DB2 10 Technical Discussion

Bill Arledge
Apr 2011
Presentation Agenda

• DB2 10 overview (Quick Recap)
• Details on a few new capabilities
  – Temporal Tables
  – Hash table access
  – New security paradigm
• Open discussion on your plans for DB2 10
  – What interests you about DB2 10?
  – What’s your current schedule?
  – Will you use skip migration from DB2 V8 to 10?
DB2 10 Overview and Marketing Message

• Cost reduction right out of the box
  – 5 – 10% CPU reduction for existing workloads (must rebind)
    • Numbers vary
  – 10 – 20% for nontraditional workloads (BI, etc.)

• Greatly enhanced scalability
  – 10X as many active threads (20,000)
  – Ability to consolidate workloads to fewer members (Banco do Brasil case study)
  – Enhanced 64-bit storage eliminates most (almost all) virtual storage constraints below bar

• Skip migration to get the 50% of customers on DB2 v8 to V10 more quickly
  – The risk averse will wait but IBM is pushing hard
  – Earlier pressure on ISVs
• Performance management improvements
  − DDF, buffer pool enhancements, parallelism enhancements
  − High speed INSERT processing
  − Enhanced statement caching
  − Plan stability for dynamic SQL
  − Native utility performance improvements
    • More utility workload to the zIIP (RUNSTATS for example)

• Availability enhancements
  − Major change to online schema changes (much more later)
  − Dynamically add active logs
  − Increased flexibility for reorganizing partitions in parallel
  − New performance features to reduce need for reorganizations
  − Elimination of UTSERIAL lock on SYSUTIL or SYSUTILX tables
DB2 10 Highlights Continued

• Catalog restructure
  − Catalog/directory migrated to UTS (DB2 managed/SMS controlled)
  − SPT01 can extend beyond 64G
  − Row level locking (Expected to eliminate locking problems)

• New options for data storage
  − Temporal tables (business and system)
  − Hash tables (IMS on DB2)
  − Inline LOBs

• Security Enhancements
  − Separation of security from operational administration
    • SECADM/SQLADM roles and other
    • Row and column access level control
DB2 10 Impact on CA Products
Implementing Schema Changes (Some History)

• DB2 V8 introduces ability to change information about DB2 on the fly
  – Change column data types and ability to use table-controlled partitioning
    • A good start but anything physical still required UDCL process

• DB2 9 added additional functionality (much less significant)
  – Rename column/index

• DB2 10 introduces the deferred ALTER
  – Major change to schema management process
DB2 10 Deferred ALTER
How’s it Work?

- ALTER statement validated at execution
- Statement on pending list
  - SYSIBM.SYSPENDINGDDL
- Tablespace placed in Advisory REORG pending (AREOR – new)
- SQL against object returns SQLCODE +610
- Change implemented by utility processes
  - REORG TABLESPACE or REORG INDEX with SHRLEVEL CHANGE or REFERENCE
  - LOAD REPLACE
  - REBUILD INDEX with SHRLEVEL CHANGE or REFERENCE
- ALTER ….. DROP PENDING CHANGES supported
DB2 10 Deferred ALTER
What can be Changed?

SYSIBM.SYSPENDINGDDL

ALTER TABLESPACE

Object placed in AREOR

Implemented at REORG

- Page size of a table space
- Data set size of a table space
- Segment size of a table space
- Table space type, as follows:
  - Table space with only one table to a partition-by-growth universal table space
  - Segmented table space with only one table to a partition-by-growth universal table space
  - Classic partitioned table space to a partition-by-growth universal table space
  - Classic partitioned table space to a range-partitioned universal table space
- Altering the MEMBER CLUSTER structure for a table space
DB2 10 Immediate ALTERs

DB2 10 supports immediate alterations as well

- Changes to LOB inline length
- Versioning
- ACCESS CONTROL
- MASK & PERMISSION
- TRIGGER and FUNCTION SECURED
- MAXPARTITIONS
- INDEX include columns
Restrictions on Alterations

