



DB2 10 Customer Experiences
Part 1 – What's New for the Developer
Session Number 1179

Suresh Sane
DST Systems, Inc.

IBM Software
Information On Demand 2011

This presentation provides an early look at some DB2 10 features which will be important to you as the Application Developer or DBA. We will start with the basics and demonstrate how they can be an integral part of a good design strategy. This will allow you to start the planning process early and see if/how they fit in your organization.


Unlike other DB2 10 Overview sessions, we will focus exclusively on those which impact application development.

We were a QPP customer and I will share some early customer experiences which will add value.

IBM Software
Information On Demand 2011
October 23-27
Monday - Friday | Las Vegas, Nevada

Session Outline

- 1. Temporal tables**
- 2. Hash access**
- 3. New SQL features**
- 4. Access path optimization**
- 5. Currently committed data**



1

Temporal tables

- Business driver
- Set up
- Access
- Performance
- Recommendations

Hash access

- Business driver
- Set up
- Access
- Performance
- Recommendations

New SQL features

- Recap of ROW_NUMBER, RANK and DENSE_RANK
- Moving sum
- Moving average
- Generic insert/update (extended indicator variables)
- Implicit casting
- Greater precision for timestamp

Access path optimization

- Literal replacement
- PTC for IN lists
- Scrolling
- Safe query optimization (Dealing with uncertainty, RID List failures)
- Explaining dynamic SQL


Currently committed data

- Business driver
- Access
- Comparison and recommendations

IBM Software
Information On Demand 2011
October 23-27
Marriott Bay 1 Las Vegas, Nevada

About DST Systems

<http://www.dstsystems.com>



- Leading provider of computer software solutions and services, NYSE listed – “DST”
- Revenue \$2.3 billion
- 115 million+ shareowner accounts

- 32,000 MIPS
- 150 TB DASD
- 220,000 workstations
- 752,000 DB2 objects
- Non-mainframe: 600 servers (DB2, Oracle, Sybase) with 3 million objects


2

If you have ever invested in a mutual fund, have had a prescription filled, or are a cable or satellite television subscriber, you may have already had dealings with our company.

DST Systems, Inc. is a publicly traded company (NYSE: DST) with headquarters in Kansas City, MO. Founded in 1969, it employs about 10,000 associates domestically and internationally.

The three operating segments - Financial Services, Output Solutions and Customer Management - are further enhanced by DST's advanced technology and e-commerce solutions.

Where Are We?

- 
- 1. Temporal tables**
 - 2. Hash access**
 - 3. New SQL features**
 - 4. Access path optimization**
 - 5. Currently committed data**



In this section, we will discuss how temporal tables can simplify very complex business logic in use today.

IBM Software
Information On Demand 2011
October 23-27
Marriott Bay 1 Las Vegas, Nevada

Business driver & Overview

- Externalizes error-prone business logic and embed into DBMS
- Temporal Tables with System Time (a.k.a. “versioning”)
 - Useful for auditing and compliance
- Temporal Tables with Business Time
 - Useful for tracking of business events over time
- Bi-temporal tables – those with both Business Time and versioning added
- Special clauses for select and all update processing affected

4

While the name “temporal table” applies to both types of tables, they deal with different business problems – system-time simplifies versioning and consists of two tables while business time consists of only one table with start and end.

Temporal tables with System Time

- Useful for auditing and compliance
- Two tables – main and history
- Each row on the main table has a pair of timestamps **set by DB2** - start time & end time – must be `TIMESTAMP(12)`
- Main table defines a `PERIOD SYSTEM_TIME` with these columns
- History table identical in structure, connected to main table via the `ADD VERSIONING USE HISTORY TABLE` clause

IBM Software
Information On Demand 2011
 October 23-27
 Mandalay Bay | Las Vegas, Nevada

