr>
<td>EAST</td>
<td>200911</td>
<td>4.00</td>
<td>7.00</td>
</tr>
<tr>
<td>EAST</td>
<td>200912</td>
<td>10.00</td>
<td>8.00</td>
</tr>
<tr>
<td>EAST</td>
<td>201001</td>
<td>7.00</td>
<td>7.00</td>
</tr>
<tr>
<td>EAST</td>
<td>201002</td>
<td>9.00</td>
<td>8.66</td>
</tr>
<tr>
<td>WEST</td>
<td>200910</td>
<td>8.00</td>
<td>8.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200911</td>
<td>12.00</td>
<td>10.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200912</td>
<td>7.00</td>
<td>9.00</td>
</tr>
<tr>
<td>WEST</td>
<td>201001</td>
<td>11.00</td>
<td>10.00</td>
</tr>
<tr>
<td>WEST</td>
<td>201002</td>
<td>7.00</td>
<td>8.33</td>
</tr>
</tbody>
</table>

Results:
1. Grouped by TERRITORY
2. Ordered by MONTH
3. MOVING_AVG values are computed on the average SALES value of the current row with the prior 2 rows within the TERRITORY.
### More OLAP Specifications... Examples

**SELECT TERRITORY, MONTH, SALES, SUM(SALES)**

**OVER(PARTITION BY TERRITORY ORDER BY MONTH**

**ROWS UNBOUNDED PRECEDING)** AS CUMULATIVE_SUM

**FROM SALES_HISTORY**

<table>
<thead>
<tr>
<th>TERRITORY</th>
<th>MONTH</th>
<th>SALES</th>
<th>CUMULATIVE SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>200910</td>
<td>10.00</td>
<td>10.00</td>
</tr>
<tr>
<td>EAST</td>
<td>200911</td>
<td>4.00</td>
<td>14.00</td>
</tr>
<tr>
<td>EAST</td>
<td>200912</td>
<td>10.00</td>
<td>24.00</td>
</tr>
<tr>
<td>EAST</td>
<td>201001</td>
<td>7.00</td>
<td>31.00</td>
</tr>
<tr>
<td>EAST</td>
<td>201002</td>
<td>9.00</td>
<td>40.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200910</td>
<td>8.00</td>
<td>8.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200911</td>
<td>12.00</td>
<td>20.00</td>
</tr>
<tr>
<td>WEST</td>
<td>200912</td>
<td>7.00</td>
<td>27.00</td>
</tr>
<tr>
<td>WEST</td>
<td>201001</td>
<td>11.00</td>
<td>38.00</td>
</tr>
<tr>
<td>WEST</td>
<td>201002</td>
<td>7.00</td>
<td>45.00</td>
</tr>
</tbody>
</table>

**“ROWS UNBOUNDED PRECEDING” is optional. Results are the same without it.**

**Results:**
1. Grouped by TERRITORY
2. Ordered by MONTH
3. CUMULATIVE_SUM values are computed on the sum of the SALES value of the current row with the prior row(s) within each TERRITORY.
SOUNDEX & DIFFERENCE Scalar Functions

- Powerful scalar functions for name identification

- **SOUNDEX**
  - The SOUNDEX function returns a 4 character code that represents the sound of the words in the argument.
  - The result can be used to compare with the sound of other strings.
  - The SOUNDEX function is useful for finding strings for which the sound is known but the precise spelling is not.

- **DIFFERENCE**
  - A function that returns a value from 0 to 4
  - Represents the difference between the sounds of two strings based on applying the SOUNDEX function to the strings.
    - A value of 4 is the best possible sound match.
SOUNDEX / DIFFERENCE Examples

These functions assist in working with the sound of a string.

A phonetic algorithm is used to assign a 4 character string to a string value.

The goal is for strings with the similar pronunciations to produce the same code.

DIFFERENCE generates a an integer value between 0 and 4 to determine how close two strings might sound. 4 is the best match.
Several Other Functions.....

- ASCII CHR
- ASCII STR
- EBCDIC CHR
- EBCDIC STR
- LOCATE_IN_STRING
- LPAD
- NORMALIZE_STRING
- OVERLAY
- RID
- RPAD
- UNICODE
- UNICODE STR
- Several timestamp functions for porting
Native XML Support… pureXML

- Why XML?
- What is XML
- XML Relational Comparison
- Why use XML with Databases
- Native XML & pureXML
- Native XML Features
- DB2 Objects for XML
- XML Publishing
- Multi-Versioning Support
- Subdocument Update

- Indexing
- XMLTABLE
- Scheme Support & Validation
- Native SQL Procedures & UDFs
- Date & Time Support
- Restrictions and Limits
- Rounding out XML support
- Data Studio support
- Binary XML
Why XML?

- XML is vendor and platform independent
- Flexibility, Flexibility, Flexibility!
  - Any platform, vendor, OS, software, language
- XML is a very flexible data model: for structured data, semi-structured data, schema-less data
- Easy to extend: define new tags as needed
- XML is self-describing: any XML parser can "understand" it!
- Easy to "validate" XML, i.e. to check compliance with a schema - any XML parser can do it!
- Easy to transform XML documents into other formats (HTML, etc.)
- Fully Unicode compliant
- XML has become the “data interchange” format between B2B/B2C, inter- and intra-enterprise environments
What is XML?

- XML = eXtensible Markup Language
- XML is self-describing
- XML tags describe each element and their attribute
- "Lingua franca" for information exchange between organizations, applications, services, processes, ...

XML: Describes data
HTML: Describes display

```xml
<DEPT DEPTNO="D01" NAME="DEVELOPMENT CENTER">
  <PROJ PROJNO="AD3100" NAME="ADMIN SERVICES">
    <EMP EMPNO="000010" EMPNAME="BRIAN BARTAK">
      <EMP EMPNO="000010" EMPNAME="BRIAN BARTAK">
        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
          <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
            <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                  <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                    <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                      <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                          <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                            <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                  <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                    <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                      <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                          <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                            <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                  <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                    <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                      <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                          <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                            <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                  <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                    <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                      <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                          <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                            <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                  <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                    <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                            <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                             <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                           <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                               <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                   <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                      <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                          <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                  <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                      <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                          <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                  <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                     <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                         <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                             <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                 <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                     <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                           <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                              <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                 <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                     <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                         <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                             <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                                   <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                                       <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                                           <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                                               <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                                                   <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                                                       <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                                                                        <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
                                                                                                                                                </EMP>
                                                                          </EMP>
                                                                        </EMP>
                                                                      </EMP>
                                                                    </EMP>
                                                                  </EMP>
                                                                </EMP>
                                                              </EMP>
                                                            </EMP>
                                                          </EMP>
                                                        </EMP>
                                                      </EMP>
                                                    </EMP>
                                                  </EMP>
                                                </EMP>
                                              </EMP>
                                            </EMP>
                                          </EMP>
                                        </EMP>
                                      </EMP>
                                    </EMP>
                                  </EMP>
                                </EMP>
                              </EMP>
                            </EMP>
                          </EMP>
                        </EMP>
                      </EMP>
                    </EMP>
                  </EMP>
                </EMP>
              </EMP>
            </EMP>
          </EMP>
        </EMP>
      </EMP>
    </EMP>
  </EMP>
</PROJ>
</DEPT>
```
XML or Relational?

**XML**
- Model is hierarchical in nature.
- Data is self describing. A single document can contain multiple types of data.
- The data contains an inherent ordering.
- More tolerant of structure / schema changes.
- Parsing and publication (serialization) costs involved in processing the data.
- Also consider the type of data / processing expected by downstream processes.
- Does the data need to be updated often, and with what granularity?
- Is referential integrity needed?

**Relational**
- Model is flat. Structure can be achieved through parent / child relationships.
- Data is not self describing. Data content is described by the column definition.
- Data is not ordered, unless explicitly requested.
- Fairly rigid implementation of the data.
- Provides optimal performance.

Also consider the type of data / processing expected by downstream processes.
Why use XML with Databases?

- Managing large volumes of XML data is a problem!
  - Efficient Search & Retrieval of XML Documents
  - Persistency, Recovery, Transactions
    - File system recovery vs. database recovery
  - Performance, Scalability
    - Indexing
  - …all the same reasons as for relational data!

- Integration
  - Integrate XML data with existing relational data
  - Publish (relational) data as XML
  - Database support for web applications, SOA, web services (SOAP)
Native XML and pureXML™ in DB2

- **Native XML**
  - Hierarchical data model: XDM (XQuery Data Model)
  - XML query languages: XPath

- **pureXML in DB2**
  - Designed specifically for XML from the ground up
    - Supports XML hierarchical structure storage
    - Native operations and languages: XPath, SQL/XML
  - No transformation into relational
  - Not using objects or nested tables
  - Not using LOBs
DB2 Objects For XML Support

- **BASE Table**
  - Cols:
    - DOCID
    - XMLCol1
    - XMLCol2

- **Segmented base table space**

- **Table for XMLCol1**
  - Cols:
    - DOCID
    - NODEID
    - XMLDATA
  - INDEX

- **PBG TS for XMLCol1**
  - INDEX

- **Table for XMLCol2**
  - Cols:
    - DOCID
    - NODEID
    - XMLDATA
  - INDEX

- **PBG TS for XMLCol2**
  - INDEX
XML Objects for Partitioned Base Table

Partitioned Base TS
2 Parts, Table has 2 XML Columns

Part TS XMLCOL1

DOCID
INDEX (NPI)

Cols:
DOCID  
XML COL1  
XML COL2

BASE Table Part1

Cols:
DOCID  
XML COL1  
XML COL2

BASE Table Part2

Part1

DOCID
NODEID
XMLDATA

Range-partitioned table space with partitions for XMLCOL1

Part2

DOCID
NODEID
XMLDATA

XML INDEX

NODEID INDEX

Part TS XMLCOL2

DOCID
NODEID
XMLDATA

Range-partitioned table space with partitions for XMLCOL2

Part1

DOCID
NODEID
XMLDATA

XML INDEX

NODEID INDEX
Physical Implementation of Table with an XML Column

```
create table XML_TEST_IN_DEFAULT_DB
(eye_catcher_1 char(03),
SEQUENCE XML_KEY integer,
eye_catcher_2 varchar(07),
XML_DOCUMENT XML)
```

DATABASE: DSN00004 (DSNxxxxxx)

TABLESPACE: XMLRTEST (table name or xxxxynnn), Segmented

<table>
<thead>
<tr>
<th>Fixed Length Columns</th>
<th>Varchar Ptr</th>
<th>Varchar Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocID</td>
<td>VC1</td>
<td>VC2</td>
</tr>
<tr>
<td>EC2</td>
<td>XML Indicator</td>
<td></td>
</tr>
</tbody>
</table>

XML_TEST_DEFAULT_DB

TABLESPACE (Unicode): XXML0000 (table name or Xyyynnnn), Segmented

16K Page

<table>
<thead>
<tr>
<th>Clustered on</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DocID</td>
<td>MinNodeID</td>
</tr>
<tr>
<td>XMLData</td>
<td></td>
</tr>
</tbody>
</table>

BIGINT       VARBIN(128)    VARBIN(15850)
XML Object Data Sets & DISPLAY

```
DSLST - Data Sets Matching ISC910P8.DSNDBD.DSN00004

Command - Enter "/" to select action

ISC910P8.DSNDBD.DSN00004.IRDOCIDX.I0001.A001
ISC910P8.DSNDBD.DSN00004.IRNODEID.I0001.A001
ISC910P8.DSNDBD.DSN00004.XMLRTEST.I0001.A001
ISC910P8.DSNDBD.DSN00004.XXXML0000.I0001.A001

********************************************************************** End of Data Set list *************
```

```
NAME      TYPE PART STATUS
--------- ---- ----- -----
XMLRTEST  TS   0001  Rw
XXXML0000 XS   0001  Rw
IRDOCIDX  IX   L*   Rw
IRNODEID  IX   L*   Rw

********** DISPLAY OF DATABASE DSN00004 ENDED  **********
DSN9022I @ISC9 DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```
Native XML Features

- First-class XML data type, native storage of XQuery Data Model
- SQL/XML constructor functions
  - Construct XML from relational data in V8: XMLElement, XMLAttributes, XMLNamespaces, XMLForest, XMLConcat, XMLAGG
  - New constructor functions in DB2 9: XMLText, XMLPI, XMLComment, XMLDocument, XMLTABLE, XMLCAST and binary string support and more null handling options
- XMLPARSE and XMLSERIALIZE
- XML indexes
XML Publication Example

SELECT
XMLSERIALIZE
(XMLELEMENT
(NAME "DEPT",
XMLATTRIBUTES
(D.DEPTNO AS "DEPTNO",
D.DEPTNAME AS "NAME")
,
SELECT XMLAGG
(XMLELEMENT
(NAME "PROJ",
XMLATTRIBUTES
(P.PROJNO AS "PROJNO",
P.PROJNAME AS "NAME")
,
SELECT XMLAGG
(XMLELEMENT
(NAME "EMP",
XMLATTRIBUTES
(E.EMPNO AS "EMPNO",
E.FIRSTNME || ' ' || E.LASTNAME AS "EMPNAME")

)
FROM DBA015.EMPPROJECT EP,
DBA015.EMP E
WHERE EP.PROJNO = P.PROJNO AND
EP.EMPNO = E.EMPNO
)
)
FROM DBA015.PROJ P
WHERE P.DEPTNO = D.DEPTNO
)
AS CLOB
FROM DBA015.DEPT D
WHERE D.DEPTNO = 'D01'

Query Result: (formatted for easy viewing)

<DEPT DEPTNO="D01" NAME="DEVELOPMENT CENTER">
  <PROJ PROJNO="AD3100" NAME="ADMIN SERVICES">
    <EMP EMPNO="000010" EMPNAME="BRIAN BARTAK">
    </EMP>
  </PROJ>
  <PROJ PROJNO="MA2100" NAME="WELD LINE AUTOMATION">
    <EMP EMPNO="000010" EMPNAME="BRIAN BARTAK">
    </EMP>
    <EMP EMPNO="000110" EMPNAME="VINCENZO LUCCHESI">
    </EMP>
  </PROJ>
</DEPT>
Native XML Features

- **Important SQL/XML functions with XPath**
  - XMLQUERY – used in the SELECT clause
  - XML EXISTS – used with the WHERE predicates

```sql
select pub_date,
xmlserialize(
  xmlquery('//AIRLINE/FLIGHTNO' passing pub_schedule)
as clob(1k)) as "Flight No Data"
from publish_schedule
where xmlexists('$/s/AIRLINE[NAME="AA"]',
passing pub_schedule as "s")
```

- Returns:

```xml
2007-02-20
<FLIGHTNO DEPTIME="10:31:00" ARRTIME="13:18:00" DEPARPT="FLL" ARRARPT="13:18:00">
<FLIGHTDTLS MEALDESC="Mystery Meat" EQUIPMENT="Boeing 757"/>
<FLIGHTNO DEPTIME="10:49:00" ARRTIME="13:13:00" DEPARPT="TPA" ARRARPT="13:13:00">
<FLIGHTDTLS MEALDESC="Mystery Meat" EQUIPMENT="McDonnell Douglas MD80"/>
<FLIGHTNO DEPTIME="12:30:00" ARRTIME="13:30:00" DEPARPT="ORD" ARRARPT="13:30:00">
<FLIGHTDTLS MEALDESC="Nothing for you" EQUIPMENT="Boeing 757"/>
<FLIGHTNO DEPTIME="10:31:00" ARRTIME="13:18:00" DEPARPT="FLL" ARRARPT="13:18:00">
```
Native XML Features…

- **COBOL DCLGEN:**

