

# 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

# DB2 10 Highlights Continued

- 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

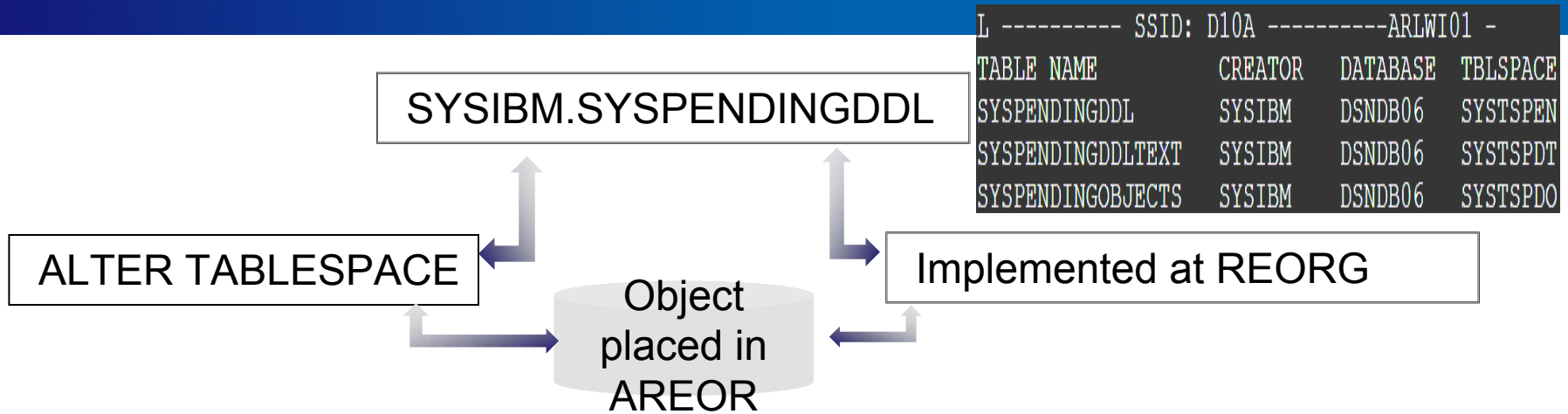
# DB2 10 Impact on CA Products

## 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?



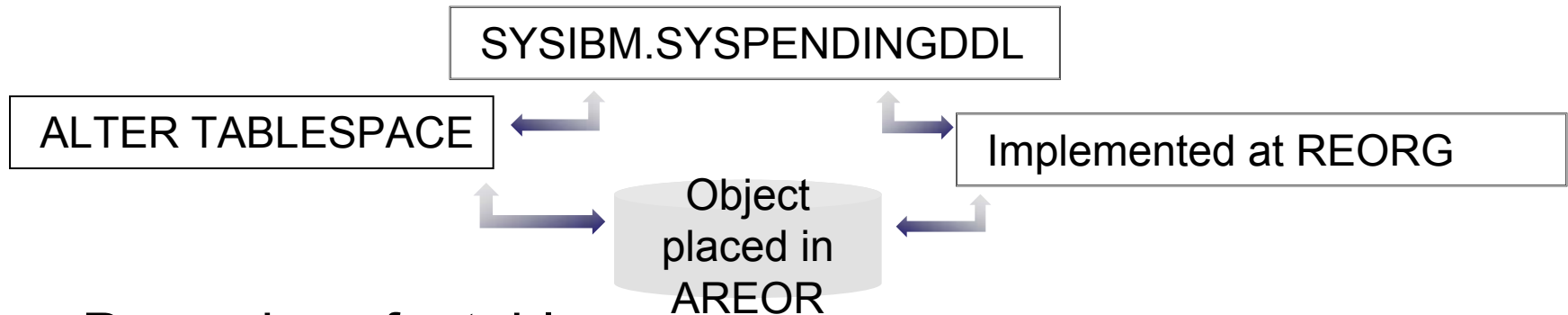
```

L ----- SSID: D10A -----ARLWI01 -
TABLE NAME      CREATOR  DATABASE  TBLSPACE
SYSPENDINGDDL   SYSIBM   DSNDB06   SYSTSPEN
SYSPENDINGDDLTEXT  SYSIBM   DSNDB06   SYSTSPDT
SYSPENDINGOBJECTS  SYSIBM   DSNDB06   SYSTSPDO
  
```

- 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?