• Can’t recover to point in time prior to changes being materialized
• Can’t mix pending and immediate changes in a single ALTER
• Immediate ALTERs may not be possible if pending changes in place
• Only supported for single table tablespaces
• Only supported for UTS unless converting type
DB2 10 New Storage Options
Temporal Data

Temporal – of or pertaining to time

• New way of managing DB2 historical data
  – Embed support for data versioning in DB2 engine
  – Introduce the concept of time periods in DB2 tables

• System Time Temporal Tables
  – Positioned as an audit/compliance feature

• Business Time Temporal Tables
  – Useful for tracking business events over time

• Bitemporal tables can combine both System and Business Time

• SQL extensions to access and update data
DB2 10 New Storage Options
System_Time Temporal Data

- **System Time Temporal Tables**
  - Implemented with two tables – main and history
  - Data is versioned after insert, update, and delete operations

- **Sequence of events**
  1. CREATE TABLE with SYSTEM_TIME CLAUSE is issued
     ```sql
     CREATE TABLE policy_info
     (policy_id CHAR(10) NOT NULL,
     coverage INT NOT NULL,
     sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
     sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
     create_id TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID,
     PERIOD SYSTEM_TIME(sys_start,sys_end));
     ```
  2. CREATE TABLE statement for history table to receive old rows from main table
     ```sql
     CREATE TABLE hist_policy_info
     (policy_id CHAR(10) NOT NULL,
     coverage INT NOT NULL,
     sys_start TIMESTAMP(12) NOT NULL,
     sys_end TIMESTAMP(12) NOT NULL,
     create_id TIMESTAMP(12));
     ```
  3. ALTER TABLE ADD VERSIONING statement with USE HISTORY TABLE clause
     ```sql
     ALTER TABLE policy_info
     ADD VERSIONING USE HISTORY TABLE hist_policy_info;
     ```

Timestamp columns used to manage define time intervals
- `sys_start` indicates timestamp when row was inserted
- `sys_end` indicates timestamp when row was updated/deleted and copied into history table

History table must exactly match main table definition
### DB2 10 System Period Temporal Tables

**How’s it Work?**

#### policy_info table

<table>
<thead>
<tr>
<th>policy_id</th>
<th>coverage</th>
<th>sys_start</th>
<th>sys_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARLTX973 94</td>
<td>500000</td>
<td>2009-03-15-08.30.00.000000</td>
<td>9999-12-31-24.00.00.000000</td>
</tr>
<tr>
<td>ARLTX973 94</td>
<td>700000</td>
<td>2009-11-01-11.15.00.000000</td>
<td>9999-12-31-24.00.00.000000</td>
</tr>
</tbody>
</table>

#### Policy_info table

<table>
<thead>
<tr>
<th>Policy_id</th>
<th>coverage</th>
<th>Sys_start</th>
<th>Sys_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARLTX973 94</td>
<td>500000</td>
<td>2009-03-15-08.30.00.000000</td>
<td>2009-11-01-11.15.00.000000</td>
</tr>
<tr>
<td>ARLTX973 94</td>
<td>700000</td>
<td>2009-11-01-11.15.00.000000</td>
<td>2010-05-21-16.30.00.000000</td>
</tr>
</tbody>
</table>

1. Row initially inserted March 15th of 2009 (SQL INSERT)
2. Coverage updated Nov 1st of 2009 (SQL UPDATE – original row migrated)
3. Policy updated May 21st 2010 (SQL UPDATE – first update migrated to history)
### DB2 10 System Period Temporal Tables

#### Selecting the Data

**policy_info table**

<table>
<thead>
<tr>
<th>policy_id</th>
<th>coverage</th>
<th>sys_start</th>
<th>sys_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARLTX97394</td>
<td>650000</td>
<td>2010-05-21-16.30.00.000000</td>
<td>9999-12-31-24.00.00.000000</td>
</tr>
</tbody>
</table>

**Policy_id**

<table>
<thead>
<tr>
<th>policy_id</th>
<th>coverage</th>
<th>Sys_start</th>
<th>Sys_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARLTX97394</td>
<td>500000</td>
<td>2009-03-15-08.30.00.000000</td>
<td>2009-11-01-11.15.00.000000</td>
</tr>
</tbody>
</table>