```
* DCLGEN TABLE(PBARTAK.PUBLISH_SCHEDULE) *
* LIBRARY(PBARTAK.V9.COBOL(DCLGNXML)) *
* LANGUAGE(COBOL) *
* QUOTE *
* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
*******************************************************************
EXEC SQL DECLARE PBARTAK.PUBLISH_SCHEDULE TABLE
( PUB_DATE DATE NOT NULL,
  PUB_SCHEDULE XML ) END-EXEC.
*******************************************************************
* COBOL DECLARATION FOR TABLE PBARTAK.PUBLISH_SCHEDULE *
*******************************************************************
01 DCLPUBLISH-SCHEDULE.
  10 PUB-DATE PIC X(10).
  10 PUB-SCHEDULE USAGE SQL TYPE IS XML AS CLOB(1M).
*******************************************************************
```

- XML Schema repository, Validation UDF, (and decomposition)
- DRDA (distributed support) and application interfaces
- Utilities
Multi-Versioning (MV) Support...

- Requires the base table to be a Universal Tablespace (UTS)
- MV format required for several V10 features:
  - XMLMODIFY
  - Temporal data (“AS OF”)
  - Currently Committed
  - SELECT FROM OLD TABLE (V9 feature)
- The MV format reduces locking
  - XML data is not kept with base row data in the work files
  - MV eliminates the need for a lock to be maintained on the XML row until it is processed, except for UR Readers (DOCID lock). Avoids a UR Reader from getting an incomplete document.
Multi-Versioning (MV) Support...

- **DB2 9 pureXML Table Formats**

  **Base Table**
  - Relational Columns
  - **DOCID**
  - XML Indicator

  **XML Table(s)**
  - **DOCID**
  - **MIN_NODEID**
  - **XMLDATA**

**DOCID:**
- Colname: DB2_GENERATED_DOCID_FOR_XML
- Data Type: BIGINT
  - One per base table that has 1 or more XML column(s)

**XML:**
- Colname: DDL XML column name
- Data Type: VARBIN(6)
  - One per XML column associated with the base table

**MIN_NODEID:**
- Colname: MIN_NODEID
- Data Type: VARBIN(128)

**XMLDATA:**
- Colname: XMLDATA
- Data Type: VARBIN(15850)
Multi-Versioning (MV) Support...

- DB2 10 pureXML Table Formats

**Base Table**

<table>
<thead>
<tr>
<th>Relational Columns</th>
<th>DOCID</th>
<th>MIN_NODEID</th>
<th>XMLDATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**XML Table(s)**

<table>
<thead>
<tr>
<th>DOCID</th>
<th>MIN_NODEID</th>
<th>XMLDATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DOCID:**
- Colname: DB2_GENERATED_DOCID_FOR_XML
- Data Type: BIGINT
- One per base table that has 1 or more XML column(s)

**XML:**
- Colname: DDL XML column name
- Data Type: VARBIN(14). Now includes 8 byte TS.
- One per XML column associated with the base table

**DOCID:**
- Colname: DOCID
- Data Type: BIGINT

**MIN_NODEID:**
- Colname: MIN_NODEID
- Data Type: VARBIN(128)

**XMLDATA:**
- Colname: XMLDATA
- Data Type: VARBIN(15850)

**START_TS:**
- Colname: START_TS
- Data Type: BINARY(8)

**END_TS:**
- Colname: END_TS
- Data Type: BINARY(8)

RRF makes physical column order:
- DOCID, START_TS, END_TS, MIN_NODEID, XMLDATA
Subdocument Update with XMLMODIFY...

- **DB2 9 pureXML supported the updating of an XML column**
  - But the entire column needed to be replaced
- **DB2 10 delivers the XMLMODIFY function to:**
  - Insert nodes within an existing document
    
    ```sql
    update time_accounting_part
    set xml_time_account =
    xmlmodify ('insert node $m/PROJ after /DEPT/PROJ[@PROJNO="z01"]',
                xmlparse(document
                '<PROJ PROJNO="z02" NAME="pureXML V10 Updates"/>
                as "m")
    )
    WHERE ....
    ```
  - Replace existing nodes of a document
    
    ```sql
    update time_accounting_part
    set xml_time_account =
    xmlmodify ('replace_value of node /DEPT/@DEPTNO with "R32"'),
    WHERE ...
    ```
  - Delete nodes from an existing document
    
    ```sql
    set xml_time_account =
    xmlmodify ('delete node /DEPT/PROJ[@PROJNO="z01"]'),
    WHERE ...
    ```
- Requires MV format

    ```sql
    DSNT408I SQLCODE - -4730, ERROR: INVALID SPECIFICATION OF XML COLUMN
    XML_MV_TEST_FROMMV10.XMLCOL IS NOT DEFINED IN THE XML VERSIONING
    FORMAT, REASON 1
    ```
CREATE INDEX ON PurchaseOrders(XMLPO) Generate Keys Using XMLPATTERN ‘/purchaseOrder/items/item/desc’ as SQL VARCHAR(100);

- Mandatory keyword - GENERATE KEYS USING XMLPATTERN
- XPath for XMLPATTERN specifies nodes to be indexed
- XML Pattern is path expression without predicates
- 50 steps maximum in XPath pattern
- The maximum length of the XML pattern text is 4000 bytes (UTF-8).
- No partitioned index support
- Data Type: VARCHAR(n) (n<=1000) in UTF-8 or DECFLOAT
- No ASC/DESC support
XMLTABLE Example

This XML document is in column XML_TIME_ACCOUNT for a row with a DEPTNO of ‘D21’

This SQL/XML with XMLTABLE surfaces the XML data in relational format
XMLTABLE Example

This can be coded into a View to provide transparent access to XML information.

```
XML Example:
- <DEPT DEPTNO="D21" NAME="ADMINISTRATION SYSTEMS">
-  <PROJ PROJNO="AD3110" NAME="GENERAL AD SYSTEMS">
-   <EMP EMPNO='000070' EMPNAME='EVA PUJASKI'>
    <STARTDATE>1982-01-01</STARTDATE>
    <ENDDATE>1983-02-01</ENDDATE>
    <ACTIVITY>10</ACTIVITY>
-   </EMP>
-  </PROJ>
- </DEPT>
```

Table Example:

<table>
<thead>
<tr>
<th>DEPTNO</th>
<th>DEPTNAME</th>
<th>PROJNO</th>
<th>PROJNAME</th>
<th>EMPNO</th>
<th>EMPNAME</th>
<th>STARTDATE</th>
<th>ENDDATE</th>
<th>ACTIVITY</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>GENERAL AD SYSTEMS</td>
<td>000070</td>
<td>EVA PUJASKI</td>
<td>1982-01-01</td>
<td>1983-02-01</td>
<td>10</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PAYROLL PROGRAMMING</td>
<td>000230</td>
<td>JAMES JEFFERSON</td>
<td>1982-01-01</td>
<td>1983-03-15</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PAYROLL PROGRAMMING</td>
<td>000230</td>
<td>JAMES JEFFERSON</td>
<td>1982-03-15</td>
<td>1982-04-15</td>
<td>60</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PAYROLL PROGRAMMING</td>
<td>000230</td>
<td>JAMES JEFFERSON</td>
<td>1982-05-15</td>
<td>1983-09-15</td>
<td>70</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PAYROLL PROGRAMMING</td>
<td>000230</td>
<td>JAMES JEFFERSON</td>
<td>1982-03-15</td>
<td>1982-10-15</td>
<td>70</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PAYROLL PROGRAMMING</td>
<td>000230</td>
<td>JAMES JEFFERSON</td>
<td>1982-04-15</td>
<td>1982-10-15</td>
<td>80</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PAYROLL PROGRAMMING</td>
<td>000230</td>
<td>JAMES JEFFERSON</td>
<td>1982-09-15</td>
<td>1983-01-01</td>
<td>80</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PAYROLL PROGRAMMING</td>
<td>000230</td>
<td>JAMES JEFFERSON</td>
<td>1982-10-15</td>
<td>1983-01-01</td>
<td>10</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-01-01</td>
<td>1982-02-01</td>
<td>60</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-02-01</td>
<td>1982-03-15</td>
<td>60</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-12-01</td>
<td>1983-01-01</td>
<td>60</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1983-01-01</td>
<td>1983-02-01</td>
<td>60</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-02-01</td>
<td>1982-03-15</td>
<td>70</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-03-15</td>
<td>1982-08-15</td>
<td>70</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-08-15</td>
<td>1983-10-15</td>
<td>70</td>
<td>0.25</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-08-15</td>
<td>1982-10-15</td>
<td>80</td>
<td>0.25</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1982-10-15</td>
<td>1982-12-01</td>
<td>80</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>PERSONNEL PROGRAMMG</td>
<td>000260</td>
<td>DANIEL SMITH</td>
<td>1983-01-01</td>
<td>1983-01-01</td>
<td>18</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000270</td>
<td>MARIA PEREZ</td>
<td>1982-03-01</td>
<td>1982-04-01</td>
<td>60</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000270</td>
<td>MARIA PEREZ</td>
<td>1982-04-01</td>
<td>1982-09-01</td>
<td>60</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000270</td>
<td>MARIA PEREZ</td>
<td>1982-09-01</td>
<td>1982-10-15</td>
<td>60</td>
<td>0.25</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000260</td>
<td>SYBIL JOHNSON</td>
<td>1982-06-15</td>
<td>1982-07-01</td>
<td>70</td>
<td>0.50</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000260</td>
<td>SYBIL JOHNSON</td>
<td>1982-07-01</td>
<td>1983-02-01</td>
<td>70</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000270</td>
<td>MARIA PEREZ</td>
<td>1982-09-01</td>
<td>1982-10-15</td>
<td>70</td>
<td>0.75</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000270</td>
<td>MARIA PEREZ</td>
<td>1982-10-15</td>
<td>1983-02-01</td>
<td>70</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000260</td>
<td>SYBIL JOHNSON</td>
<td>1982-01-01</td>
<td>1983-03-01</td>
<td>80</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000270</td>
<td>MARIA PEREZ</td>
<td>1982-01-01</td>
<td>1982-03-01</td>
<td>80</td>
<td>1.00</td>
</tr>
<tr>
<td>D21</td>
<td>ADMINISTRATION SYSTEMS</td>
<td>AD311</td>
<td>ACCOUNT PROGRAMMG</td>
<td>000260</td>
<td>SYBIL JOHNSON</td>
<td>1982-03-01</td>
<td>1982-04-15</td>
<td>80</td>
<td>0.50</td>
</tr>
</tbody>
</table>
XML Schema Support & Validation

- **XML Schema Support**
  - XML Schema Repository (Database DSNXSR)
  - Tables to store XML schemas
  - Created by DSNTIJNX in NFM
  - Multiple versions of a schema can be stored
  - Stored procedures to register XML schemas

- **SYSIBM.DSN_XMLVALIDATE built-in function**
  - Test XML values for validity against XML schema
  - Obtain default values and schema normalized values from XML schema
  - Validation of document size up to 2GB – 1.
  - 100% zIIP / zAAP redirectable, if available

- **XML Column Type Modifier**
  - Allows a “constraint” type of definition on an XML column via CREATE or ALTER TABLE
  - All documents INSERTed, UPDATEd, or LOADed in the column are validated to conform to the specified schema(s)
  - Caution using a Column Type Modifier & DSN_XMLVALIDATE function
Native SQL Procedure & UDF Enhancements

- XML is now valid as a passed variable
- XML can be a data type of a declared variable
- With these enhancements parsed XML can be processed within the routine without repetitively reparsing the serialized XML string

This snippet of code shows an example of a DECLARE statement for the XML data type for a case when XML is received as a CLOB (DB2 10 could pass the document in to the procedure as type XML)

```
-- Declare variables
DECLARE Return_SQLCODE_TMP VARCHAR(500) DEFAULT ' ';
DECLARE Return_Message_TMP VARCHAR(500) DEFAULT ' ';
DECLARE SQLCODE INTEGER;
DECLARE num_rows INTEGER DEFAULT 0;
DECLARE Parsed_XMLDOC XML;
DECLARE EXIT HANDLER FOR SQLSTATE '23502'
BEGIN
SET Return_Message_TMP = Return_Message_TMP ||
' NULL INSERT Exit Handler Terminated routine!!';
SET Return_SQLCODE = Return_SQLCODE_TMP;
SET Return_Message = Return_Message_TMP;
END;
SET Parsed_XMLDOC = XMLPARSE(DOCUMENT XMLDOC);
```

The body of the Native SQL Procedure then uses the XMLPARSE function to store a document in the variable.

Subsequent use of the Parsed_XMLDOC variable does not require parsing.
XML Date & Time Support

- **DATE and TIME are not supported types in DB2 9 XPath**

- **DB2 10 provides**
  - `xs:dateTime`
  - `xs:time`
  - `xs:date`
  - `xs:duration`
  - `xs:yearMonthDuration`
  - `xs:dayTimeDuration`
  - Several functions (fn) for comparison, value extraction, arithmetic operations

- **Implicit time zone will be UTC if not specified**

- **These types can be used in the creation of XML Value Indexes**

```sql
CREATE INDEX XMLPOTX3
ON TIME_ACCOUNTING_PART
(XML_TIME_ACCOUNT)
GENERATE KEYS USING XMLPATTERN
'/DEPT/PROJ/EMP/STARTDATE' AS SQL_DATE

SELECT DEPTNO,
   ROW_TIMESTAMP,
   XMLPOT_SEQUENCE,
   XMLSERIALIZE(XML_TIME_ACCOUNT AS CLOB) AS XML_TIME_ACCOUNT,
   XMLSERIALIZE(XML_AUDIT AS CLOB) AS XML_AUDIT
FROM TIME_ACCOUNTING_PART
WHERE XML_EXISTS('/DEPT[DEPTNO="D21"]' PASSING XML_TIME_ACCOUNT)
AND XML_EXISTS('/DEPT/PROJ/EMP[STARTDATE >= xs:date("1983-01-01")]' PASSING XML_TIME_ACCOUNT)
```