- 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

```

L ----- SSID: D10A -----ARLWI01 -
TABLE NAME      CREATOR   DATABASE  TBLSPACE
SYSPENDINGDDL   SYSIBM    DSNDB06   SYSTSPEN
SYSPENDINGDDLTEXT  SYSIBM    DSNDB06   SYSTSPDT
SYSPENDINGOBJECTS  SYSIBM    DSNDB06   SYSTSPDO
    
```

# 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

```
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));
```

Timestamp columns used to manage time intervals

- sys\_start indicates timestamp when row was inserted
- Sys\_end indicates timestamp when row was updated/deleted and copied into history table

2. CREATE TABLE statement for history table to receive old rows from main table

```
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))
```

History table must exactly match main table definition

3. ALTER TABLE ADD VERSIONING statement with USE HISTORY TABLE clause to enable data versioning

```
ALTER TABLE policy_info
ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

# DB2 10 System Period Temporal Tables

## How's it Work?

policy\_info table

- 1
- 2
- 3

policy_id	coverage	sys_start	sys_end
ARLTX97394	500000	2009-03-15-08.30.00.000000	9999-12-31-24.00.00.000000
ARLTX97394	700000	2009-11-01-11.15.00.000000	9999-12-31-24.00.00.000000

- 2
- 3

Policy_id	coverage	Sys_start	Sys_end
ARLTX97394	500000	2009-03-15-08.30.00.000000	2009-11-01-11.15.00.000000
ARLTX97394	700000	2009-11-01-11.15.00.000000	2010-05-21-16.30.00.000000

1. Row initially inserted March 15th of 2009 (SQL INSERT)
2. Coverage updated Nov 1<sup>st</sup> of 2009 (SQL UPDATE – original row migrated)
3. Policy updated May 21<sup>st</sup> 2010 (SQL UPDATE – first update migrated to history)

# DB2 10 System Period Temporal Tables

## Selecting the Data

policy\_info table

policy_id	coverage	sys_start	sys_end
ARLTX97394	650000	2010-05-21-16.30.00.000000	9999-12-31-24.00.00.000000
Policy_id	coverage	Sys_start	Sys_end
ARLTX97394	500000	2009-03-15-08.30.00.000000	2009-11-01-11.15.00.000000
		2010-05-21-16.30.00.000000	

```
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

```
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 table to enforce temporal

```
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

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'

policy_id	coverage	type	bus_start	bus_end
ARLTX973	500000	PWL	2010-03-15	2010-06-15

policy_id	coverage	type	bus_start	bus_end
ARLTX973	700000	PWL	2010-06-15	2010-10-01

policy_id	coverage	type	bus_start	bus_end
ARLTX973	500000	PWL	2009-03-15	2010-06-15

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

2. A new row for the same policy id is inserted; then there were two

3. Policy coverage updated for a specific period

policy_id	coverage	type	bus_start	bus_end
ARLTX973	600000	TRM	2010-09-01	2010-12-31

An existing row is updated and a new row inserted

# DB2 10 Business Period Temporal Tables

## Selecting the Data

policy\_info table

policy_id	coverage	type	bus_start	bus_end
ARLTX97394	500000	PWL	2009-03-15	2009-06-15
ARLTX97394	700000	PWL	2009-06-15	2010-09-01
ARLTX97394	600000	TRM	2010-09-01	2010-12-31

-- 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  $\geq$  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 – Creating a Hash Table

### —New CREATE TABLE Organization Clause

Partition by  
Growth

```
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
```