- SELECT coverage FROM policy_info WHERE policy_id = 'ARLTX97394' -- Returns instance 3 from main = 650000
- SELECT coverage FROM policy_info FOR SYSTEM_TIME AS OF '2009-05-05-16.45.00.000000' WHERE policy_id = 'ARLTX97394' -- Returns instance 1 from history = 500000
- SELECT coverage FROM policy_info FOR SYSTEM_TIME AS OF '2010-01-01-08.00.00.000000' WHERE policy_id = 'ARLTX97394' -- Returns instance 2 from history = 700000
System Time Temporal Tables
Operational consideration

• No alterations of schema (data type, add column) on either table
  – Can be removed via DROP VERSIONING

• Can’t drop history tablespace or table
  – Automatically done by dropping Main

• No clones

• Single table tablespaces

• No renames on column or tables

• Point-in-time recovery as a set
  – ENFORCE NO will override

• No utilities that delete data from system time table

• Can’t use DML against the history table that specifies the application period
DB2 10 New Storage Options
Business_Time Temporal Data

- Business Time Temporal Tables (Also called application period tables)
  - Implemented in a single table
  - Data is versioned after insert, update, and delete operations
  - Identifies a period of time when a row is valid

- Sequence of events
  1. CREATE TABLE with BUSINESS_TIME clause is issued:
     ```sql
     CREATE TABLE policy_info
     (policy_id CHAR(4) NOT NULL,
     coverage INT NOT NULL,
     bus_start DATE NOT NULL,
     bus_end DATE NOT NULL,
     PERIOD BUSINESS_TIME(bus_start, bus_end));
     ```
     - Timestamp column pairs used to manage define time intervals
     - `bus_start` indicates timestamp when row was inserted
     - `bus_end` indicates timestamp when row was updated/deleted and copied into history table
     - Column must be DATE or `TIMESTAMP (6) WITHOUT TIME ZONE`
     - Different from system time
     - `WITH DEFAULT` can be used
     - System check constraint is generated to make sure start date is less than end date

  2. Create index on metadata table:
     ```sql
     CREATE UNIQUE INDEX ix_policy
     ON policy_info (policy_id, BUSINESS_TIME WITHOUT OVERLAPS);
     ```
     - Index must be unique
     - Adds business_time end and start columns to the index in ascending sequence
## DB2 10 Business Period Temporal Tables

### How’s it Work?

#### policy_info table

<table>
<thead>
<tr>
<th>policy_id</th>
<th>covera</th>
<th>type</th>
<th>bus_start</th>
<th>bus_end</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>2009-03-15</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2009-03-15</td>
<td></td>
</tr>
</tbody>
</table>

1. Row initially inserted March 15th of 2009 (SQL INSERT)

2. A new row for the same policy_id is inserted; then there are two
   3. Policy coverage updated for a specific period

4. An existing row is updated and a new row inserted

---

```sql
1. INSERT INTO policy_info VALUES ('ARLTX97394', 500000,'PWL','2009-03-15','2010-06-15')
2. INSERT INTO policy_info VALUES ('ARLTX97394', 700000,'PWL','2010-06-15','2010-10-01')
3. UPDATE policy_info FOR PORTION OF BUSINESS TIME FROM '2010-09-01' TO '2010-12-31'
   SET coverage = 600000 WHERE policy_id = 'ARLTX97394'
```
DB2 10 Business Period Temporal Tables
Selecting the Data

policy_info table

<table>
<thead>
<tr>
<th>policy_id</th>
<th>coverage</th>
<th>type</th>
<th>bus_start</th>
<th>bus_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARLTX97394</td>
<td>500000</td>
<td>PWL</td>
<td>2009-03-15</td>
<td>2009-06-15</td>
</tr>
<tr>
<td>ARLTX97394</td>
<td>700000</td>
<td>PWL</td>
<td>2009-06-15</td>
<td>2010-09-01</td>
</tr>
<tr>
<td>ARLTX97394</td>
<td>600000</td>
<td>TRM</td>
<td>2010-09-01</td>
<td>2010-12-31</td>
</tr>
</tbody>
</table>