```xml
,xmlquery('fn:max(/DEPT/PROJ/EMP/xs:date(ENDDATE))'
  - fn:min(/DEPT/PROJ/EMP/xs:date(STARTDATE))'
  passing xml_time_account) as "xs:date Duration"
```
XML Restrictions and Limits

- **Following not supported if XML column exists**
  - ALTER TABLE ADD PART, ROTATE PART, ALTER PART
  - NO MQT – cannot have XML as a result column

- **XML Table Space attributes are inherited from the base table, only some can be altered, e.g.:**
  - Buffer Pool, STOGROUP, PCTFREE, GBPCACHE, Primary/Secondary Quantity, MAXROWS
  - Default Buffer Pool - ZParm

- **Stored documents may not maintain the exact copy as the original document**
  - Due to whitespace stripping, default values, XML standard to handle line feeds

- **Parallelism**
  - Parallelism will be disabled if SQL statement references any XML data type

- **XML type has no size limit, but**
  - DB2 has an effective limit of 2 gigabytes (actually, 2GB-1) for serialized XML data

- **Stored Procedure and trigger transition variables cannot be of an XML type**
  - Use CLOB and cast within the routine
  - Native SQL Procedure can have an XML data type variable
Rounding Out XML Support

**Utility Support**

- LOAD/UNLOAD, CHECK DATA/INDEX, COPY, REBUILD, RECOVER, REORG, etc.
- REORG to reclaim space, reorder rows (several may make up one document), and for compression

- **DB2 10 CHECK DATA** does:
  - Base table consistency with NODEID index (CHECK DATA)
  - NODEID index consistency with XML TS (CHECK INDEX)
  - Check document structure consistency for each document
  - Schema validation if column(s) have a type modifier
    - SHRLEVEL REFERENCE moves invalid documents to an automatically created exception table
      - With XMLERROR INVALIDATE or AUXERROR INVALIDATE
    - SHRLEVEL CHANGE will generate REPAIR statements
Rounding Out XML Support…

- DSNTEP2, DSNTEP4, and DSNTIAUL are modified to support the XML data type.

- IVP job DSNTEJ1 (creates the sample database) is modified to:
  - Create a new sample database for the XML data
  - Populate the database with serialized XML values using both SQL and the LOAD utility
Data Studio Support of pureXML

- Data Studio can:
  - Create a table with a XML column
  - Edit XML documents
  - Insert XML into the database
  - View XML data in the database
  - Extract / Load XML data
  - Create XML Schemas
Temporal Data – Time Travel Query

- What is temporal data?
- Basic Temporal Concepts
  - Business Time & System time
  - Row Maintenance
  - Period Versioning Information
- Example of a table with bi-temporal data
What is temporal data?

- One of the major improvements in DB2 10 will be the ability for the database to reduce the complexity and amount of coding needed to implement “versioned” data, data that has different values at different points in time.

- Data that you need to keep a record of for any given point in time

- Data that you may need to look at for the past, current or future situation

- The ability to support history or auditing queries

- Supporting Business Time and System Time
Basic Temporal Concepts

- **Business Time (Effective Dates, Valid Time, From/To-dates)**
  - Every row has a pair of TIMESTAMP(6) or DATE columns set by Application
    - Begin time: when the business deems the row valid
    - End Time: when the business deems row validity ends
  - Constraint created to ensure Begin time < End time
  - Query at current, any prior, or future point/period in business time

- **System Time (Assertion Dates, Knowledge Dates, Transaction Time, Audit Time, In/Out-dates)**
  - Every row has a pair of TIMESTAMP(12) columns set by DBMS
    - Begin time: when the row was inserted in the DBMS
    - End Time: when the row was modified/deleted
  - Every base row has a Transaction Start ID timestamp
  - Query at current or any prior point/period in system time

- **Times are inclusive for start time and exclusive for end times**
Basic Temporal Concepts

- **Bi-temporal**
  - Inclusion of both System Time and Business Time in row

- **Temporal Uniqueness**
  - PK or Unique Key with BUSINESS_TIME WITHOUT OVERLAPS
  - Support for a unique constraint for a point in time
  - This is optional, however without it:
    - Unique constraints will likely return errors due to multiple rows per key
    - If there’s no key and System Time is used
      - All history table rows are considered new additions
      - Not associated with previous rows with same column values

- **History Table**
  - Table to save “old” rows when using System Time
Table Defined with Business and System time

CREATE TABLE POLICY
    EMPL VARCHAR(4) NOT NULL,
    TYPE VARCHAR(4),
    PLCY VARCHAR(4) NOT NULL,
    COPAY VARCHAR(4),
    SYS_BEG TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
    SYS_END TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
    CRT_ID TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID NOT NULL,
    PERIOD SYSTEM_TIME (SYS_BEG, SYS_END),
    EFF_BEG DATE NOT NULL,
    EFF_END DATE NOT NULL,
    PERIOD BUSINESS_TIME (EFF_BEG, EFF_END),
    PRIMARY KEY (EMPL, PLCY, BUSINESS_TIME WITHOUT OVERLAPS);

Adding the PERIOD BUSINESS_TIME clause enables business time.
Adding BUSINESS_TIME WITHOUT OVERLAPS guarantees there can only be one row for a given business time.
It is possible to define the TRANSACTION START ID (required for System Time) as NULLABLE.
Any System Time columns may also define as Implicitly Hidden.
ALTER TABLE ADD PERIOD... can be used to add Business / System Time periods to existing tables.
History table for SYSTEM TIME

CREATE TABLE POLICYHISTORY
(EMPL VARCHAR(4) NOT NULL,
 TYPE VARCHAR(4),
 PLCY VARCHAR(4),
 COPAY VARCHAR(4),
 EFF_BEG DATE NOT NULL,
 EFF_END DATE NOT NULL,
 SYS_BEG TIMESTAMP(12) NOT NULL,
 SYS_END TIMESTAMP(12) NOT NULL,
 CRT_ID TIMESTAMP(12) NOT NULL);

OR

CREATE TABLE POLICYHISTORY LIKE POLICY;

To enable SYSTEM TIME you then alter the table:

ALTER TABLE POLICY
ADD VERSIONING USE HISTORY TABLE POLICYHISTORY;

Note:

If you need to make changes to the table, you will need to alter the table to drop the versioning, make the changes, and then alter the table to add versioning.
Row Maintenance with System Time

- **No temporal syntax for System Time maintenance**
  - Use regular Update, Delete, Insert statements

- **If the modification impacts existing base table rows**
  - Insert or Update
    - The base table row(s) are created / updated with a current System Start Time and high value System End Time.
  - Delete
    - Remove the base table row.
  - Update or Delete
    - Create a “before-image” copy of all qualified base table rows in the History Table.
    - If a previous copy of a row exists in the History table
      - Limit the System timeframe of the previous copy
        - Set previous copy’s End System Time = copied row’s Begin System Time
    - The newly created History row(s) are added with a System End Time equal to the current time (System Start Time of the associated base table row for an update)
**Row Maintenance with Business Time**

- **Temporal syntax is used for Business Time maintenance**
  - FOR PORTION OF BUSINESS_TIME FROM x TO y

- **If the modification impacts existing base table rows**
  - Insert
    - If a PK includes Business Time check for overlaps for the same PK of different base table rows
      - 803 returned if overlaps are found
    - Insert the base row with the specified Begin & End Business Times
  - Update / Delete
    - Check the specified row against existing qualified rows
    - Rows *contained within* the specified Business Time range are updated / deleted
      - row Business Time remains unchanged for the update
    - Rows that *span the specified From OR To* Business Time are
      - Updates: split into two rows, and updates applied to the portion of Business Time within the From and To
      - Deletes: The Begin or End Business Time is updated so no portion of the specified range remains
    - Row that *span the specified From AND To* are split into:
      - Updates: three rows, and updates applied to the portion of Business Time within the From and To
      - Deletes: two rows representing the remaining Business Time on either end of the specified range

---

In a bi-temporal implementation any changes to existing rows would also go through the System Time steps on the prior slide.
Business Period Versioning Information

- Business Time can be altered on an existing table

- FOR PORTION OF BUSINESS_TIME must specify valid Date/Time values in the FROM and TO

- Consider the implications of non-temporal UPDATE & DELETE statements
  - These statements are allowed

- SQLERRD(3) does not reflect rows added due to a temporal UPDATE / DELETE
  - Consistent with RI handling

- It is possible to have contiguous Business Time ranges with the same non-temporal data in the row
System Period Versioning Information...

- **Cannot TRUNCATE**
  - INSERT, UPDATE, DELETE, and MERGE are accepted

- **To find the Base / History Tables**

  ```sql
  SELECT VERSIONING_SCHEMA, VERSIONING_TABLE
  FROM SYSIBM.SYSTABLES
  WHERE NAME = 'table-name'
  AND CREATOR = 'creator-name'
  ```

- **System Time can be altered on an existing table**
  - See Information Center topic:

- **QUIESCE of the Base or History**
  - Will cause a quiesce against all tables in the versioning relationship, including auxiliary spaces
Bi-temporal example …

```
INSERT INTO POLICY
(EMPL, TYPE, PLCY, COPAY, EFF_BEG, EFF_END)
VALUES ('CO54', 'HMO', 'P667', '$10', '1/1/2004', '12/31/9999');
```

Policy Table

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14.745082721000</td>
<td>9999-12-31-24.00.00.000000000000</td>
</tr>
</tbody>
</table>

Business Time start | Business Time end | SYSTEM Time start | SYSTEM Time end

```
UPDATE POLICY FOR PORTION OF BUSINESS_TIME
FROM '01/01/2011' TO '12/31/9999'
SET COPAY = '$15'
WHERE PLCY = 'P667';
```

Policy Table

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22.50672497000</td>
<td>9999-12-31-24.00.00.000000000000</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22.50672497000</td>
<td>9999-12-31-24.00.00.000000000000</td>
</tr>
</tbody>
</table>

Policy History table

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
</tr>
</thead>
</table>
System Time / Point In Time...

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF '2010-09-22-00.00.00.00000000000';
```

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

As of 09-22-2010, the only row that qualifies is the row from the history table, because on 09-24-2010 we updated the rows, and both rows in the current table begin on 09-24-2010.

As of 09-24-2010-17.33 and after, rows from the current table would be returned

Only the POLICY appears in the SELECT statement. POLICYHISTORY is automatically accessed.

Results only come from the history table

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
</tbody>
</table>
### System Time / Range ...

```sql
SELECT * FROM POLICY FOR SYSTEM_TIME
FROM '2010-09-22-00.00.00.0000000000'
TO '2010-09-25-00.00.00.0000000000';
```

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBLOCKNO</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TBNNAME</th>
<th>TABNO</th>
<th>ACCESTYPE</th>
<th>PREFETCH</th>
<th>QBLOCK_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>DBA015</td>
<td>POLICYHISTORY</td>
<td>2</td>
<td>R</td>
<td>S</td>
<td>NCOSUB</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>UNIONA</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>DBA015</td>
<td>POLICY</td>
<td>1</td>
<td>R</td>
<td>S</td>
<td>NCOSUB</td>
</tr>
</tbody>
</table>

**Result of query**

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>
Bi-temporal example ...

On 09/24/2010, the customer cancelled the policy

```
DELETE FROM POLICY FOR PORTION OF BUSINESS_TIME FROM CURRENT DATE TO '12/31/9999'
WHERE EMPL='CO54' AND PLCY='P667';
```
## Business time example

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>2010-09-24-19.44.47</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>2010-09-24-19.44.47</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2010-09-24</td>
<td>2010-09-24-19.44.47</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

```
SELECT * FROM POLICY FOR BUSINESS_TIME AS OF '2010-09-23' ORDER BY EFF_BEG;
```

Business time only looks at base table, not the history table.
LOB Enhancements

- LOB FETCH CONTINUE
- File Reference Variables
- Inline LOBs
- LOB Utility
- LOB Locking Changes
- Additional LOB Flexibility
- Streaming LOB Data
LOB FETCH CONTINUE

- **Specific FETCH that contains LOB or XML columns**
  - Used with programs that materialize LOBs
  - Application uses a buffer that might not be large enough to hold the entire LOB or XML value.
  - If any of the fetched LOB or XML columns do not fit, DB2 returns information about truncated columns and the actual length.

- **Retrieve LOB or XML data in multiple pieces without use of locators**
  - Must specify WITH CONTINUE on initial FETCH
  - Subsequent fetches use FETCH CURRENT CONTINUE
  - Application must manage buffers & reassemble data
  - Not required to fetch entire object before moving to next
  - SQLCA indicates whether data is truncated
A file reference variable is a variable defined in a host language.

A file name that allows direct transfer of LOB data between DB2 and the file.

Reduces the memory required to process the LOB in your user address space.

Language support: Assembler, C, C++, COBOL, Java, PL/1 and REXX supports z/OS and HFS sequential files.

LOAD/UNLOAD support for file reference variables (Also provided in V7/V8 via PK22910) supports PDS, PDSE & HFS.

For more details see Redbook: “LOBs with DB2 for z/OS: Stronger and Faster” SG24-7270
Benefits of Inline LOBs

- Ability to store a LOB entirely or partially in the base table
- Reduces I/O and CPU consumption, auxiliary table storage, and potentially virtual storage.
- This enables new workload that employs LOB data
- Eases LOB handling and management
- Inline portion of LOB compressible even though LOBs are not compressed
- Inline portion of LOB can be used in index on expression
- Inline LOBs stored in the base table can achieve similar performance characteristics as VARCHAR columns
Inline LOBs...

- **LOB data can also be split**
  - Entirely or partially
    - Split LOB
      - DASD storage associated with small LOBs may be reduced
      - Avoids I/O’s associated with LOB table spaces
      - Base table can become larger.
        - SQL limited to non-LOB columns may be impacted
      - Can CREATE or ALTER a table to implement or change the inline length
        - Will become a Pending Alter or REORG Pending, depending on the type of change

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Fixed Cols</th>
<th>Varchar Indicators</th>
<th>Varying Length Cols</th>
<th>InLine Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUX ID</td>
<td>AUX</td>
<td>CLOB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
- AUX table
- Prefix
- Fixed Cols
- Varchar Indicators
- Varying Length Cols
- InLine Length
- CLOB
Inline LOB DDL…

- **Pre-requisites**
  - DB2 10 NFM
  - RRF
  - Default value of other than NULL or empty string is supported
  - Define length of inline portion of LOB
    - **DDL**
      - Default 1000 bytes
      - Valid range 0 – 32680 bytes
    - **New ZParm** **LOB_INLINE_LENGTH**
      - Default 1000 bytes
      - Range 0 – 32680 bytes
      - Value of 0 = no inline length. Stored entirely in AUX.
  - **Restrictions**
    - Cannot alter shorter if used in index on expression or shorter than default data value
Inline LOB DDL...

- Create table with LOB column & Inline portion

```sql
CREATE TABLE mytable (
   ID SMALLINT,
   mylobcolumn CLOB(1M) INLINE LENGTH 1000);
```

- Alter table INCREMENT length of Inline LOB portion

```sql
ALTER TABLE mytable
ALTER mylobcolumn
SET DATA TYPE CLOB(1M) INLINE LENGTH 2000;
```

- Alter to larger size is an immediate change. Base table space is put in AREOR status. If rows exist at time of change, length change is materialized upon REORG.
Inline LOB DDL…

- **Alter table** DECREMENT length of Inline LOB portion

  ```sql
  ALTER TABLE mytable
  ALTER mylobcolumn
  SET DATA TYPE CLOB(1M) INLINE LENGTH 200;
  ```

  - Hard REORG pending status (REORP)

- **Alter add a lob column with inline portion**

  ```sql
  ALTER TABLE mytable
  ADD COLUMN mylobcolumn2
  CLOB(1M) INLINE LENGTH 2000;
  ```

  - AREO* status for table space
LOB Utility

- LOAD / Cross load LOB column lengths > 32KB supported
- Logging for > 1GB LOBs
- REORG LOB reclaim space
  - SHRLEVEL(REFERENCE)
  - Allows LOG NO
  - SHRLEVEL(NONE) is still an option.
- Online CHECK LOB and DATA
- PK75216 for File Reference Variable Performance
LOB Locking Changes

- **Elimination of LOB locks**
  - Now using LRSN & page latching for consistency checks
  - Prior to DB2 9, LOB locks were held until commit
    - Even for UR

- **Space search for LOB allocation**
  - No LOB locks acquired for space search
  - Read granularity improved to page level in LOB table space

- **Improved availability & performance**
  - Particularly for UR readers

- **Requirements:**
  - NFM
  - “Locking protocol 3” GBP changes
    - Automatic in non-data sharing
    - Clean group-wide quiesce in data sharing once NFM enabled
      - No quiesce is needed with PK62027
Additional LOB Flexibility

- AUX (and XML) table spaces & dependent indexes can be created DEFINE NO
  - Defers create time to first INSERT operation
  - Improves application installation time
  - Simplifies/improves backup as empty datasets don’t have to be backed up

- LOAD/UNLOAD LOBs along with non-LOB columns
  - Spanned record support
Advanced Design Options

- New Data Types
- Greater Timestamp Precision
- Timestamp with Time Zone
- Optimistic Locking
- CLONE Tables
- Table Space
  - Past Table Spaces
  - Universal Table Spaces (UTS)
- Online Schema Evolution
- Reordered Row Format
- NOT LOGGED Tables
- Table APPEND Option
- Compress on Insert
- Implicit Casting
- Hash Access
- Extended Address Volumes
- INSTEAD OF Triggers
- RENAME COLUMN, INDEX
New Data Types

- **DECFLOAT**
  - DECFLOAT(16)
    - 10+384 to 10-383 Positive & Negative
  - DECFLOAT(32)
    - 10+6144 to 10-6143 Positive & Negative
    - Well suited to typical customer financial calculations

- **BIGINT**
  - -9223372036854775808 to 9223372036854775807
  - Compatible with all numeric types

- **BINARY**
  - 1 to 255 bytes

- **VARBINARY**
  - 1 to 32704 bytes; maximum length determined by the maximum record size associated with the table

- **XML**
Greater Timestamp Precision

- **Number of digits for the fractional second in a timestamp extended**
  - The DB2 9 default of 6 digits remains
    - TIMESTAMP is the same as TIMESTAMP(6)
    - String representation: `yyyy-mm-dd-hh.mm.ss.nnnnnn``
  - Range supported in DB2 10 NFM is 0 to 12 digits
    - E.g. `TIMESTAMP(12)` is the maximum
    - String representation: `yyyy-mm-dd-hh.mm.ss.nnnnnnnnnnn`
  - Other capabilities like timestamp duration and CURRENT TIMESTAMP extended to support new precision capability
    - Can be altered to higher precision. Example: `TS(6)` to `TS(10)`
    - Will fail with SQLCODE -190 if lowered. Example: `TS(10)` to `TS(6)`
Timestamp with Time Zone

- **New data type** TIMESTAMP WITH TIME ZONE
  - SET SESSION TIME ZONE or rely on IMPLICIT_TIMEZONE (DECP)
  - Year.month.day.hour.minutes.seconds+-timezone
    - Example: 2011-07-06-11.00.000000-06:00

- **Comparison takes place after converting internally to UTC**

- **DB2 Special Registers:**
  - CURRENT TIME ZONE special register: ( data type = DECIMAL(6,0) )
    - Contains the difference between UTC and local time at the current server.
  - SESSION TIME ZONE special register: ( data type = VARCHAR(128))
    - Identifies the time zone of the application process.
Optimistic Locking Support

- **Built-in timestamp for each row or page**
  - Automatically updated by DB2 as a GENERATED ALWAYS column
  - Allows simple timestamp predicate to validate that row has not changed since last access

- **Eliminates need for complex predicates on WebSphere CMP updates, improves performance**

```sql
SELECT ROW CHANGE TIMESTAMP FOR FLIGHTS AS RCTIME,
       ROW CHANGE TOKEN FOR FLIGHTS AS RCTOKEN
FROM FLIGHTS;
```
CLONE Tables

- Allows fast replacing of production data without renames and rebinds
  - A capability to support online load replace

- ALTER TABLE to create a Clone Table
  - All indexes are also cloned
  - Table and Index data are not copied
  - Base and Clone tables share the same table space and index names
  - Underlying data sets are differentiated by a data set instance number