System Time SQL

```

CREATE TABLE EMP
( EMPNO          SMALLINT          NOT NULL
, SALARY         DEC(9,2)         NOT NULL
, REASON         CHAR(10)         NOT NULL
, START_TS       TIMESTAMP(12)    NOT NULL
  GENERATED ALWAYS AS ROW BEGIN
, END_TS         TIMESTAMP(12)    NOT NULL
  GENERATED ALWAYS AS ROW END
, WHEN_CREATED  TIMESTAMP(12)    GENERATED
  ALWAYS AS TRANSACTION START ID
, PERIOD SYSTEM_TIME (START_TS,END_TS))
  
```

```

CREATE TABLE HIST
( EMPNO          SMALLINT          NOT NULL
, SALARY         DEC(9,2)         NOT NULL
, REASON         CHAR(10)         NOT NULL
, START_TS       TIMESTAMP(12)    NOT NULL
, END_TS         TIMESTAMP(12)    NOT NULL
, WHEN_CREATED  TIMESTAMP(12))
  
```

```

ALTER TABLE EMP
  ADD VERSIONING
  USE HISTORY TABLE HIST
  
```

6

Sample DDL for creating the base table and its associated history table. They are linked via the “ADD VERSIONING USE HISTORY TABLE” clause.

The column transaction start id is required (SQLCODE -20490, SQLSTATE 428HM when absent) but null is permissible. ‘with default’ is not permitted. If defined as nullable, the column will contain null values. If defined as NOT NULL, the value will be equal to the value of the ROW_BEGIN column. The value will be a TIMESTAMP(12) value that is unique per transaction and per data sharing member. The last 3 digits of the column value will contain the data sharing member number.

Any system time tables can be defined as implicitly hidden.

ALTER table can convert an existing table to a system temporal table (see Info Center).

System Time example

```
SELECT * FROM EMP  
WHERE EMPNO = 1
```

➔ PROMO 2,000

```
SELECT * FROM EMP  
FOR SYSTEM_TIME AS OF  
'2010-07-01-09.00.00.000000'  
WHERE EMPNO = 1
```


➔ RAISE 1,200

```
SELECT * FROM EMP  
FOR SYSTEM_TIME AS OF  
'2010-10-10-09.00.00.000000'  
WHERE EMPNO = 1
```

➔ PROMO 2,000

The logic to determine "AS OF" condition is now built into the DB2 engine itself.

Syntax which uses SYSTEM_TIME FROM... TO.. or SYSTEM_TIME BETWEEN ... AND ... is also supported.



Oracle Software
Information On Demand 2011
October 23-27
Manchester Bay, Las Vegas, Nevada

Versioning (tables w/ SYSTEM_TIME) restrictions

- No alter of schema (data type, add column, etc.) allowed on both tables - i.e. DROP VERSIONING required!
- Cannot drop the history table or its tablespace (dropping main drops it)
- Cannot define a clone table to either of the tables
- Each table must be the only table in tablespace
- Cannot RENAME column or table
- For point-in-time recovery, both must be recovered as a set (individual recovery disallowed unless overridden with VERIFYSET NO)
- REPORT TABLESPACESET recognizes the set

9


Some restrictions. Most important at this time is probably the tool support.

An attempt to drop the history table results in SQCODE -478 (EMP is dependent on it).

IBM Software
Information On Demand 2011
October 23-27
Marriott Bay 1 Las Vegas, Nevada

Versioning (tables w/ SYSTEM_TIME) restrictions

- No TRUNCATE or utility operation allowed that will delete data from the system period temporal table
 - LOAD REPLACE
 - REORG DISCARD
 - CHECK DATA DELETE YES
- While it is possible to modify (typically purge) rows from history, I suggest that “normal” users **NOT** be granted access to history – they do not need it, since access to the history table is made on behalf of DB2.
- Third-party tool support?



10

Some restrictions. Most important at this time is probably the tool support.

An attempt to drop the history table results in SQCODE -478 (EMP is dependent on it).