-- Returns instance 1 from table; coverage = 500000

SELECT coverage FROM policy_info FOR BUSINESS_TIME BETWEEN ‘2009-08-05’ and ‘2009-08-31’ WHERE policy_id = ‘ARLTX97394’

-- Returns instance 2; coverage = 700000

SELECT coverage FROM policy_info FOR BUSINESS_TIME FROM ‘2010-09-15’ TO ‘2010-10-15’ WHERE policy_id = ‘ARLTX97394’

-- Returns instance 3; coverage = 600000

BETWEEN is really ≥ bus_start and < bus_end!!!
DB2 10 New Storage Options

Business_Time Restrictions

- ALTER INDEX does not support the ADD BUSINESS_TIME WITHOUT OVERLAPS option
- No SELECT FROM DELETE or SELECT FROM UPDATE or DELETE with FOR PORTION OF specified
- SQLCODE -104, SQLSTATE 20522 generated in response to these errors
DB2 10 New Storage Options

Bitemporal Tables

- Combines system Time and application features in a single Table
  - Implemented in a single table
  - Data is versioned after insert, update, and delete operations

- Sequence of events

1. Create main table with SYSTEM_TIME and BUSINESS_TIME clauses

2. Create history table to support System Time

3. Alter main table to implement System Time

4. Create unique index to support Business Time (if required)
DB2 10 New Storage Options
Hash Tables

- IMS comes to DB2
- How’s it Work

SELECT coverage FROM policy_info
WHERE policy_id = ‘ARLTX97394’

Hash Calculation

Factors
- Fixed hash data pages
- Anchors (IDs) per data page

Compared with traditional DB2 access using an index
- One GETPAGE for each index level page
- GETPAGE for data page
- I/Os for pages not in BP

DATA PAGES

Data retrieved with a single page access
—New CREATE TABLE Organization Clause

CREATE TABLE BIGEMP
(EMPKEY DECIMAL (12,0) NOT NULL,
FNAME VARCHAR(25) NOT NULL,
LNAME VARCHAR(25) NOT NULL,
DEPT CHAR(3),
...,
PRIMARY KEY (EMPKEY))
IN EMPDB.EMPTS01
ORGANIZE BY HASH UNIQUE EMPKEY
HASH SPACE 64 M

CREATE TABLE BIGEMP_PBR
(EMPKEY DECIMAL(12,0) NOT NULL,
...,
PARTITION BY RANGE
PARTITION 1 ..... HASH SPACE 1G
...,
ORGANIZE BY HASH UNIQUE EMPKEY
HASH SPACE 2G

Partition by Growth

Partition by Range

• Requires UTS implementation
  • PBG or PBR
• Hash key must be UNIQUE
• Hash key not updateable
  • DELETE/INSERT Required
• Hashed space will be larger than current table size
  • As much as 100%
DB2 10 New Storage Options
Hash Tables – Physical Organization

— Organization in PBG and PBR tablespaces

Partition by Growth

PART 1
PART 2
PART 3
PART 4
PART 5
PART 6
PART 7
PART 8

FIXED HASH AREA

OVERFLOW AREA

Partition by Range - UTS

Hash Overflow Index

PART 1

FIXED HASH AREA

OVERFLOW AREA

PART 2

PART 3
DB2 10 Hash Tables
Utility Considerations

• **REORG**
  - Requirements no higher than non-hash table spaces
  - AUTOESTSPACE recommended to calculate hash space using RTS
  - Use to adjust fixed hash space and deal with overflow rows

• **LOAD**
  - Very slow performance because of random insertions
  - Consider sorting input data in hash sequence
    • IBM will provide details of hash algorithm

• **CHECK DATA and INDEX**
  - Verifies hash chains and overflow indexes are correct and consistent

• **RECOVER** except no PIT recovery to point before hash
• **REBUILD INDEX** for hash overflow area for rows
DB2 10 Security
DB2 9 Security (and earlier) Review

Authority: Installation SYSADM
EXECUTE privilege on all plans;
All privileges on all packages;
USAGE privilege on distinct types
and sequences
DEBUGSESSION privileges