- Use SQL or LOAD utility to populate clone tables
- Use EXCHANGE to switch logical names with underlying data
- Clone indicated in CLONE column of SYSIBM.SYSTABLESPACE
- DISPLAY TABLESPACE command also indicates which data sets are the base and clone
CLONE Tables…

Base

Table: FLIGHTS
   Airline
   Flight_No
   Depart_Date
   Depart_Time
   From_Airport
   To_Airport
   Arrive_Date

Production Application

Table: FLIGHTS
   Airline
   Flight_No
   Depart_Date
   Depart_Time
   From_Airport
   To_Airport
   Arrive_Date

Loading Application

Table: FLIGHTS_CLONE
   Airline
   Flight_No
   Depart_Date
   Depart_Time
   From_Airport
   To_Airport
   Arrive_Date

Clone

Table: FLIGHTS
   Airline
   Flight_No
   Depart_Date
   Depart_Time
   From_Airport
   To_Airport
   Arrive_Date

Actions

DDL

ALTER TABLE MPW.FLIGHTS
   ADD CLONE MPW.FLIGHTS_CLONE;

DISPLAY TABLESPACE

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>PART</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLIGHTS</td>
<td>TSB1</td>
<td>0001</td>
<td>RW</td>
</tr>
<tr>
<td>FLIGHTS</td>
<td>TSC2</td>
<td>0001</td>
<td>RW</td>
</tr>
</tbody>
</table>

© 2011 IBM Corporation
CLONE Tables…

Base

Table: FLIGHTS
Airline
Flight_No
Depart_Date
Depart_Time
From_Airport
To_Airport
Arrive_Date

Production Application

Table: FLIGHTS
Airline
Flight_No
Depart_Date
Depart_Time
From_Airport
To_Airport
Arrive_Date

Table: FLIGHTS_CLONE
Airline
Flight_No
Depart_Date
Depart_Time
From_Airport
To_Airport
Arrive_Date

Actions

DDL

EXCHANGE DATA
BETWEEN TABLE MPW.FLIGHTS AND MPW.FLIGHTS_CLONE;

DISPLAY TABLESPACE

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>PART</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLIGHTS</td>
<td>TSB2</td>
<td>0001</td>
<td>RW</td>
</tr>
<tr>
<td>FLIGHTS</td>
<td>TSC1</td>
<td>0001</td>
<td>RW</td>
</tr>
</tbody>
</table>

© 2011 IBM Corporation
CLONE Tables…

- **Other notes**
  - Created in the same table space as the base table.
  - Separate data sets are used, differentiated by the Instance node.
    - I0001
    - I0002
  - Must be in a DB2 Managed, Universal Table Space
  - Restricts ALTERs while Clones exist
  - No pending changes
  - No active versioning
  - No RI or AFTER triggers
  - No fast switch
  - Cannot be an MQT
Past Table Spaces Options

- Past table space options
  - Simple
    - Multi table, interleaved
  - Segmented
    - Multi table, no page sharing
    - Good with mass deletes
    - 64GB
  - Partitioned
    - One table per table space
    - 128TB
    - Doesn’t have the internal space map like that of a segmented table space.
Universal Table Spaces (UTS)

- Combination of segmented with partitioning options
  - Better space management
  - Support of mass deletes / TRUNCATE
  - Required for many new features and functions

- Two types:
  - Range-partitioned (PBR)
    - Similar to partitioned table space using partitioning key and key ranges
  - Partition-by-growth (PBG)
    - Single-table table space, where each partition contains a segmented page set (allows segmented to increase from 64GB to 16TB or 128TB)
    - Eliminates need to define partitioning key and assign key ranges
    - Partitions are added on demand when a given partition reaches DSSIZE up to maximum number of partitions defined by MAXPARTITIONS.
    - Retains benefits of utilities and SQL parallelism optimizations for partitioned tables
## Universal Table Spaces

**What kind of Table Space will be created? (* optional)**

<table>
<thead>
<tr>
<th>CREATE TABLESPACE...</th>
<th>SEGSIZE</th>
<th>NUMPARTS</th>
<th>MAXPARTITIONS</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmented</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>*SEGSIZE is optional. Default for explicitly created TS &amp; implicitly created TS for CM8. SEGSIZE defaults to 4.</td>
</tr>
<tr>
<td>UTS PBG</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Default for CM9 and NFM with implicitly created TS. Single table TS. *SEGSIZE will default to 32.</td>
</tr>
<tr>
<td>UTS PBR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Single table TS *SEGSIZE will default to 32.</td>
</tr>
<tr>
<td>Classic Partitioned TS</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Partitioning TS prior to V9 *SEGSIZE 0 will create classic partitioned and CM8 behavior is same as V8 NFM</td>
</tr>
</tbody>
</table>

*SEGSIZE is optional and default for explicitly created TS & implicitly created TS for CM8.*
Online Schema Evolution – Improved ALTER

- **Availability and Productivity**
  - Enhancement to ALTERs performed by DBAs
    - Prior DB2 versions introduced the concept of immediate ALTERs
    - DB2 10 expands the concept by introducing pending changes
  - No need to CREATE new table spaces, UNLOAD/LOAD, GRANT, DROP, REBIND
  - DB2 materializes pending changes at DBA’s discretion on a future date during the execution of an online REORG
  - “Bottom line”… Less application outages due to database changes
Online Schema – Details on Execute ALTER Statement

- **Statement is validated**
  - Semantic checking against effective catalog definition

- **Assuming all checks out ok:**
  - Statement is put on pending list
  - Table space is placed in Advisory-REORG pending: **AREOR** (non-restrictive)
    - Not to be confused with REORG-pending advisory (**AREO***) which says access path could be degraded
  - Statement completes with SQLCODE +610 to advertise the advisory state

- **Drop changes**
  - ALTER TABLESPACE… DROP PENDING CHANGES
    - still in AREOR
    - all changes for that table space will be dropped

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>DBID</th>
<th>PSID</th>
<th>OBJSCHEMA</th>
<th>OBJNAME</th>
<th>...</th>
<th>OPTION_ KEYWORD</th>
<th>OPTION_ VALUE</th>
<th>...</th>
<th>STATEMENT_ TEXT</th>
</tr>
</thead>
</table>
Online Schema – Details on Online REORG

- **Pending DDL is materialized** - DSNU1163I
  - Catalog & Directory are updated with the new attributes
  - Data sets are updated with the new attributes
  - Materialized SYSPENDINGDDL entries are removed

- **Stats are collected**
  - Default is TABLE ALL INDEX ALL UPDATE ALL HISTORY ALL unless overridden
  - Warning message is issued to indicate that some partition statistics may no longer be accurate - DSNU1166I
    - (COLGROUP, KEYCARD, HISTOGRAM …)

- **SYSCOPY entries show inability to recover object prior to changes**

- **AREOR state is reset**
Pending ALTERs other than changing table space type are supported only for UTS

ALTERing table space type only for single-table table spaces

Not permitted to mix immediate and pending options in an ALTER statement (SQLCODE -20385)

Many immediate DDL statements are not allowed while there is pending DDL awaiting materialization (-20385)

- CREATE/DROP/ALTER
  - E.g. alter of FREEPAGE for a partition

Pending DDL only materialized by REORG SHRLEVEL REFERENCE or CHANGE
  - REORG SHRLEVEL(NONE) and part-level REORGs are not blocked, but do not materialize pending DDL

If MAXPARTITIONS is exceeded during REORG DSNU1170I will be issued
Online Schema...

- Restrict RECOVER across materializing REORGs

- Plans and packages are invalidated if changing table space type
  - When changing the MAXPARTITIONS attribute of a simple or segmented table space to convert it to a partition-by-growth universal table space
  - The SEGSIZE attribute of a partitioned table space is changed to convert the table space to a range-partitioned universal table space
Reordered Row Format (RRF)

- Automatic repositioning of variable length columns to end of row
  - Length attributes replaced with indicators positioned after fixed length columns
- Any table space created in DB2 9 NFM
- To Convert:
  - REORG or LOAD REPLACE a table space or partition
  - ADD PARTITION
  - No EDITPROCs or VALIDPROCs
    - EDITPROCs may need to be updated if implemented for specific columns
    - Byte RFMTTYPE passed to indicate fixed length, basic, or reordered format
    - Consider this impact on tables encrypted via an EDITPROC
  - DSN1COPY impact during the transition period across environments
- PIT RECOVER will set the table space to the row format of the PIT
- Catalog / Directory remains in Basic Row Format (BRF)

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Fixed Length Cols</th>
<th>Varchar Indicators</th>
<th>Varying Length Cols</th>
</tr>
</thead>
</table>
NOT LOGGED Tables

- Is actually NOT LOGGED tables, indexes, LOB, XML
- ALTER / CREATE a TABLESPACE as NOT LOGGED
  - ALTER not allowed if in same UOW with an update to the table space
- Indexes, LOB, and XML inherit the logging attribute of the base
  - This are considered “Linked” objects
- Effects the UNDO / REDO records
  - Control information is still logged
- LOB continue to log system pages & auxiliary indexes
- A Unit of Recovery (UR) is still created

LOG YES is a synonym for LOGGED
LOG NO is a synonym for NOT LOGGED
NOT LOGGED Tables

- Cannot explicitly specify for XML & Index objects
- LOBs can be set independent of the base table
  - However, if a LOB is LOGGED, the base must also be logged
  - This “dissolves the link” with the base
- Not compatible with CHANGE DATA CAPTURE attribute
  - Applies to any tables in the table space
- Not allowed for DSNDB06 or Work File database
- SYSCOPY activity
  - ALTER LOGGED to NOT LOGGED creates a recoverable point
  - ALTER NOT LOGGED TO LOGGED marks the object COPYP for the base table space
  - Frequent ALTERing may require more SYSCOPY space
NOT LOGGED Tables

- A FULL COPY should be taken
  - Just before ALTERing to NOT LOGGED
  - Just after ALTERing to LOGGED

- If changes are made while NOT LOGGED
  - The space is marked ICOPY
  - An ALTER to LOGGED will set COPYP

- Image copies can be SHRLEVEL NONE or REFERENCE
  - Full or incremental

- Be careful with a CANCEL command that impacts a thread acting on NOT LOGGED objects

- CANCEL, ROLLBACK, LOAD RESUME failures, and Restart
  - Can cause the object (and XML space) to end up in a RECP state and in the LPL
  - Indexes often end up in RBDP & in the LPL
Table APPEND Option

- **Balancing qualities of:**
  - Storing the data quickly at the cost of space & certain queries
  - More effective use of space at the cost of searching for space

- **CREATE / ALTER Table APPEND option:**
  - Maximizes performance for “INSERT at end”
  - Avoids overhead of attempting to preserve clustering sequence
  - REORG after bulk insert to establish clustering

- **Useful for:**
  - Cases where cluster is not important
  - Large batch INSERTs following by a REORG
Compress on INSERT

- INSERT, MERGE and LOAD trigger the creation of a compression directory if:
  - The table space or partition is defined with COMPRESS YES
  - The table space or partition has no compression dictionary built
  - Inserted data reaches a threshold that allows the build of the compression dictionary

- If threshold is reached, dictionary build is done asynchronously
  - Data continues to be inserted uncompressed until dictionary is ready
  - Rows read with UR in order to build the dictionary
  - DSNU235I – dictionary not built

- Default behaviour Compress on insert
  - Use COMPRESS NO if unwanted
Location of compression dictionary pages

- If the dictionary pages are scattered throughout table a REORG or COPY SYSTEMPAGES(YES) is needed to UNLOAD rows.

Dictionary pages not necessarily after space map page!
- Could use inline COPY during REORG
  To ensure dictionary pages up front.

DSNU1232I - COMPRESSED ROW IS IGNORED BECAUSE THE DICTIONARY IS NOT AVAILABLE FOR TABLE table-name

COPY TS... SYSTEMPAGES NO

COPY TS... SYSTEMPAGES YES
  (option makes DB2 copy the dictionary pages after space map page)
Implicit Casting of string and numeric data

- V8 could compare CHAR to VARCHAR; SMALLINT to INT
- CHAR FOR BIT DATA -> Binary explicitly in DB2 9
- What about archive table with different data types
  - *Select from MY.emp into ARCH.emp gets SQLCODE -401*
- DB2 10 implicitly casts character or graphic string to DECIMALFLOAT, and numeric back to VARCHAR
- Allows for index matching
Hash Access Candidates

- **Candidate Tables**
  - For queries needing single row access via the unique key
  - Queries having equal predicates on keys
  - With known and static approximate size of data
  - Having large N-level indexes

- **Not for Tables**
  - Needing sequential processing
  - Frequently updated
  - Either using BETWEEN or > and <

- **Follow-up**
  - Run REBIND with EXPLAIN option and query the PLAN_TABLE to check access path
  - SYSTABLESPACESTATS.REORGHASHACCESS
    - Number of times data is read using hash access in the last time interval
  - Check LASTUSED & REORGINDEXACCESS on overflow and other indexes to validate HASH access
Fastest Available Pre-DB2 10: Index access

Query:

```
SELECT * ...
WHERE ITEMNO = 'W0133-1662996'
```
Fastest Available In DB2 10: Hash Access

Query:

```
SELECT * ...WHERE
ITEMNO = 'W0133-1662996'
```

1 data page access

- Key finds the row without index
- Reduced:
  - Page visits
  - CPU time
  - Elapsed time

1 data page disk I/O
(Possibly in buffer pool)

- Trade-off: extra space used
Create Objects with Hash Organization

```
CREATE TABLE...
    ... ORGANIZE BY HASH UNIQUE (LASTNAME, FIRSTNAME)
    HASH SPACE 4G
```

```
CREATE TABLE...
    ... PARTITION BY RANGE...
        PARTITION 1  ...
        ....
        PARTITION 5  ...
    ...
    ... ORGANIZE BY HASH UNIQUE (LASTNAME, FIRSTNAME)
    HASH SPACE 2G
```

**TABLESPACE options:**
- Table space MUST be UTS PBG or PBR
- Can be created explicitly
- Or DB2 creates it implicitly – 64MB for HASH space

1. `SYSTABLESPACESTATS.TOTALROWS`: Actual number of rows in the table
2. `SYSTABLESPACESTATS.DATASIZE`: Total number of bytes used for rows
3. `SYINDEXXSPACESTATS.TOTALENTRIES`: Number of overflow records with keys in the overflow index

**Overflow INDEX**
- DB2 creates it implicitly

New variation of index object created to store hash column information. The index will only contain entries to rows in the overflow area.
ALTER TABLE ADD ORGANIZE BY HASH

- ALTER is IMMEDIATE to enforce uniqueness
  - Table in advisory reorg state
  - Overflow index in rebuild pending state
    - Inserts not allowed
    - Can delete/update (if not on key)
  - Rebuild Index (optional to allow inserts)
    - Builds a large index (contains entry for all rows)
    - Allows inserts after rebuild
    - Index will be very large
- Reorg table space
  - AUTOESTSPACE(YES) uses RTS
  - Part level REORG is not allowed
  - Changes the organization to hash
  - Index shrinks

Non-Hash Table Space

Hash Overflow Index

Part 1 Part 2 Part 3 Part 4 Part 5

ALTER ADD HASH

Index created, put in Rebuild pending

REORG

Fixed Hash Area

Part 1 Part 2 Part 3 Part 4 Part 5 Part 6 Part 7
**ALTER TABLE DROP ORGANIZATION**

- **Removes Hash organization**
  - Table is placed in REORP
  - Table and related indexes become inaccessible until after REORG
  - Implicitly created hash overflow index dropped during ALTER
  - Table space MUST be Reorged immediately
  - Consider which index will control clustering after this
  - Part level REORG is not allowed after DROP ORGANIZATION

![Diagram showing ALTER TABLE DROP ORGANIZATION and REORG processes](image)
Security & Administration for Today’s Business

- Security Challenges
- Trusted Context & Roles
- Auditors concerns about security
- Separation of Duties
- Revoke Not Including Dependent Privileges
- New Bind Options
- Row/Column Access control
- New Audit Capabilities
Security Challenges

- **Three-tier architecture issues**
  - All interactions with database server occur under middle-tier authorization ID
    - Loss of end-user identity
    - Over granting of privileges to the middle-tier authorization ID
    - Diminished user accountability
    - Weakened security
  - Performance problems with using end user credentials

- **Dynamic SQL Support**
  - Users need dynamic access to table within the context of an application
  - You don’t want end users to have this same access outside of the application
Trusted Context & Roles