Temporal tables with Business Time

- Useful for tracking of business events over time
- Only 1 table (no history table as in System Time)
- Each row has a pair of dates **set by application** – start date and end date (can be future dates) – must be `TIMESTAMP(6) NOT NULL` or `DATE NOT NULL` (not null with default is OK for both)
- The table defines a `PERIOD BUSINESS_TIME` with these columns
- Unique index possible on period to prevent overlaps – while optional, I view this as required! (else possible errors – more than 1 row)

Business Time SQL

```
CREATE TABLE PRICES  
( ITEMNO    SMALLINT    NOT NULL  
, PRICE    DEC(9,2)    NOT NULL  
, REASON   CHAR(10)    NOT NULL  
, START_DT DATE        NOT NULL  
, END_DT   DATE        NOT NULL  
, PERIOD BUSINESS_TIME  
(START_DT,END_DT))  
  
CREATE UNIQUE INDEX PRICESK0  
ON PRICES  
(ITEMNO, BUSINESS_TIME WITHOUT  
OVERLAPS)
```

12

DDL to create a table with business time.

ALTER table can convert an existing table to a system temporal table (see Info Center).

Business Time example

PRICES

ITEMNO	PRICE	REASON	START_DT	END_DT
1	100	REG	2010-04-01	2010-05-31
1	90	SALE	2010-05-31	2010-06-15
1	100	REG	2010-04-01	2010-05-25
1	80	MEM	2010-05-25	2010-05-31
1	80	MEM	2010-05-31	2010-06-05
1	90	SALE	2010-06-05	2010-06-15

➔ INSERT INTO PRICES VALUES (1,100,'REG','2010-04-01','2010-05-31')

➔ INSERT INTO PRICES VALUES (1,90,'SALE','2010-05-31','2010-06-15')

➔ INSERT INTO PRICES VALUES (1,80,'REG','2010-05-25','2010-06-05')

➔ UPDATE PRICES **FOR PORTION OF BUSINESS TIME** FROM '2010-05-25'
 TO '2010-06-05' SET PRICE = 80, REASON = 'MEM' WHERE ITEMNO = 1

➔ **ERROR
-803**

13

An example of how this would work in practice. Notice all the activity which is automatically generated by the update.

Also note that the end-date is NOT included as part of the interval (but start date is).

Data affected by update/delete FOR PORTION OF

	Update	Delete
True superset FOR PORTION OF OLD NEW		
Partial overlap FOR PORTION OF OLD NEW		
True subset FOR PORTION OF OLD NEW		

Examples of what data is affected.

For update, rows in the red section are updated. For delete, the deleted rows are shown as missing.

Business Time example

```
SELECT ...  
FOR BUSINESS_TIME AS OF  
'2010-04-15'  
WHERE ITEMNO = 1
```

→ 100 REG

```
SELECT ...  
FOR BUSINESS_TIME AS OF  
'2010-05-30'  
WHERE ITEMNO = 1
```

→ 80 MEM

Example of how "AS OF" queries are supported. The specified date is such that it is \geq START_DT and $<$ END_DT.

Syntax which uses BUSINESS_TIME FROM... TO.. or BUSINESS_TIME BETWEEN ... AND ... is also supported.

Business time restrictions

- ALTER INDEX does not support ADD BUSINESS_TIME WITHOUT OVERLAPS
- No SELECT FROM DELETE or SELECT FROM UPDATE when UPDATE or DELETE with FOR PORTION OF specified
- SQLERRD(3) does not show rows affected due to temporal update or delete (just like RI or triggers)

Some restrictions.

Bi-temporal SQL

```
CREATE TABLE EMP
( EMPNO SMALLINT          NOT NULL
, SALARYDEC(9,2)         NOT NULL
, REASON      CHAR(10)   NOT NULL
, START_TS    TIMESTAMP(12) NOT NULL
  GENERATED ALWAYS AS ROW BEGIN
, END_TS      TIMESTAMP(12) NOT NULL
  GENERATED ALWAYS AS ROW END
, WHEN_CREATED TIMESTAMP(12) GENERATED
  ALWAYS AS TRANSACTION START ID
, PERIOD SYSTEM_TIME (START_TS,END_TS)
, START_DT DATE          NOT NULL
, END_DT  DATE          NOT NULL
, PERIOD BUSINESS_TIME (START_DT,END_DT))
```

```
CREATE TABLE HIST
( EMPNO SMALLINT          NOT NULL
, SALARYDEC(9,2)         NOT NULL
, REASON      CHAR(10)   NOT NULL
, START_TS    TIMESTAMP(12) NOT NULL
, END_TS      TIMESTAMP(12) NOT NULL
, WHEN_CREATED TIMESTAMP(12)
, START_DT DATE          NOT NULL
, END_DT  DATE          NOT NULL )
```

```
ALTER TABLE EMP
ADD VERSIONING
USE HISTORY TABLE HIST
```