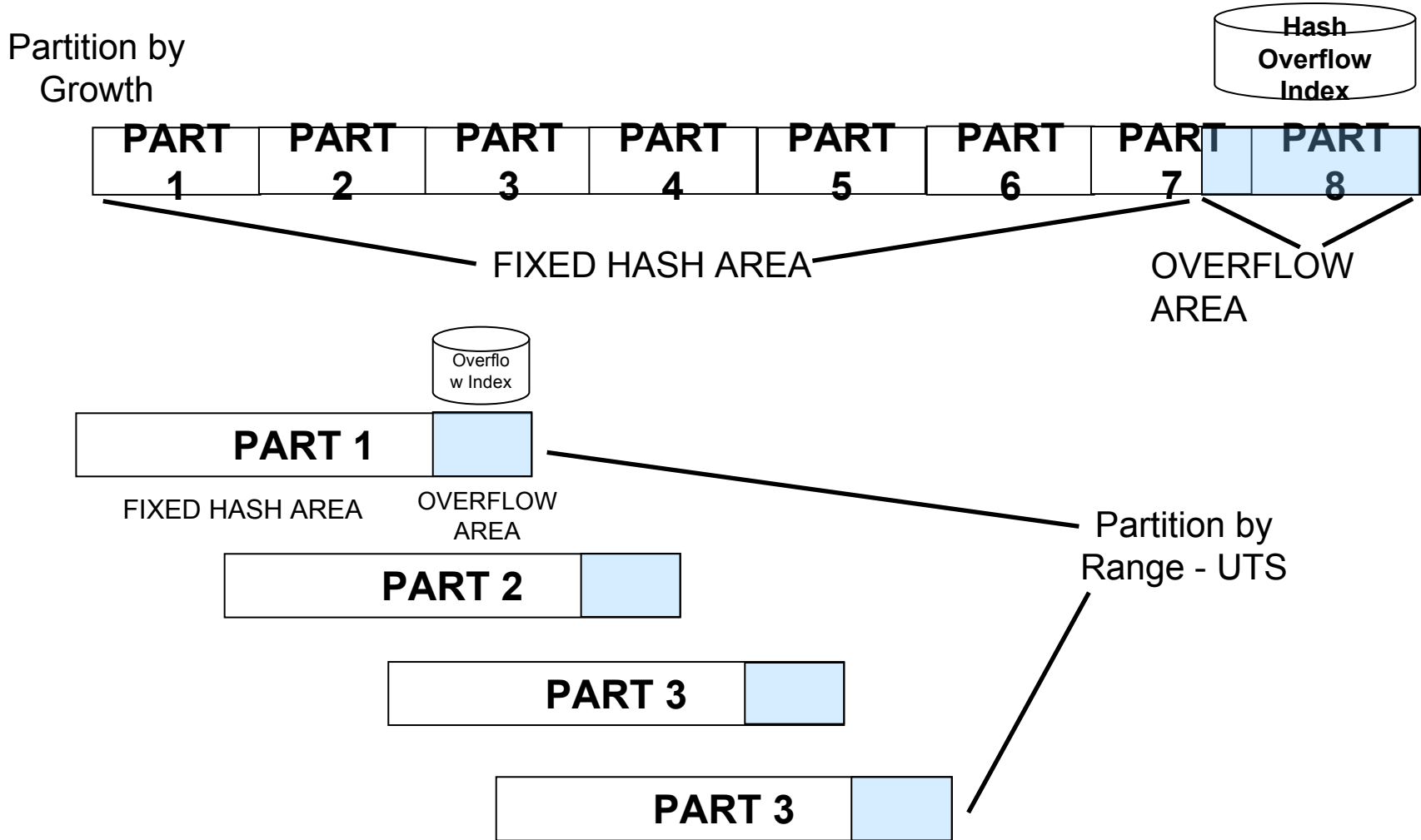
Partition by Range

- 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

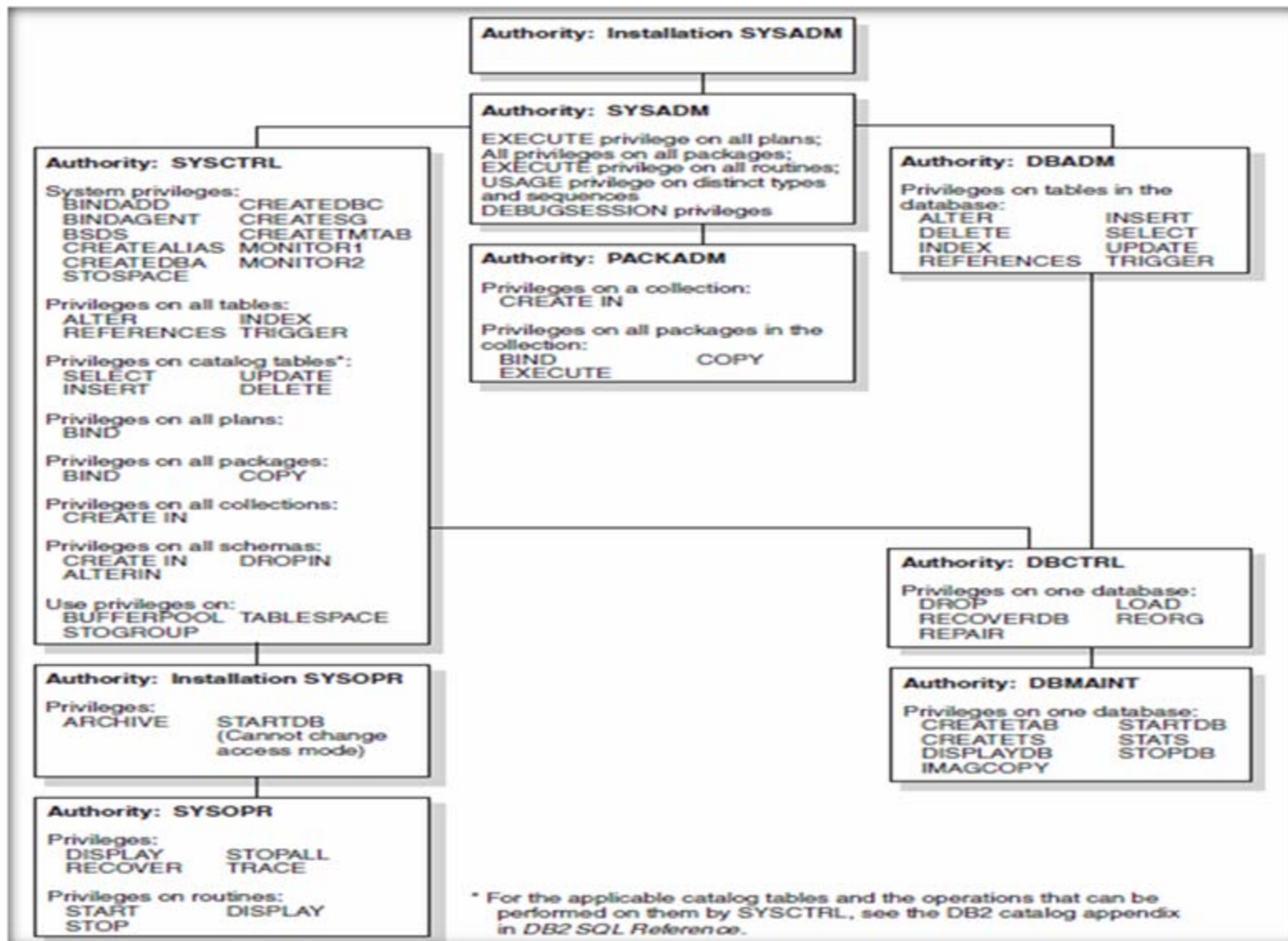


# DB2 10 Hash Tables Utility Considerations

- 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

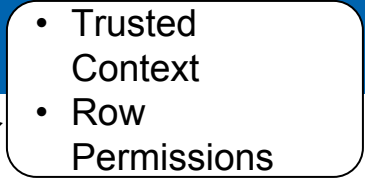
# DB2 10 Security

## DB2 9 Security (and earlier) Review



# DB2 10 Security Enhancements

- 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

- 
- Roles
  - Trusted Context
  - Row Permissions
  - Column Masks

# 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

# DB2 10 Security

## Row Permission Example

ROW PERMISSION stored in  
SYSIBM.SYSCONROLS catalog table

```
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;
```

Verifies the user has 'TELLER'  
as a secondary AUTHID