- Establishes a trusted relationship between DB2 and an external entity
  - A Server or a User ID
  - Once established, a provides for specialized privileges only available via the Trusted Context via a Role
  - Remote: IP Address, Domain Name or RACF SERVAUTH security attributes
    - Can use three levels of network encryption of the data stream.
  - Local: Job or Task name attributes

- Role
  - Database entity that groups privileges
  - Can be assigned to a User ID
  - Can be owner of DB2 objects
  - Trusted Context has a Default Role

- See Admin Guide, Ch. 3, “Implementing your database design”
Auditors concerns about security …

- Worried as much about internal as external threats
- Reduce the number of people with access to sensitive data
- Overloading applications with security logic
  - Security logic can be bypassed by malicious users
  - Hampers the ability to use ad-hoc query tools
  - Difficult to maintain
- Different views for different groups of users
- Evolution of security policies
  - Affect the security logic in applications
  - Affect the organization and number of views
- Granularity of Database privileges
  - Privileges are granted at the database object level
  - Difficult to protect personal and sensitive data
  - Can not easily comply with data protection laws
Separation of Duties …

– New SECADM authority to manage security of data
  • Does not have access to the data
  • Is able to manage GRANTS on all objects
– New Administrative authorities
  • DATAACCESS – to control who can see data
  • ACCESSCTRL – to control who can govern access to the data
– New DBADM ON SYSTEM authority
  • Gives the individual or role the ability to
    – Manage all user tables in the subsystem
      > With or without DATAACCESS – default is with DATAACCESS
      > With or without ACCESSCTRL - default is with ACCESSCTRL
– Administrative authorities can be divided among individuals without overlapping responsibilities
– Allows security administrator to grant the minimum privilege to a user to perform a specific task
### Administrative Authority …

<table>
<thead>
<tr>
<th>Collections</th>
<th>User data</th>
<th>Plans, packages &amp; Routines</th>
<th>All schemas</th>
<th>JARS Sequences</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBADM ON SYSTEM</td>
<td>CREATEIN</td>
<td>BIND</td>
<td>CREATEIN ALTERIN DROPIN</td>
<td>ALTER</td>
<td></td>
</tr>
<tr>
<td>DATAACCESS</td>
<td>SELECT</td>
<td>EXECUTE</td>
<td>USAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCESSCTRL</td>
<td>GRANT</td>
<td></td>
<td>GRANT REVOKE REVOKE BY</td>
<td></td>
<td>GRANT REVOKE REVOKE BY</td>
</tr>
<tr>
<td>SECADM</td>
<td>GRANT</td>
<td></td>
<td>GRANT REVOKE REVOKE BY</td>
<td></td>
<td>GRANT REVOKE REVOKE BY</td>
</tr>
<tr>
<td>SQLADM</td>
<td>NONE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 – Except tables defined with row permissions or column masks

The information on this slide and the next slide can be found in the Administration Guide under the topic Administrative authorities.
## Administrative Authority

<table>
<thead>
<tr>
<th>Distinct types</th>
<th>User databases</th>
<th>System privileges</th>
<th>Catalog tables (Update when available)</th>
<th>Issue Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBADM ON SYSTEM</strong></td>
<td>USAGE privileges on BUFFERPOOLS TABLESPACE STOGROUP</td>
<td>CREATETABLE CREATETS DROP DISPLAYDB IMAGECOPY RECOVERDB STARTDB STATS STOPDB</td>
<td>BINDADD BINDAGENT CREATEALIAS CREATEDBA CREATEDBC CREATETMTAB DISPLAY EXPLAIN MONITOR1/MONITOR2 SQLADM STATISTICS</td>
<td>SELECT INSERT UPDATE DELETE</td>
</tr>
<tr>
<td><strong>DATAACCESS</strong></td>
<td>USAGE</td>
<td>RECOVERDB REORG REPAIR LOAD</td>
<td>DEBUGSESSION</td>
<td>SELECT INSERT UPDATE DELETE</td>
</tr>
<tr>
<td><strong>ACCESSCTRL</strong></td>
<td></td>
<td></td>
<td></td>
<td>SELECT INSERT UPDATE DELETE</td>
</tr>
<tr>
<td><strong>SECADM</strong></td>
<td>GRANT REVOKE REVOKE BY</td>
<td>GRANT REVOKE REVOKE BY</td>
<td>GRANT REVOKE REVOKE BY</td>
<td></td>
</tr>
<tr>
<td><strong>SQLADM</strong></td>
<td>STATS</td>
<td>MONITOR1 MONITOR2 EXPLAIN</td>
<td>SELECT INSERT UPDATE DELETE</td>
<td>START STOP DISPLAY PROFILE</td>
</tr>
</tbody>
</table>
INSTALL SECADM

- A person or role that manages DB2 objects
  - This separates the object management from data access and data control
- No inherent access to data
- Specified in SECADM1 and SECADM2
  - Install panel DSNTIPB
  - In conjunction with SECADM_TYPE
    - which can be AUTHID or ROLE
- Set INSTALL SECADM before setting SEPARATE_SECURITY to YES
- Activated by SEPARATE_SECURITY ZParm
  - If YES, then SYSADM and SYSCTRL can not perform GRANTS for others
SQLADM

- Designed to be used by Performance analyst
- This will allow performance analyst to do all performance work, except access data
- What can a person with SQLADM do?
  - Issue SQL EXPLAIN statement
  - Issue START, STOP and DISPLAY PROFILE commands
  - Perform actions involving
    - EXPLAIN privilege
    - STATS privilege on all user databases
    - MONITOR2 privilege
    - Execute DB2 supplied stored procedures and routines
  - Ability to SELECT, INSERT, UPDATE, DELETE on DB2 catalog tables
  - CAN NOT access data, perform DDL or EXECUTE plans or packages
EXPLAIN

- Designed for the application architect

- What can a user do with the EXPLAIN privilege?
  - Issue SQL EXPLAIN ALL statement without being able to EXECUTE that statement
  - Issue SQL PREPARE and DESCRIBE TABLE statements without having privileges on the object
  - BIND EXPLAIN(ONLY) and SQLERROR(CHECK)
  - Explain dynamic SQL statements executing under new special register
    - CURRENT EXPLAIN MODE = EXPLAIN
REVOKE DEPENDENT PRIVILEGES …

- Provides additional controls regarding cascading effects of a REVOKE statement
  - INCLUDING DEPENDENT PRIVILEGES
  - NOT INCLUDING DEPENDENT PRIVILEGES
    - When ACCESSCTRL, DATAACCESS, or DBADM ON SYSTEM is revoked,
      - the default is always NOT INCLUDING DEPENDENT PRIVILEGES and
      - the NOT INCLUDING DEPENDENT PRIVILEGES clause must be explicitly specified
Row and Column level access …

- **What is the purpose of row level security?**
  - Filter rows out of answer set
  - Policy can use session information like SQL ID is in what group or user is using what role to control when row is returned in result set
  - Applicable to SELECT, INSERT, UPDATE, DELETE & MERGE
  - Defined as a row permission:
    ```
    CREATE PERMISSION policy-name ON table-name
    FOR ROWS WHERE search-condition
    ENFORCED FOR ALL ACCESS ENABLE
    ```
  - Optimizer inserts search condition in all SQL statements accessing table. If row satisfies search-condition, row is returned in the answer set
Row and Column level access …

- **What is the purpose of column level security?**
  - Mask column values in answer set
  - Applicable to the output of outermost subselect
  - Defined as column masks:

```sql
CREATE MASK mask-name ON table-name FOR COLUMN column-name RETURN CASE expression ENABLE;
```

Optimizer inserts case statement in all SQL accessing table to determine mask value to return in answer set
Row and Column level access …

- Define a column or row policy based on who is accessing the table
  - `SESSION-USER`
    - Primary authorization ID of the process
  - `CURRENT SQLID`
    - SQL authorization ID of the process
    - `SET CURRENT SQLID = some authorization id`
  - `VERIFY_GROUP_FOR_USER` (new BIF)
    - Get authorization IDs for the value in `SESSION_USER`
      - Gets both primary and secondary auth ids
      - Return 1 if any of those auth IDs are in the argument
        
```
WHERE VERIFY_GROUP_FOR_USER(SESSION_USER,'MGR','PAYROLL') = 1
```
  - `VERIFY_ROLE_FOR_USER` (new BIF)
    - Get the role for the value in `SESSION_USER`
    - Return 1 if the role is in the argument list
        
```
WHERE VERIFY_ROLE_FOR_USER(SESSION_USER,'MGR','PAYROLL') = 1
```
Row and Column level access

- **Row and Column Access Control**
  - When activated row and column access controls:
    - Make row permissions and column masks become effective in all DML
      - All row permissions are connected with ‘OR’ to filter out rows
      - All column masks are applied to mask output
    - All access to the table if no user-defined row permissions
  - When deactivated row and column access controls:
    - Make row permissions and column masks become ineffective in DML
    - Opens all access to the table

```sql
ALTER TABLE table-name
  ACTIVATE ROW LEVEL ACCESS CONTROL
  ACTIVATE COLUMN LEVEL ACCESS CONTROL;
```

```sql
ALTER TABLE table-name
  DEACTIVATE ROW LEVEL ACCESS CONTROL
  DEACTIVATE COLUMN LEVEL ACCESS CONTROL;
```
DB2 Audit Capability …

- New audit capabilities without expensive data collectors
  - New Audit Policies are managed in the catalog
    - New SYSIBM.SYSAUDITPOLICIES table
    - Audit policy provides wild carding of table names
  - Ability to Audit privileged users
  - Ability to audit SQL activity against a table
    - Audit policy does not require AUDIT clause to be specified
    - Audit policy generates records for all read and update access, not just first access in the transaction
    - Audit policy includes additional records identifying the specific SQL statements reading or updating an audited UTS table
  - Ability to audit distributed identities
## Optimization Evolution

<table>
<thead>
<tr>
<th>Access Path Stability</th>
<th>Real Time Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other Access Path Stability Improvements</td>
<td></td>
</tr>
<tr>
<td>– Optimization Hints</td>
<td></td>
</tr>
<tr>
<td>EXPLAIN Changes</td>
<td></td>
</tr>
<tr>
<td>REBIND PLAN...COLLID</td>
<td></td>
</tr>
<tr>
<td>Literal Replacement</td>
<td></td>
</tr>
<tr>
<td>Statistics Enhancements</td>
<td></td>
</tr>
<tr>
<td>– Autostats</td>
<td></td>
</tr>
<tr>
<td>– RUNSTATS Profiles</td>
<td></td>
</tr>
<tr>
<td>– HISTOGRAM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual Indexes</td>
<td></td>
</tr>
<tr>
<td>Index Usage Improvements</td>
<td></td>
</tr>
<tr>
<td>Parallelism Enhancements</td>
<td></td>
</tr>
</tbody>
</table>
Access Path Stability

- REBIND PACKAGE... PLANMGMT(OFF | BASIC | EXTENDED)
  - Or REBIND TRIGGER PACKAGE

- Options:
  - OFF:
    - Do not use plan stability
    - Package continues to have one active copy
  - BASIC:
    - Package has one active copy (Current) and one old (Previous) copy
  - EXTENDED (default):
    - One active (Current) and two old / preserved package copies
    - The preserved copies are the Previous and Original copies
Access Path Stability

- **REBIND PACKAGE...SWITCH(PREVIOUS | ORIGINAL)**
  - SWITCH(PREVIOUS): toggles previous and current copies
  - SWITCH(ORIGINAL): previous deleted; current->previous; original cloned to current

- **FREE PACKAGE...PLANMGMTSCOPE(ALL | INACTIVE)**
  - ALL is the default and frees all copies
  - INACTIVE frees all old copies

- **SYSPACKAGE reflects the current copy**
  - Other package related tables reflect dependencies of all packages

- **DTYPE column of SYSPACKDEP indicates ‘P’revious or ‘O’riginal**

- **Restrictions**
  - No support for DBRMs bound into Plans.
Access Path Stability

- **New Catalog Table**
  - SYSPACKCOPY
    - Package copy information from V9 is copied into SYSPACKCOPY during ENFM
    - COPYID column
      - 1: Indicates a Previous copy
      - 2: Indicates an Original copy

- **APRETAINDUP for PLANMGM(T(BASIC or EXTENDED)**
  - APRETAINDUP keyword defaults to YES
  - NO: A new access path identical to a Current access path does not get copied to a Previous or Original
Access Path Stability

- **PLANMGMTSCOPE**
  - New System Parameter and Profile Attribute
  - For now, the only option (and default) is STATIC (V9 behavior)

- **PLANMGMT extended to Native SQL Procedures**

- **EXPLAIN PACKAGE command**
  - To retrieve PLAN_TABLE records for existing package
    - Original bind/rebind may have been EXPLAIN(NO)
    - Must have been bound on V10 to work.
    - Populate explain tables without BIND or REBIND.

```plaintext
>>-EXPLAIN----PACKAGE----------->

<<<<COLLECTION--collection-name--PACKAGE--package-name--------->

+---VERSION-version-name---+ | | +---COPY--copy-id---+
```
Other Access Path Stability Improvements (APSI)...

- **Static SQL provides many benefits, including:**
  - Performance
  - Security
  - Predictability / Stability

- **Optimization Hints**
  - Pre-DB2 10 offered User Level Optimization Hints
  - DB2 10 introduces Statement Level Optimization Hints
    - Access Path Repository
      - Statement level hints with **BIND QUERY** for static or dynamic SQL
      - Can **enforce an existing access path** from a PLAN_TABLE (hints) or
      - Can **customize optimization options** for a statement
The Access Path Repository (APR) is implemented during the ENFM mode of migration:
- SYSIBM.SYSQUERY
- SYSIBM.SYSQUERYPLAN
- SYSIBM.SYSQUERYOPTS

The new input table, DSN_USERQUERY_TABLE, is under the schema of a userid:
- SYSQUERY
  - Contains data for all optimization hints
- SYSQUERYPLAN
  - Contains data for optimization hints to enforce an access path
- SYSQUERYOPTS
  - Contains data for hints that customize optimization parameters
EXPLAIN Changes...

- **EXPLAIN tables must be in:**
  - V8 or later format (Otherwise -20008, RC 2 is returned)
  - Unicode encoding scheme
    - Is preferred for V8 & V9 formats
    - Required for V10 format (Otherwise SQLCODE -878 returned)
  - [DB2 10 Installation & Migration Guide – Chap. 2 – Converting EXPLAIN tables](#)

- **EXPLAIN CAPTURE Special Register (defaults to NO)**
  - Set to YES
    - Explainable SQL statements run normally and explain information is captured after each statement is prepared and executed
  - Set to EXPLAIN
    - Explainable SQL statements do not execute but explain information is captured during prepare
    - Note that application logic may fail when expecting successful execution
  - Requires a PLAN_TABLE and DSN_STATEMENT_CACHE_TABLE

- **The DSNAEXP stored procedure is deprecated**
  - This procedure allowed the EXPLAIN of a SQL statement with the authorization for the statement
  - The EXPLAIN privilege or SQLADM should be used
REBIND PLAN...COLLID

- DBRMs bound into plans has been deprecated.
- PK62876 provides for conversion of DBRMs bound in Plans to Packages in a Package List via the REBIND command
- REBIND PLAN...COLLID(‘collection-id’|*)
  - Default COLLID is DSN_DEFAULT_COLLID_planname
  - SYSDBRM & SYSSTMT records deleted
  - SYSPACKAGE & SYSPACKSTMT populated
Literal Replacement...

- Dynamic SQL statements with literals can now be made more reusable with literal replacement

- Enabled via:
  - PREPARE: CONCENTRATE STATEMENT WITH LITERALS in the ATTRIBUTES
  - ODBC: LITERALREPLACEMENT specified in the initialization file
  - JCC Driver: enableLiteralReplacement='YES'

- Statements that benefit by including literal values will run better:
  - Without literal replacement
  - Or with REOPT (all REOPT options supported with literal replacement)

- Better suited statements are likely smaller workload SQL statements with literal that could benefit from DSC

- Cannot have literal replacement and parameter markers
Statistics Enhancement... AutoStats

- **Autostats is implemented through a set of Stored Procedures**
  - Stored procedures are provided to enable administration tools and packaged applications to automate statistics collection.
  - **ADMIN_UTL_MONITOR**: Look for statistics needing update
  - **ADMIN_UTL_EXECUTE**: Execute RUNSTATS
  - **ADMIN_UTL_MODIFY**: Clean up the Autostats Repository
  - **ADMIN_ADD_TASK**: Called by MONITOR/EXECUTE & start up
  - **STATS ADVISOR (Data Studio)**

- **Run automatically honoring specified time windows**

- **Working together, these procedures**
  - Determine what stats to collect
  - Determine when stats need to be collected
  - Schedules and Performs the stats collection
  - Records activity for later review

See Chapter 11 "Designing DB2 statistics for performance" in the DB2 10 for z/OS Managing Performance Guide for details on how to configure autonomic monitoring directly within DB2.