Sample DDL for creating the base table and its associated history table. They are linked via the “ADD VERSIONING USE HISTORY TABLE” clause. In this case, each table also has the business times.

Considerations

- RI – need to version all related code tables?
 - Lack of declarative temporal RI (range vs. value based)
 - Temporal RI is a lot harder (must account for “is contained in” construct) – not any time soon?
- Single table OK but versioning for a group – more complex?
- Bi-temporal may have limited use (my opinion only)
- For `SYSTEM_TIME`, increased storage for the table (depending on update/delete activity) – same as application maintained
- For `BUSINESS_TIME`, understanding the functionality of `UPDATE` and `DELETE FOR PORTION OF`.

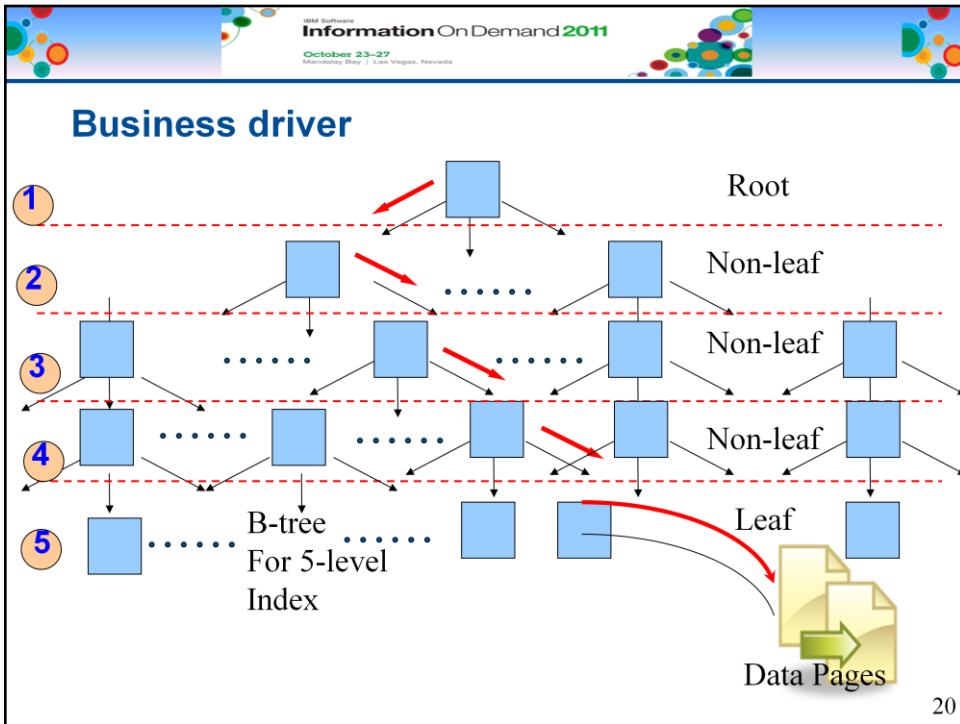
A few considerations and my preliminary recommendations (but I do reserve the right to change my mind as we get more experience!!).

Where Are We?

1. Temporal tables
2. **Hash access**
3. New SQL features
4. Access path optimization
5. Currently committed data



In this section we will discuss how hash access works and when it might be appropriate.



A typical B-tree structure for a large index consisting of 5 levels.