Constrains the user to accounts  
from their HOME\_BRANCH only

```
COMMIT;
```

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

Similar to previous row permission  
except for 'CSR' and no check on  
BRANCH

```
COMMIT;
```

```
ALTER TABLE CUSTOMER
  ACTIVATE ROW ACCESS CONTROL;
```

Row permissions must be activated  
and can be deactivated as well

```
COMMIT;
```

```
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)
```

### 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.SYSCONROLS 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.

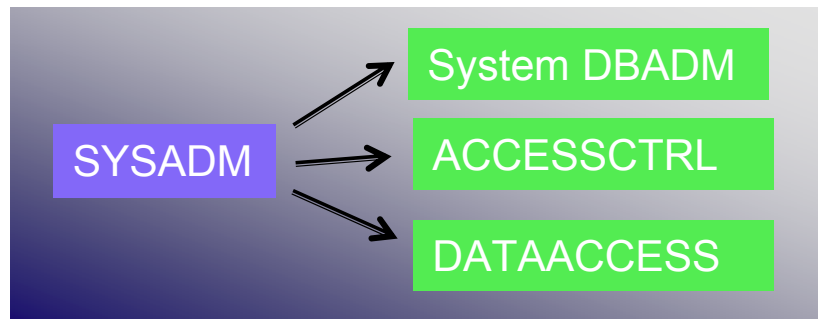
## How's it Work

- 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 DSNZPARAM (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) **Red**book
  - IDUG Presentations
- CA is interested in your input on support priorities