Also see the ADMIN_UTL_MONITOR definition for criteria of Missing, Out of Date, or Inconsistent statistics.
Statistics Enhancement... AutoStats

- **Configuration / Communication via DB2 catalog tables**

  - **SYSAUTOTIMEWINDOWS**
    - Defines when autonomic procedures can be run
  
  - **SYSAUTORUNS_HIST**
    - Keeps history of what procedures have executed autonomically
  
  - **SYSAUTOALERTS**
    - Populated when MONITOR procedure detects that an action needs to be scheduled for execution (e.g. RUNSTATS needs to be scheduled)

  - **SYSTABLES_PROFILES**
    - Contains the RUNSTATS options for a particular table
Statistics Enhancement... Profiles / Performance

- **RUNSTATS**
  - New options to SET / UPDATE / USE a statistics profile
    - RUNSTATS ... TABLE tbl COLUMN(C1) ... **SET PROFILE**
    - Alternatively use **SET PROFILE FROM EXISTING STATS**
    - RUNSTATS ... TABLE tbl COLUMN(C5) ... **UPDATE PROFILE**
    - RUNSTATS ... TABLE tbl **USE PROFILE**
  - New option to do page-level sampling
    - RUNSTATS ... TABLE tbl **TABLESAMPLE SYSTEM AUTO**
Histogram Statistics - RUNSTATS

- Summarizes data distribution on an interval scale
- DB2 uses equal-depth histograms
  - Each quantile has about the same number of rows
  - Example - \(1, 3, 3, 4, 4\) \(6, 7, 8, 9\) \(10, 12, 15\)

  (sequenced), cut into 3 quantiles

<table>
<thead>
<tr>
<th>Seq No</th>
<th>Low Value</th>
<th>High Value</th>
<th>Cardinality</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>5/12</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>4/12</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>15</td>
<td>3</td>
<td>3/12</td>
</tr>
</tbody>
</table>
Real-Time Statistics Evolution

- **Real Time Statistics (RTS) have been collected automatically since V7**

- **DB2 9 moves RTS into the Catalog & externalizes the information automatically**
  - DSNDB06.SYSRTSTS
  - SYSIBM.SYSTABLESPACESTATS
  - SYSIBM.SYSINDEXSPACESTATS
  - During ENFM processing, the data is moved from user-defined tables to the new catalog tables
  - Drop user-defined tables after ENFM

- **Gather information about indexes that are used / not used**
  - SYSINDEXSPACESTATS.LASTUSED
  - "Used", as defined by DB2 is the following:
    - As an access path for query or fetch.
    - For searched UPDATE / DELETE SQL statement.
    - As a primary index for referential integrity.
    - To support foreign key access.
Virtual Indexes

- Determines the impact of creating or dropping an index WITHOUT creating or dropping the index
- Retrofitted into V8 via PK46687
- DSN_VIRTUAL_INDEXES table
  - Profile table used by Optimization Expert
  - Can be populated via an INSERT from SYSINDEXES
  - Includes several columns similar to SYSINDEXES, plus
    - MODE: ‘C’reate / ‘D’rop
    - ENABLE: ‘Y’es / ‘N’o
    - Column Number and Ordering occurrences.
- Only effects queries that specify EXPLAIN
- Don’t forget to disable (ENABLE=‘N’) or delete these rows, or they will continue to impact EXPLAIN
- Does not support Index on Expression
Index Enhancements

- Larger Page Sizes
- Asymmetric Page Splits
- Index Compression
- Index on Expression
- I/O Parallelism
- Include Columns
- More Index Enhancements
- Randomized Indexes
- Online Index Rebuild
- XML Indexes
Business value of index enhancements

- **Application performance and CPU savings**
  - Sequential insert operations see a vast improvement
  - Lessen frequency of REORG utility while maintaining access path consistency
  - Less waiting on I/O operations for applications
Index Page Sizes

- Larger index pages allow for more efficient use of storage
  - Fewer page splits for long keys
  - More key values per page

- Multiple processes inserting sequential keys can create hot spots on indexes
  - Randomized index keys avoid hot spots
  - Application throughput improved via avoidance of locking conflicts

- Rebuild Index
  - SHRLEVEL CHANGE
Asymmetric Index Page Splits

Multiple Sequential Insert Patterns on an Index

Sequential inserts into the middle of an index resulted in some pages with 50% free space prior to V9

New algorithm dynamically accommodates a varying pattern of inserts
Index Compression

- Solution provides page-level compression:
  - Data is compressed to 4K pages on disk
  - 32K/16K/8K pages results in up to 8x/4x/2x disk savings
    - Use DSN1COMP to estimate
    - May result in unused space with the large page so that it can be compressed into 4K
  - No compression dictionaries
    - Compression on the fly
    - No LOAD or REORG required
  - Version information in the index can reduce compression
  - Changes impacting index ordering will mark the index REBUILD PENDING (COMPRESS YES/NO, PADDED / NOT PADDED)
Index Compression Versus Data Compression

There are differences between index and data compression

<table>
<thead>
<tr>
<th></th>
<th>Data</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level</strong></td>
<td>Row</td>
<td>Page (1)</td>
</tr>
<tr>
<td><strong>Comp in DASD</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Comp in Buffer Pool</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Comp in Log</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Comp Dictionary</strong></td>
<td>Yes</td>
<td>No (2)</td>
</tr>
<tr>
<td>‘Typical’ Comp Ratio</td>
<td>10 to 90%</td>
<td>25 to 75% (3)</td>
</tr>
<tr>
<td><strong>CPU Overhead (4)</strong></td>
<td>In Accounting</td>
<td>In Accounting and/or DBM1 SRB</td>
</tr>
</tbody>
</table>

1. No compression or decompression at each insert or fetch; instead, it is done at I/O time
2. LOAD or REORG not required for compression; compression on the fly
3. Based on very limited survey and usage
4. CPU time impact under study – sensitive to index BP Hit Ratio, larger index BP recommended, higher impact on relatively unique indexes with long keys
Index on Expression

- Simple indexes can contain concatenated columns
  
  ```
  create index totalComp on W2_TABLE(salary, bonus)
  ```

- Index on expression
  - Value of the index has been transformed
  - May not be the value of any of the columns that it is derived from
  - Optimizer can use this index
  
  ```
  Create index totalComp on W2_TABLE(salary+bonus)
  ```

<table>
<thead>
<tr>
<th>name</th>
<th>salary</th>
<th>bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gary</td>
<td>20,000</td>
<td>500</td>
</tr>
<tr>
<td>Paul</td>
<td>40,000</td>
<td>5,000</td>
</tr>
<tr>
<td>Matt</td>
<td>400,000</td>
<td>10,000</td>
</tr>
<tr>
<td>Rex</td>
<td>40,000</td>
<td>2,000</td>
</tr>
<tr>
<td>Rob</td>
<td>200,000</td>
<td>210,000</td>
</tr>
</tbody>
</table>

Query / Order on Total Compensation
I/O Parallelism for Index Inserts...

- Transactions that perform inserts into tables with many indexes defined on the table previously may have had high response times due to index I/O wait time.
  - DB2 executes index updates sequentially

- **I/O parallelism : overlap the I/Os against non-clustering indexes**
  - Utilized if there are more than 3 indexes defined on the table and one of them is a clustering index, or 2 indexes if neither is a clustering index
  - DB2 can prefetch pages from different indexes defined on the same table into buffer pool in parallel for insert operation.

- For a table with 6 indexes and 2000 inserts, elapsed time improvements of up to 50% are measured

- This functionality is enabled for DB2 10 Conversion mode
Additional Non-key Columns In An Index

- Indexes are used to enforce uniqueness constraints on tables
- To achieve index only access on columns not part of the unique constraint, additional indexes are often created for the non-unique columns
  - Slower DB2 transaction time
  - Increased storage requirements
- In DB2 10 Additional Non-key Columns can be defined in a unique index to reduce total amount of needed indexes
- Indexes that participate in referential integrity (RI) will support additional columns, but INCLUDE(d) columns will not be used to enforce RI
- Improves:
  - Insert performance as less indexes need to be updated
  - Space usage
  - Can stabilize access path as optimizer has fewer similar indexes to choose from

```
CREATE UNIQUE INDEX ON t1(c1,c2,c3) INCLUDE (c4,c5)
```
More Index Enhancements

- RID list overflows to workfile instead of relational scan (MAXRBLK was 8MB, now 400MB)
  - Eliminate RID list failures from all four causes
  - DB2 9 had it for pair-wise join
  - MAXTEMPS_RID new ZParm

- Referential integrity check performance
  - Sequential detection and index look aside for RI
    - Avoid RI check for each insert of a child under the same parent
Distributed Data Enhancements

- DDF enhancements
- High Performance DBATs
- Elimination of private protocol
- Other Distributed workload improvements
Business value of DDF Enhancements

- Lessen CPU consumption of remote threads
  - Improved performance by means of internal restructuring of distributed processing on the server
    - Optimizes communication between DDF and DBM1
  - High performance DBAT can reuse threads very efficiently
High Performance DBAT…in 2 steps

- High Performance DBATs reduce CPU consumption by
  - Supporting RELEASE(DEALLOCATE) to avoid repeated package allocation/deallocation
  - xPROCs, CTs and PTs, lookaside and prefetch will not have to be re-initialized
  - Bigger CPU reduction for short transactions

- High Performance DBAT behavior …**STEP 1 package must be RELEASE(DEALLOCATE)**
  - DBAT will stay active with connection when DBAT is about to go be pooled and there is at least one RELEASE(DEALLOCATE) package still existing
  - Normal idle thread time-out detection will be applied to these DBATs. If DBAT is in completed unit-of-work status, Connection will turn inactive instead of being canceled (POOLINAC value controls this)
    - Connections will turn inactive after 200 times (not changeable) to free up DBAT (same as in previous releases)

- IBM plans to change default bind option for client packages from RELEASE(COMMIT) to RELEASE(DEALLOCATE)
High Performance DBAT...

- New -MODIFY DDF PKGREL(BNDOPT/COMMIT) command ...

**STEP 2**
- to alter DDF's inactive connection processing which is activated via the ZPARAM, CMTSTAT=INACTIVE
- Display command shows DSNL106I message with PKGREL= BNDOPT or COMMIT
- 2 options
  - PKGREL(BNDOPT) honors package bind option
  - PKGREL(COMMIT) forces package bind option
    - Same as v6- DB2 9 inactive connection behavior
    - Will allow BIND and DDL to run concurrently with distributed work
Handling of Private Protocol Requests in V10

- Jobs to analyze and prepare for private protocol elimination
  - PK64045 – delivered DSNTP2DP update, as well as DSNTPPCK

- Ability to Enable or Disable Private protocol (PK92339)
  - Via ZPARM PRIVATE_PROTOCOL in DSN6FAC Macro
  - Enables testing. PRIVATE_PROTOCOL=NO ZParm will mimic DB2 10 behavior in V8/V9

- DB2 10 will respond to a Private protocol response as follows, from a v9 or prior system
  - Reject request
    - VTAM sense code ‘10086021’
  - Requestor will receive
    - -904 Reason Code '00D31026' Type ‘1001’
  - DB2 10 DDF will reject a BIND with Private protocol
    - SQLCODE -30104

- Attempt to load/execute object with DBPROTOCOL column = ‘P’ will fail with SQLCODE -904, reason code 00E3001E, saying it needs to convert to DRDA, except the following case
  - If a package with PRIVATE PROTOCOL accesses local data only it is allowed. If it attempts to access remote data it will fail
Other Distributed workload improvements

- **64-bit exploitation by DDF**
  - Special “shared private” with DBM1 to eliminate many of the data moves on SQL operations

- **VTAM definition is now optional**

- **Support for SSL**
  - New security port (SECPORT)
  - Allow for separation of sensitive data flow connection
Other Distributed workload improvements

- **Support for IPv6**
  - NFM & TCPIP Dual Stack Support
  - 128-bit IP addresses (as opposed to 32-bit IPv4 addresses)
    - IPv4 addresses are mapped to IPv6 format (::FFFF:1.2.3.4)
  - Theoretically provides $6.6 \times 10^{23}$ addresses per square meter of the planet’s surface.
  - Colon Hexadecimal (as opposed to IPv4 dotted decimal) representation.
    - 2001:DB8:0:0:8:800:200C:417A
    - Double colon used to suppress multiple groups of zeros (can only be used once).
      - 2001:DB8::8:800:200C:417A
Other Distributed workload improvements

- Potentially 39 characters to display & store
  - IPADDR, LOCATION, LINKNAME
  - PK51019 for longer Location Name in DISPLAY output

- Changed to DSNJU003 & DSNJU004 to support
  - Also moved DDF information to the front of the report
Utility Enhancements

- Utility Highlights
- UTSERIAL Lock
- REORG Table Space
- REORG AUX
- REORG INDEX/REBUILD INDEX
- LOAD/UNLOAD
- UNLOAD
- COPY
- BACKUP/RECOVER
Business value statements

- Utilities are an integral part of DB2 to ensure availability while maintaining the highest degree of integrity and minimizing downtime
- Availability of the system catalog is improved by the removal of UTSERIAL locks so more utilities can run simultaneously
- Improved access to LOB data during REORG’s by allowing Reorg share level change
- We can now ensure utilities will complete by canceling blocking threads
- Recovery times can be minimized by rolling backward through the log
Utility Enhancements

- Unless explicitly stated, new utility functions are only available in NFM
- Improved access to SYSUTILX by removal of the UTSERIAL lock
- REORG SHRLEVEL(CHANGE) for complete catalog/directory
  - Hashes and links are removed during ENFM processing
  - Row level locks are relied upon
  - No changes to utility jobs are necessary
  - Greatly improves access to catalog/directory
- Improves concurrency for online REORG
  - Update of catalog statistic information for inline REORGs occurs after access to the table is allowed
Utilities Highlights

- Extensive support has been added to DB2 utilities for the pureXML
  - CHECK, COPY, LOAD, MERGECOPY, REBUILD INDEX, RECOVER, REORG, …

- More online utilities
  - Rebuild Index SHRLEVEL CHANGE
  - Reorg LOB now supports SHRLEVEL REFERENCE (space reclamation)
  - Check data, LOB and repair locate … SHRLEVEL CHANGE
  - Check index SHRLEVEL REFERENCE supports parallel for > 1 index

- Online REORG BUILD2 phase elimination
Utilities Highlights (cont.)

- COPY utility includes SCOPE PENDING support to improve usability
- The ability to recover to any point in time with consistency
  - Uncommitted changes are backed out
  - Significantly reduces (eliminates?) the need to run QUIESCE which can be disruptive to applications
- Fast log apply buffer default increased from 100MB to 500MB for RESTORE SYSTEM
  - LOGAPSTG remains @ 100MB
- Volume-based COPY/RECOVER
  - FlashCopy technology used to capture entire content of disk volumes
  - RECOVER modified to enable object-level recovery from volume FlashCopy
    • Restore assumes that the object has not moved volumes
  - Eliminates labor associated with setting up COPY jobs for each database / table space
  - Full integration of tape into BACKUP/RESTORE SYSTEM utilities
REORG...

- **REORG TABLESPACE**
  - **SHRLEVEL CHANGE/REFERENCE**
  - Materializes pending changes
    - Table space type, DSSIZE, SEGSIZE, data page size, index page size, member cluster
    - Partitioned table spaces must be REORG’d in their entirety
  - Runs even if table space is in hard REORP
    - Prior to DB2 10 required SHRLEVEL(NONE)
  - **REORG SHRLEVEL(NONE)**
    - Will not materialize pending DDL changes, but will proceed
REORG

- **LOB REORG**
  - **SHRLEVEL NONE in CMx / ENx**
    - DSNU125I REORG SHRLEVEL NONE ON LOB TABLE SPACE WILL BE RESTRICTED IN A FUTURE RELEASE OF DB2.
  - **SHRLEVEL NONE** is not supported in NFM
    - The job will end with RC=0 and NO LONGER SUPPORTED message
    - DSNU126I REORG SHRLEVEL NONE ON LOB TABLE SPACE IS NO LONGER SUPPORTED.