Authority: SYSCtrl
System privileges:
BINDADD  CREATEDBC
BINDAGENT  CREATEDS
BDS  CREATEDMB
CREATEALIAS  MONITOR1
CREATEDBA  MONITOR2
STOSPACE

Privileges on all tables:
ALTER  INDEX
REFERENCES  TRIGGER

Privileges on catalog tables:
SELECT  UPDATE
INSERT  DELETE

Privileges on all plans:
BIND

Privileges on all packages:
BIND  COPY

Privileges on all collections:
CREATE IN

Privileges on all schemas:
CREATE IN  DROPIN
ALTERIN

Use privileges on:
BUFFERPOOL  TABLESPACE
STOGROUP

Authority: Installation SYSOPR
Privileges:
ARCHIVE  STARTDB
(Cannot change access mode)

Authority: SYSCtrl
Privileges:
DISPLAY  STOPALL
RECOVER  TRACE

Privileges on routines:
START  DISPLAY
STOP

Authority: SYSADM
EXECUTE privilege on all plans;
All privileges on all packages;
USAGE privilege on distinct types
and sequences
DEBUGSESSION privileges

Authority: PACKADM
Privileges on a collection:
CREATE IN

Privileges on all packages in the collection:
BIND  EXECUTE  COPY

Authority: DBADM
Privileges on tables in the database:
ALTER  INDEX
DELETE  UPDATE
INSERT  REFERENCES  TRIGGER

Authority: DBCTRL
Privileges on one database:
DROP  RECOVER  REORG
LOAD  REPAIR

Authority: DBMAINT
Privileges on one database:
CREATETAB  STARTDB
CREATETS  STATS
DISPLAYDB  STOPDB
IMAGECOPY

* For the applicable catalog tables and the operations that can be performed on them by SYSCtrl, see the DB2 catalog appendix in DB2 SQL Reference.
New system-level authorities

- **SECADM** - Manages security-related objects in DB2 and controls access to all database resources
  - No inherent privilege to access user data
- **ACCESSCTRL** – Allows granting and revoking of explicit privileges from AUTHIDs or roles
  - Can’t revoke DBADM, DATAACCESS, and ACCESSCTRL authorities
- **DATAACCESS** – Allows access to data in tables, views, as well as execution of plans, packages, functions and procedures
- **System DBADM** - DB2 subsystem-wide authority on all objects except security
  - GRANT statement – GRAND DBADM ON SYSTEM TO STAST09
  - By default has ACCESSCTRL and DATAACCESS authorities
  - Authority limited using WITHOUT ACCESSCTRL or WITHOUT DATAACCESS clauses
- **SQLADM** – Allows execution of SQL EXPLAIN statements, RUNSTATS, and MODIFY STATISTICS on user databases without access to underlying data
**DB2 10**

**Additional Security Enhancements**

- **EXPLAIN privilege** – Validate SQL before movement into production without access to user data
  
  - EXPLAIN, PREPARE, DESCRIBE without privilege to execute statements

- **Row and Column Access Control**
  
  - **Row Permissions** – Database object that qualifies access to a specific row of a table
    
    - Implemented in the form of an SQL search condition
  
  - **Column Mask** – Database object that qualifies access to a specific column of a table
    
    - Specifies conditions under which a user (or group or role) can receive masked values returned for a column
    
    - Implemented using SQL CASE expression

- **Audit Policies** – Set of criteria that determines categories to be audited
  
  - Enabled via row in new catalog table (SYSIBM.SYSAUDITPOLICIES)
  
  - Activated and deactivate using new AUDTPLCY keyword of START/STOP TRACE commands

  - 8 categories of audits include object access, administrative authority usage etc
CREATE PERMISSION TELLER ROW ACCESS ON CUSTOMER
FOR ROWS WHERE VERIFY_GROUP_FOR_USER(SESSION_USER, 'TELLER') = 1
AND
BRANCH = (SELECT HOME_BRANCH FROM INTERNAL_INFO
WHERE EMP_ID = SESSION_USER)
ENFORCED FOR ALL ACCESS
ENABLE;