- **REORG TS with SHRLEVEL REFERENCE / CHANGE** with **AUX YES**
  - pending changes that are associated with the base table space are materialized
  - pending changes that are associated with the LOB table spaces are not materialized.
REORG TABLESPACE…

- Performance improvement for partition-level REORG with NPI’s and REORG INDEX
  - Index list prefetch results in up to 60% ET reduction

- Messaging to estimate log apply phase time to completion and SWITCH phase begin is estimated to miss the DEADLINE
  - SHRLEVEL REF or CHANGE only
  - DSNU383I in addition to DSNU105I

- REORG of multiple non-contiguous partitions allowed
  - Changes behavior from DB2 9
  - Customers who perform multi-part REORGs can now specify multiple parts in a single command
    - Ex: REORG TABLESPACE PARTS (1,5,10:15, 20:23)
  - Retro-fitted into v9 via PM22336
REORG TABLESPACE…

- **REORG** can CANCEL threads on last RETRY processing
  - Ensures **REORG** can break in on applications
  - Allows **REORG** to cancel threads that hold claims on objects that prevent drains
    - FORCE NONE|READERS|ALL

- **New Advisory REORG Pending status – AREOR**
  - Indicates that there are pending alters that require a table space level **REORG**

- **Statistics**
  - Now defaults to STATISTICS TABLE ALL INDEX ALL UPDATE ALL HISTORY ALL
REORG with LOB columns…

- New keyword AUX used in REORG to specify how to handle AUX table spaces
  - Allows LOB rows to flow with base table rows for partitioned table spaces
  - Allows REBALANCE
  - Allows ALTER LIMITKEY
  - Makes sure associated LOB rows are deleted with DISCARD
  - May leave auxiliary spaces COPYP

- **REORG…AUX(YES|NO)**
  - YES – Include LOB table spaces in the REORG
  - NO – v9 behavior
  - Default = YES in the following cases:
    - REORG of entire user PBG space with LOB(s)
    - REORG with REBALANCE, DISCARD, or to materialize altered limit keys
    - REORG of SPT01 if in REORP or AREOR status
REORG INDEX

- **REORG INDEX SHRLEVEL(REFERENCE or CHANGE)**
  - Will materialize pending index changes so long as there are no pending table space changes
  - If there are pending TS changes, REORG proceeds and message DSNU275I RC=4 is issued indicating pending changes were not materialized
  - For indexes defined COPY YES, an entry will be made in SYSCOPY

- **REORG INDEX SHRLEVEL(NONE)**
  - Will not materialize pending index changes
  - DSNU1165I RC=4 will be issued indicating pending changes were not done
  - For indexes defined COPY YES, an entry will be made in SYSCOPY
REORG INDEX

- When pending definitions are materialized in REORG INDEX with SHRLEVEL REFERENCE or CHANGE
  - Index statistics are collected by default and updated in the catalog
  - Default = STATISTICS UPDATE ALL HISTORY ALL

- STATISTICS if used
  - Will override defaults
  - New warning message will indicate some partition stats may no longer be accurate due to materialization of pending changes.
  - Users should execute RUNSTATS to recollect partition statistics to re-gather COLGROUP, KEYCARD, HISTOGRAM, frequency stats with NUMCOLS>1, and statistics for extended indexes
UNLOAD Spanned Record Support…

- LOB and XML columns can now be UNLOADed or LOADed to/from the same data set with other non-LOB/XML columns via spanned records
  - Performance of reading from or writing to a single sequential file is much faster than using separate files or partition data set members because the utilities will not have to use Open/Close for each LOB
  - When unloading LOB/XML columns to a sequential file, LOBs and XML documents will be written at the end of the record in their column definition order
UNLOAD

- **Spanned records support for UNLOAD**
  - `UNLOAD TABLESPACE xxx.yyy SPANNED YES FROM TABLE ttt (field spec list with CLOB and xml data at end)`
  - Option for records > 32K
  - FRV were needed in the past
FLASHCOPY SUPPORT…

- **Dataset-level Flashcopy support**
  - For COPY, REORG, LOAD, REBUILD INDEX, REORG INDEX, CHECK
  - Significant CPU and elapsed time reductions vs image copy

![Graph showing CPU time per object (z10)](image1)

![Graph showing elapsed time per object (z10)](image2)
COPY

- Create transaction consistent image copies from a COPY SHRLEVEL CHANGE run using Flashcopy
  - DB2 10 uses the Flashcopy copy to make a transaction consistent image copy.
  - Don’t have to wait for blockers. No application outage and no quiesce.
  - Uncommitted units of work are rolled back against the Flashcopy
  - Image copy is then taken from the Flashcopy
  - Objects must be SMS managed residing on FC v2 volumes

- Improved dataset management & performance
  - Will not allocate incremental copy dataset unless copy taken
  - &ICTYPE now matches actual image copy
    - Regardless of the type specified on a template i.e. TYPE=C
  - Utilize RTS for CHANGELIMIT performance
**RECOVER**

- **Point-in-time recovery with BACKOUT**
  - Prior to DB2 10 recovered from a backup image then applied logs to reach the desired point in time.
  - DB2 now provides an option on utilities that specifies if the recovery should be performed by processing the log backwards or in the traditional restore and roll forward through the log process.
  - Changes are backed out from the current state. IC not used.
  - Customer decides if BACKOUT is desirable option to use
    - IBM Recovery Expert can assist in making the best Recovery decision
  - In order to work successfully, indexes must be marked COPY YES since DB2 needs the SYSCOPY records for non-recoverable events to know whether the RECOVER can proceed or not.
    - PM30991 HIPER should be on to disallow a BACKOUT recovery when certain unlogged events exist.

```sql
RECOVER TABLESPACE myschema.mytablespace BACKOUT YES
TOLOGPOINT x'00000518DFEF'
```
DFSsort

- PK59399
  - Utilities will read DFSsort install parameters from PARMLIB with z/OS 1.10.

- PK85889
  - Allows utilities to redirect portions of sort workload to zIIPs (fixed length records)
  - Indicated via MSGICE256I DFSORT CODE IS ELIGIBLE TO USE ZIIP FOR THIS DB2 UTILITY RUN.
### Virtual Storage Improvements

- **64-bit Evolution (VStor Relief)**
- **Usage of z/OS Shared Private Storage**
- **More Storage Above the Bar**
  - Ability to run more threads
- **In-memory Buffer pool objects**
- **Workfile Enhancements**
  - Converged TEMP Space
  - Spanned Workfile records
  - PBG support for workfile database

- **1 MB Page size**
- **System Latching changes**
- **On Demand Buffer Pool storage**
Virtual Storage Business Value

- Changes that began in force in V8
- Can offer significant constraint relief
- Leads to greater scalability
- Can reduce administration and reduce the need for horizontal scaling
- Enables new performance options in DB2 10
64 bit Evolution (Virtual Storage Relief)

DIST

IRLM

16 EB

DBM1

- Buffer pool
- EDM pool
  - DBDs
  - Global Dyn Stmt Cache
- RID pool
- Sort pool
- Compression dictionaries
- Castout Buffers
- Buffer Mgr Control Blocks
- Trace Tables
- Accounting Blocks
- Parse tree
- EDM pool
  - SKPT / SKCT
  - Portions of CT / PT
  - Portions for stmt cache
  - Page set & RTS blocks
  - Some thread Runtimes
- Archive Log Buffers (R/W)

The bar

2 GB

The bar

0

16M - Line

I R L M

DIST

Environmental Descriptor Manager Pool

Skeleton Pool (EDM_SKELETON_POOL)
  SKCT/ SKPT

Global Stmt Pool (EDMSTMTC)

DBD Pool (EDMDBDC)

EDM Pool (EDMPOOL)
  SKCT / SKPT
  CT / PT
  Others

IFCID 217: detailed DBM1 virtual storage health
IFCID 225: consolidated DBM1 virtual storage health
IFCID 002: Mapping macro for EDM Pool, DSNDQISE

Enabled in CM
64 bit Evolution (Virtual Storage Relief)

- Com Buffers
- Control Blocks
- Shared Memory

DIST

Enables in CM

Environmental Descriptor Manager Pool

- SKCT / SKPT
- CT/PT

Global Stmt Pool (EDMSTMTC)

DBD Pool (EDMDBDC)

EDM Pool (EDMPOOL)

- XPROCS
- Some stack code
- Some DB2 code
- Thread storage

DBM1

- Buffer pool
- EDM pool
  - DBDs
  - Global Dyn Stmt Cache
- RID pool
- Sort pool
- Compression dictionaries
- Castout Buffers
- Buffer Mgr Control Blocks
- Trace Tables
- Accounting Blocks
- Parse tree
  - EDM pool
  - SKPT / SKCT
  - Portions of CT / PT
  -- Portions for stmt cache
- Page set & RTS blocks
- Some thread Runtimes
- Archive Log Buffers (R/W)

16 EB

16M - Line

2 GB

The bar

Enabled in CM

© 2011 IBM Corporation
DB2 10: 64 bit Evolution (Virtual Storage Relief)

Scalability: Virtual storage constraint is still an important issue for many DB2 customers, until DB2 10

- DB2 10 supports 5-10x more active threads, up to 20,000 per member
  - 80-90% of thread storage moved above the bar
  - More concurrent work
  - Reduce need to monitor
  - Consolidate members and LPARs
  - Reduced cost, easier to manage, easier to grow
  - REBIND required to get most of the savings
V9 Improvements for VStor Constraint Relief

- **DIST address space runs in 64-bit addressing mode**
  - Shared 64-bit memory object avoids xmem moves between the DBM1 and DIST address spaces and improves performance
  - Constraint relief

- **Reduced amount of data formatting and data movement for DRDA**
  - It exists once, instead of in each address space

- **New virtual storage type sharable by multiple address spaces that register with z/OS**

- **Has storage key & fetch protection**

- **Defaults to 2TB size**

- **DB2 requires a minimum of 128GB configured or V9 will not run**
  - Even if not running DIST

- **Set by HVSHARE in Parmlib**

- **DISPLAY VIRTSTOR,HVSHARE**

**z/OS Shared Private Storage**

- 2GB Bar

**RESPONSE=MY50**

IAR019I 23.20.10 DISPLAY VIRTSTOR 472
SOURCE = DEFAULT
TOTAL SHARED = 522240G
SHARED RANGE = 2048G-524288G
SHARED ALLOCATED = 131072M
Running a Large Number of Threads

**Pre DB2 10**

- **LPAR1**
  - DB2A (500 thds)
  - DB2D (500 thds)

- **LPAR2**
  - DB2B (500 thds)
  - DB2E (500 thds)

- **LPAR3**
  - DB2C (500 thds)
  - DB2F (500 thds)

**Coupling Technology**

- Data sharing and sysplex allows for efficient scale-out of DB2 images
- Sometimes multiple DB2s / LPAR

**DB2 10**

- **LPAR1**
  - DB2A (2500 thds)

- **LPAR2**
  - DB2B (2500 thds)

- **LPAR3**
  - DB2C (2500 thds)

**Coupling Technology**

- More threads per DB2 image
- More efficient use of large n-ways
- Easier growth, lower costs, easier management
In Memory Table Spaces…

- We are able to let DB2 know that you have cached an entire tablespace into a buffer pool.
  - Data will be preloaded into buffer pool when object is opened and will remain until closed.
    > Specified by PGSTEAL=NONE
  - Excellent for lookup tables and indexes
    > Small tables that have high I/O rates
  - Avoid LRU chain maintenance and LC14 contention
  - Avoid unnecessary prefetch and LC24 contention
  - IFCID 201 & 202 will be updated to signify this condition
WORKFILE ENHANCEMENTS

- Converged TEMP Space
- Spanned Workfile records
- PBG support for workfile database
- In memory workfiles
Converged TEMP Space

- **Single source for all temporary space in DB2**
  - Workfile (work files and Created Global Temporary Tables)
  - Temp DB (Static Scrollable Cursors and Declared Global Temporary Tables)
  - Merged into Workfile Database
  - In CM & NFM

- **The workfile database is the only temporary database**

- **Supports 4K and 32K page sizes, with automatic selection of the appropriate page size**
  - Expect an increased use of the 32K temp space
  - Consider sizing 32K buffer allocation @ 50% - 100% of the 4K size
    - Monitor statistics and adjust for actual usage

- **New ZParm for preventing workfile monopolization (MAXTEMPS)**
Converged TEMP Space (continued)

- **IFCID 002 & 343 updated to report usage and exceptions**
  - Make sure to review PK70060 & PK92163
    - Soft separation of sort workfile and DGTT requests (DB2 Managed Only)
    - Work files marked unavailable after extents
  - PM02528
    - ZParm WFDBSEP for hard separation of workfile and DGTT usage
  - II14587 – DB2 9 workfile recommendations

- **Access is virtualized for small amounts of data, eliminating cost of work file creation (reduced CPU and I/O)**
  - At runtime, a result fitting in 1 data page does not create a workfile
  - ORDER BY and FETCH FIRST n ROWS without index support
    - Uses memory replacement technique if result fits within a 32k page
    - Sort is avoided
Spanned Workfiles

- Spanned workfile records solves problems when sorting > 32K records
- Records for joins & large sorts can now span pages
- Max record length for workfiles is increased to 65,529 bytes
- Max length of sort key is increased from 16,000 to 32,704 bytes
- Alleviate -670 when length of join is larger than max page size in workfile db
- NFM Only
DGTT changes …

- **DGTT can not span workfiles**
  - We can define DGTT workfiles in a PBG
  - Alleviate contention between DGTT and other workfile usage
  - Ability to expand workfiles to avoid abends
  - MAXPARTITIONS allowed for tablespaces in WORKFILE, AS well as DSSIZE and NUMPARTS (CREATE ONLY / no ALTER)
  - SYSTABLESTATS will record workfile data at the partition level for PBG tablespaces (disk storage monitor may need to change)
  - New partitions will remain through DB2 restart
  - ZParms in v9 & v10 to allow for soft and hard separation of DGTTs and other workfiles
DGTT changes

You can define the workfiles as PBGs so that the DGTT can grow larger than 64 GB in one tablespace.

There are workarounds in v9 to simulate this behavior.
In Memory Workfiles

- DB2 10 supports simple predicate evaluation. This enhancement reduces the CPU time for workloads that execute queries that require the use of small workfiles
- Available in CM mode, no REBIND needed
1 MB Page Size (Frame size)

- DB2 takes advantage of the new 1 MB page size on the z10 and later
- Specify PGFIX=YES to get 1 MB page size in your buffer pools
- Must be backed by real storage
- Must allocate space above the bar with LFAREA parm in IEASYSxx in Parmlib
- Potential for significant performance improvements
- Using the 1 MB page size will enable efficiencies in the hardware
System latching changes

- **Log latch reduction**
  - Hold log latch for bare minimum of time
  - reduced Latch Class 19 contention
  - reduced other latch classes
  - Will help with both data sharing and non data sharing

- **LRSN in data sharing does not have to be unique within a page.**
  - Major performance improvement for data sharing environments
On Demand Buffer Pool storage

- This gives you the ability to grow your buffer pool over time as opposed to allocate it all at first usage
- Maximum size will be VPSIZE
- This can be helpful to large buffer pools, where the pool will not grow to the maximum size over the length of time until DB2 is recycled.
- For Index only query, no data pages will be allocated in the buffer pool
  - Only index pages will be allocated
  - V9 – Index only on non padded indexes on varchar columns
  - v10 – Additional columns in index can make more queries index only
Serviceability and Instrumentation

- DISPLAY Commands
- Active Log Data Sets
- Autonomic Checkpoint
- Enhanced Monitoring Support
  - Statement Level
  - System Level
- Health Monitoring
- Statistics Management
- Start up messages
- Resource Limit Facility
- Serviceability items
- zIIP Expansion
DISPLAY Commands

- DISPLAY UTILITY
  - Recover Utility
    - Provides an estimate for log processing completed.
  - REORG SHRLEVEL(CHANGE) or SHRLEVEL(REFERENCE)
    - Now shows progress for subtasks processing in parallel

- DISPLAY THREAD(*) TYPE(SYSTEM)
  - List system type thread
    - Like Commands
  - If command is cancelable
    - A non-0 token will be provided
DISPLAY Commands

- **DISPLAY THREAD(*) TYPE(PROC)**
  - Displays thread executing Stored Procedures & UDFs
  - DSNV401I begins the report

- **DISPLAY THREAD(*) SERVICE(WAIT)**
  - Command automatically driven @ 1 min interval
  - Boosts priority of tasks in the way of high priority tasks
  - Avoids DISCRETIONARY workloads from starving PRIORITY workloads when CPU % is high
  - Made available to V7 / V8 with maintenance (PK01230)
Active Log Data Sets

- Ability to add new active log data sets without having to recycle DB2.
- NEWLOG and COPY keywords added to the –SET LOG command.
  
  SET LOG NEWLOG(dsni) COPY(log-copy)
  
  - dsn = name of newly defined active log data set
  - log-copy = “1” for copy-1 or “2” for copy-2
  - Changes are pervasive.

- Data set must be defined with IDCAMS before issuing command.

- Recommendations:
  - Format data set with DSNJLOGF utility before issuing command.
  - Add both copy 1 and 2 for dual logging.

- New DB2 messages:
  - DSNJ363I: COPY\(log-copy\) LOG DATA SET \(dsn\) ADDED TO THE ACTIVE LOG INVENTORY
  - DSNJ364I: NEWLOG OPERATION FAILED: \(reason\)
Autonomic Checkpoint

Currently DB2 allows checkpoints to be scheduled based on either number of log records created or a fixed time interval.

DB2 10 enhances checkpoints thresholds to be scheduled on either or both.

- Provides ability to use log records when subsystem is busy, and time interval when subsystem is relatively inactive.
Autonomic Checkpoint…

- **ZParm changes:**
  - New **CHKTYPE** defines threshold method being used.
    - SINGLE – either number of log records or time interval
    - BOTH – both methods used
  - New **CHKLOGR** defines log record threshold, when CHKTYPE = BOTH. Otherwise set to NOTUSED.
  - New **CHKMINS** defines time interval threshold, when CHKTYPE = BOTH. Otherwise set to NOTUSED.
  - **CHKFREQ** defines log record or time interval threshold, when CHKTYPE = SINGLE. Otherwise set to NOTUSED.
Autonomic Checkpoint…

**SET LOG command changes:**
- Provides ability to temporarily make changes to checkpoint method being used and values.
  - Values are reset to values specified in ZParm when DB2 recycled.
- Parameters SINGLE and BOTH added
- Syntax:

```
SET LOG
  |     |     |
  | SINGLE | LOGLOAD(integer) |
  |        | CHKTIME(integer) |
  | BOTH   | LOGLOAD(integer) |
```

- If SINGLE is specified, then CHKTIME or LOGLOAD must be specified.
- If BOTH is specified, and either (or both) CHKTIME or LOGLOAD is not specified, then previously set values or defaults are used.
  - CHKTIME default = 5
  - LOGLOAD default = 500000
- Use `DISPLAY LOG` command to display current checkpoint method being used and values.
Enhanced Monitoring Support

- **DB2 10 enhanced monitoring support is provided as two independent functions.**
  - Statement level
  - System level

- **At the statement level this supports performance monitoring and problem determination for both static and dynamic SQL.**
  - Uses IFI to capture and externalize information for consumption by tooling.
  - Introduces a unique statement execution identifier (STMT_ID).

- **At the system level this provide increased granularity of monitoring system level activities.**
Enhanced Monitoring Support… Statement Level

- **Problem determination enhancements at the statement level:**
  - Externalize statement type (dynamic or static) and STMT_ID in existing messages, including those related to deadlock, timeout, and lock escalation.
  - Information associated with thread information (THREAD-INFO) in these messages.
  - Can be used to correlate the statement execution on the server with the client application.

- **Performance monitoring enhancements at the statement level:**
  - Existing trace records modified to capture:
    - statement type
    - STMT_ID
    - statement-level performance metrics
  - New trace records are introduced to provide:
    - Access to performance monitoring statistics in real time.
    - Allow tooling to retrieve monitoring data without requiring disk access.
Enhanced Monitoring Support… System Level

- **Enhancements for monitoring system level activities:**
  - Profile table facility enhanced to provide improved threshold monitoring and filtering through new fields/keywords.
    - Enhancements apply only to system level monitoring related to threads and connections, not to statement level monitoring and tuning.
  - Monitor profile extended to support monitoring of:
    - idle thread timeout
    - number of threads
    - number of connections
  - Monitor profile includes ability to filter by ROLE and client product-specific identifier (PRDID).
    - Provides a finer degree of control over the monitor profiles.
  - Allows thresholds to be enforced at a more granular level, which were previously available at the system level via ZParm.
  - Provides greater flexibility and control for allocating resources to particular clients, applications and users according to priorities or needs.
Enhanced Monitoring Support… System Level

- Monitoring system level activities related to threads and connections:
  - Following profile tables must be created:
    - DSN_PROFILE_TABLE
    - DSN_PROFILE_HISTORY
    - DSN_PROFILE_ATTRIBUTES
    - DSN_PROFILE_ATTRIBUTES_HISTORY
    - Some may have been created in by optimization tools, and may require modifications.
    - Tables will be created by DSNTIJSG job.
Health Monitoring

- **Internal Service Monitor**
  - Drives – `DISPLAY THREAD SERVICE(WAIT)` code at 1 minute intervals
  - Automatically run from restart to shutdown
  - 3 monitors created at startup for failover purposes.
    - 1 in MSTR, 1 in DBM1, and 1 in DIST
    - Active and inactive monitors are displayed via
      - `DISPLAY THREAD(*) TYPE(SYSTEM)` output with a DSNV507I message:

\[
\text{V507-ACTIVE MONITOR, INTERVALS=1235, STG=47\%, BOOSTS=0, HEALTH=100%}
\]
Health Monitoring

- **DB2 Health**
  - Reported to WLM, not just available or busy.
  - Avoids routing work to a member that’s completing many tasks but they are abending.

- **DBM1 Storage Monitoring**
  - Watching DBM1 31 bit storage
  - Reports monitoring storage thresholds via SMF data or online monitors
  - Internal monitor will automatically issue console messages when storage crosses (increasing or decreasing) 88%, 92%, 96% and 98% consumption
  - DSNV508I reports reaching critical thresholds
  - DSNV512I report on agents using the most storage

  ```
  DSNVMON ) DB2 BELOW-THE-BAR STORAGE WARNING
  93% CONSUMED
  92% CONSUMED BY DB2
  ```
Statistics Management

- **SMF Compression**
  - Via the new **SMFCOMP** ZParm
  - Accounting records may compress 80-90%
  - CPU Overhead ~1%
  - When **ACCUMACC** ZParm is also set (default to 10) can compress up to 99%
    - DB2 10 rollup accounting provides more detail
  - **PM27872**
    - Adds job DSNTSMFD to decompress SMF records

- **Statistics interval changes**
  - STATIME default reduced to 1 minute
    - IFCIDs 0105, 0106, 0199, and 0365
    - IFCIDs 0001, 0002, 0202, 0217, 0225, and 0230 are no longer controlled by STATIME
    - Corresponding trace records are written at fixed, one-minute intervals
Start Up Messages

- System parameter messages at start up

  ```
  SUBSYSTEM ISC9  SYSTEM PARAMETERS ARE AS FOLLOWS:
  SYSTEM PARM   SYS2.DB2.V910.SDSNEXIT(DSNZISC9)
  DSNHDECP      SYS2.DB2.V910.SDSNEXIT(DSNHDECP)
  ACCESS EXIT   SYS2.DB2.V910.SDSNEXIT(DSNX@XAC)
  IDAUTH EXIT   SYS2.DB2.V910.SDSNEXIT(DSN3@ATH)
  SIGNON EXIT   SYS2.DB2.V910.SDSNEXIT(DSN3@SGN)
  ```

- Start up and shutdown progress messages

  ```
  DSNY024I  ) DSNYASCP MSTR INITIALIZATION IS STARTING
  DSNY024I  ) DSNYASCP DBM1 INITIALIZATION IS STARTING
  DSNY025I  ) DSNYASCP DBM1 SHUTDOWN IS COMPLETE
  DSNY025I  ) DSNYASCP MSTR SHUTDOWN IS COMPLETE
  ```

The Precompiler will also display the fully qualified data set name for module DSNHDECP
Resource Limit Facility Enhancement

- **RLF will be enhanced to allow CPU cost to be controlled based on:**
  - Client workstation name
  - Client application name
  - Client userid
  - IP address

- **There is a new RLF table: authid.DSNRLMTnn**
  - RLF will start if either (or both) DSNRLSTnn or DSNRLMTnn exists
  - Created in job DSNTIJSG for installation or migration

- **Allows fine-grain RLF limits for apps that exploit the Set Client Information APIs (SAP, PeopleSoft, Siebel, WebSphere)**
Serviceability

- **Improved usability of integrity verification tools**
  - Online CHECK INDEX
  - Online CHECK DATA
  - Online CHECK LOB
  - Restrictive states not set if error detected by online utility
  - COPY CHECKPAGE no longer sets CHKP if error detected

- **Improved FFDC for utilities**
  - Additional deadlock & timeout information supplied in job output
  - Date and time now included in utility SYSPRINT messages

- **DSN1PRNT FORMAT formats page even if broken**
Serviceability

- **Improved logging for SYSUTILX updates**
  - Change data capture enabled
  - Complete records captured for improved diagnosis of utility issues

- **Improved logging of DBET information**
  - SCA DBET data check pointed periodically

- **Restart**
  - New options to specify timestamp on Conditional Restart
  - Initiate page set open earlier in forward log phase
    - Asynchronous page set open at log read rather than waiting until LOGAPPLY
  - Defer closing SYSLGRNX ranges until after restart

- **After the IPL for DB2 9 ERLY code**
  - ERLY code can be refreshed with
    - REFRESH DB2,EARLY
zIIP Expansion & WLM

- **zIIP Expansion**
  - Prefetch & Deferred Write processes
  - RUNSTATS

- **WLM Minimum # of Address Spaces**
  - PM27097 adds MNSPAS parameter to the WLM startup JCL
  - Starts this number of address spaces initially
  - Available to DB2 9 as well
DB2 for z/OS Tooling
**Tooling**

- Developer Workbench
- Data Studio
- Optimization Service Center
- Optim Query Workload Tuner
- pureQuery
- DB2 .Net Add-Ins for Visual Studio
- Text Search Support
- Optimize Dynamic Infrastructure Performance
### Comparison - Developer Workbench vs. IBM Data Studio

<table>
<thead>
<tr>
<th>2006</th>
<th>2007</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IBM DB2 Developer Workbench V9.1</strong></td>
<td><strong>IBM Data Studio</strong></td>
</tr>
<tr>
<td>- SQL Query Editor</td>
<td>- Integrated Query Editor</td>
</tr>
<tr>
<td>- SQLJ Editor</td>
<td>- SQLJ Editor</td>
</tr>
<tr>
<td>- SQL Builder</td>
<td>- SQL Builder</td>
</tr>
<tr>
<td>- XQuery Builder</td>
<td>- Integrated Query Editor</td>
</tr>
<tr>
<td>- SQL Routine Debugger</td>
<td>- SQL Routine Debugger</td>
</tr>
<tr>
<td>- Java Routine Debugger</td>
<td>- Java Routine Debugger</td>
</tr>
<tr>
<td>- XML Editor</td>
<td>- XML Editor</td>
</tr>
<tr>
<td>- XML Schema Editor</td>
<td>- XML Schema Editor</td>
</tr>
<tr>
<td>- Data Management</td>
<td>- Data Management</td>
</tr>
<tr>
<td>- Visual Explain</td>
<td>- Visual Explain</td>
</tr>
<tr>
<td>- Project Management</td>
<td>- Project Management</td>
</tr>
</tbody>
</table>

**Data Studio is a full replacement of DB2 Developer Workbench, plus more**
Data Studio & Optimization Service Center

- **IBM Data Studio “Directional” – Support beyond DB2 9**
  - New no-charge product
  - Designed to provide database developers and DBAs with the basic set of required tools for database development and administration.
  - Available via web download:
  - IDE (Eclipse based) – Recommend for shell sharing with other tools.

- **DB2 for z/OS Optimization Service Center (EOS with DB2 9)**
  - New no-charge product
  - Simplifies the monitoring and tuning of SQL statements and workloads.
  - Available via web download:
Optim Query Workload Tuner

- **Optimize performance**
- **Reduce costs**
  - Increase overall system capacity
  - Simplify analysis for DBAs, developers, designers, and others
  - Reduce time to respond
- **Development Environment**
  - Tune SQL pre-production while costs and impact are low
- **Production Environment**
  - Optimize workload for peak performance
- **Identify query candidates from numerous sources**
  - DB2 catalog
  - Dynamic statement cache
  - Development Studio hot spots
  - Query or performance monitors

- **Get expert tuning advice**
  - Improve query design
    - Get recommendations regarding how to rewrite the query for better efficiency
  - Improve statistics quality
    - Get recommendations on the best statistics to capture to influence access path selection
  - Improve index design
    - Get recommendations on additional indexes that can reduce database scans

- **Analyze Access Plans**
  - Visualize access path
  - Assess access path stability to reduce risk of performance regression

- **Eclipse based**
  - Allows for shell sharing with other tools
pureQuery on z/OS

pureQuery is a high-performance, data access platform to simplify developing, managing, securing, and optimizing distributed data access.

Why use pureQuery?

- pureQuery provides static SQL for JAVA and .Net programs
- pureQuery runtime is available for Java z/OS applications
- pureQuery works with SQL and stored procedures (SPs)
- pureQuery can be added to existing applications without changing any line of code

Value of pureQuery:

- Reduces cost
  - Static Vs Dynamic
  - CPU
  - Development time
- Improves quality of service for new and existing Java applications
- Reduces development time for new Java applications
- Enhances security
DB2 .Net Add-Ins for Microsoft Visual Studio

- **Tight integration with Microsoft Server Explorer to create connections**
- **Key highlights:**
  - Debugging support for z/OS V8, 9 and 10
  - DB2 XML tools support for LUW and z/OS
  - Continued web services for DB2 embedded application server and Microsoft IIS
  - Rich designers to create procedures, tables, views, triggers, indices, roles and privileges
  - Alter and clone functionality
  - Independent installer for the Visual Studio Add-ins

- **IBM database support for**
  - z/OS (V7, V8, 9 and 10), iSeries (v5.1, 5.2, 5.3) and Linux, Unix, Windows (v8, v9)
  - WS II Federated database and nicknames

- **Works with:**
  - Visual Studio 2003 (.NET Framework 1.1),
  - Visual Studio 2005 (.NET Framework 2.0)
  - Visual Studio 2010 (.NET Framework 4.0)

- **Included with DB2 9 Runtime Clients**
OmniFind - Text Search Support

- No-charge feature provided with DB2 for z/OS
- Provides text search for CHAR/VARCHAR/LOB/XML columns
  - Provides support for CONTAINS and SCORES functions
- Text index server
  - Runs on Linux on Intel & Windows
- Efficient communication interactions with DB2 for z/OS
- Text indexes are persisted into DB2 tables for backup/recovery purposes
Optimize Dynamic Infrastructure Performance
OMEGAMON XE for DB2 Performance Expert 5.1 Exploitation

- **Extended Insight**
  - Surface DB2 for z/OS end-to-end response time metrics
    - Visibility to all the components that make up end-user response time
    - Facilitates platform-agnostic identification of response time bottlenecks
    - Enables near-instantaneous response to and prevention of application slowdowns
  - Leverages Tivoli Enterprise Portal GUI
  - Support DB2 9 & 10

- **Summary SQL Reporting**
- Manage thousands of Threads
- Support new DB2 10 Monitoring Data
- Lower Monitoring Overhead
- 25% zIIP offload of Near Term History *

* Customer results may vary, results based on lab testing
DB2 for z/OS Manuals Documentation

- **DB2 V8 for z/OS Manuals:**

- **DB2 9 for z/OS Manuals:**

- **DB2 10 for z/OS Manuals:**

- **Information Center:**  [http://publib.boulder.ibm.com/infocenter/imzic](http://publib.boulder.ibm.com/infocenter/imzic)
  - Primary documentation delivery mechanism
  - Search capabilities across
    - Redbooks
    - Whitepapers
    - Technotes
    - APARs
DB2 for z/OS Redbooks

  - DB2 V8 for z/OS Everything You Ever Wanted to Know… and More (SG24-6079)
  - DB2 V8 for z/OS Performance Topics (SG24-6465)
  - DB2 9 for z/OS Technical Overview (SG24-7330)
  - DB2 9 for z/OS Performance Topics (SG24-7473)
  - DB2 10 for z/OS Technical Overview (SG24-7892)
  - DB2 10 for z/OS Performance Topics (SG24-7942)
  - DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond (SG24-7604)
  - Extremely pureXML in DB2 10 for zOS (SG24-7915)
DB2 for z/OS… Other Documentation

- **DB2 9 for z/OS Web Page:**
  

- **DB2 10 for z/OS Web Page:**
  

- **DB2 for z/OS Library:** (links to additional documentation)
  

- **IBM developerWorks: DB2 for z/OS best practices**
  

- **IBM developerWorks: My developerWorks Exchange**
  
Thank You