CREATE PERMISSION CSR ROW ACCESS ON CUSTOMER
FOR ROWS WHERE VERIFY_GROUP_FOR_USER(SESSION_USER, 'CSR') = 1
ENFORCED FOR ALL ACCESS
ENABLE;

ALTER TABLE CUSTOMER
ACTIVATE ROW ACCESS CONTROL;

SELECT * FROM CUSTOMER
WHERE VERIFY_GROUP_FOR_USER(SESSION_USER, 'TELLER') = 1
AND
BRANCH = (SELECT HOME_BRANCH FROM INTERNAL_INFO
WHERE EMP_ID = SESSION_USER)

ROW PERMISSION stored in SYSIBM.SYSCONTROLS catalog table
Verifies the user has 'TELLER' as a secondary AUTHID
Constrains the user to accounts from their HOME_BRANCH only
Similar to previous row permission except for 'CSR' and no check on BRANCH
Row permissions must be activated and can be deactivated as well

How's it Work
• SQL search condition is ANDed to the original SQL statement
• Can be used to limit SYSADM access to data
DB2 10 Security Column Mask Example

CREATE MASK SSN_MASK ON EMPLOYEE FOR COLUMN SSN RETURN CASE
    WHEN (VERIFY_GROUP_FOR_USER(SSESSION_USER, 'PAYROLL') = 1) THEN SSN
    WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER, 'MGR') = 1) THEN 'XXX-XX-' || SUBSTR(SSN,8,4)
    ELSE NULL
END
ENABLE;

COMMIT;

ALTER TABLE EMPLOYEE
ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT SSN FROM EMPLOYEE WHERE EMPNO = 123456;

COLUMN MASK stored in SYSIBM.SYSCONTROLS catalog table
For ‘PAYROLL’ users the entire SSN should be visible
For “MGR” users just the last four of the SSN (XXX-XX-1234)
Column access control must be activated to turn it on.

How’s it Work
• CASE expression is added to the SQL statement at bind or execution time
DB2 10 New Security Features
Implementing New Administrative Authorities

• DB2 10 Goal: Separate administrative authorities to decrease risk and comply with more stringent government and compliance regulations

• Enabled with SEPARATE_SECURITY DSNZPARM (DSNTIPP1)
  — YES – Separates DB2 security administration duties from system administration duties
    ▪ SYSADM can no longer control access, manage security related objects, or grant/revoke authorities or privileges; SYSCTRL can no longer manage roles and grant/revoke privileges
  — NO (default) – Overlaps DB2 security administration duties with system administration duties
    ▪ SYSADM and SYSCTRL continues to function as it does in DB2 9 (and earlier)
DB2 10 Highlights
SQL Enhancements

• Support for SQL table functions
• Native SQL stored procedure support improvements
• Datetime constants
• TIMESTAMP WITH TIME ZONE datatype
  − Difference in hours/minutes between local time and UTC (Coordinated Universal Time); also known as GMT
  − Supports applications dealing with times from different locations
  − Ex: 2011-01-19.11.30.00-6:00 – UTC time value would be 15:30 (+6 hours)
• OLAP aggregation specification
• Improved support for SQL scalar functions
• Extended support for implicit casting
DB2 10 Highlights
XML Enhancements

- XML type modifier associates a set of XML schemas with the XML data type
- XML schema validation
  - New built-in function (SYSIBM.DSN_XMLVALIDATE) when inserting a document
  - Process can run on the zIIP
- Multiple versions of XML documents (UTS only)
  - Primary values include reduced contention and improved storage usage
- Updating part of an XML document (DB2 9 updates the entire document)
- Binary XML format that improves efficiency and performance
- Improved support for XML date and time data
- XML supported in native SQL stored procedures and UDFs
- DEFINE NO support for XML (and LOBs)
DB2 10 Overview

Summary

• A major release
  - Bigger than DB2 9 and smaller than V8
  - Offers a direct migration path from V8

• Sources of information
  - DB2 10 Manuals
  - DB2 10 for z/OS Technical Overview (SG24-7892-00) Redbook
  - IDUG Presentations

• CA is interested in your input on support priorities