Overview

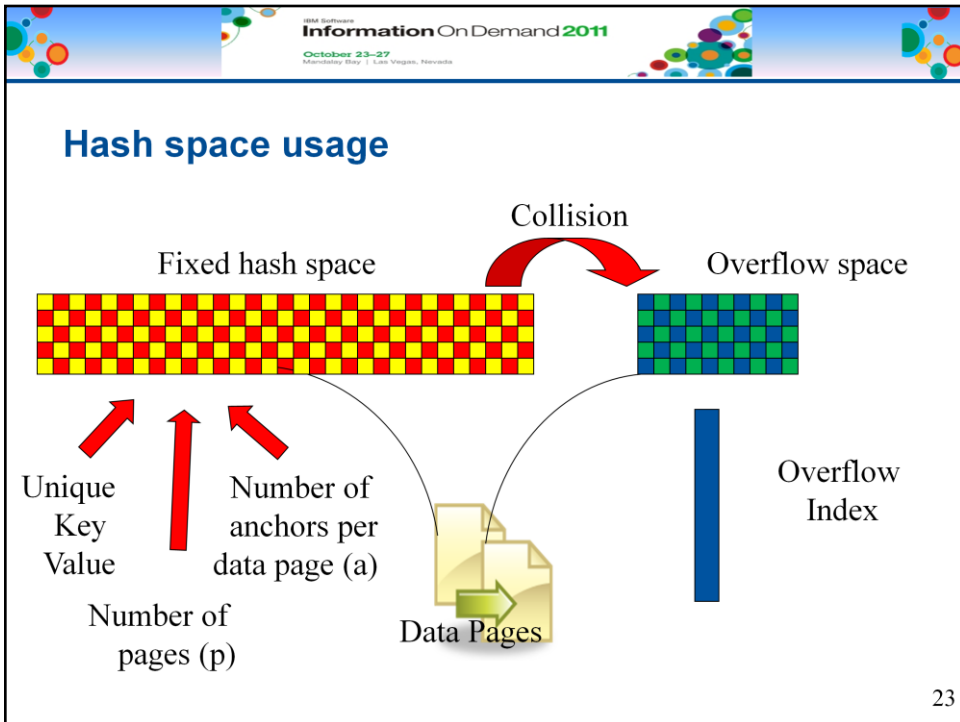
- Good for
 - Tables with a unique key, whose index is many levels (≥ 5)
 - Applications (such as OLTP) needing single row access via the unique key
 - Known approximate table size
 - Equal predicate on all hash key columns (or IN-List)
- Not good for
 - Sequential processing
 - Using range predicates e.g. BETWEEN or $>$ and $<$
- Substantial CPU savings possible (20-35% - even for 3-level index!) – see ref #5
 - May depend on BP size, nlevels, locality of leaf pages etc.

General guideline on when you should consider hash access.

Syntax

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

DDL to create a hash table.



How hashing works and how collisions are handled.

The steps involved are:

1. The key value is scrambled to produce a 64 bit hash value – good for all data types
2. The 64 bit hash value modulo p gives the relative page number - p is a prime number calculated from the size of the table
3. The relative page number is converted to the physical page number in the table space
4. The 64 bit hash value modulo a gives the hash anchor – a is a prime number from 17 to 53 calculated from page size and average number of rows on the page

Monitoring

- Important Real Time Statistics (RTS) values are:
 - SYSTABLESPACESTATS.TOTALROWS - actual number of rows in the table
 - SYSTABLESPACESTATS.DATASIZE - total number of bytes used for rows
 - SYSINDEXSPACESTATS.TOTALENTRIES - number of overflow records with keys in the overflow index
- Ideally, HASH SPACE (in DDL) should be close to DATASIZE (in RTS)
- TOTALENTRIES as a percentage of TOTALROWS should be (ideally zero) less than 10%
- ALTER HASH SPACE and REORG or let DB2 decide during REORG

Key parameters to watch.

Usage notes

- A hash access table must be the only table in the table space (dictated by UTS rules)
- **Hash access cannot support clustering indexes and member cluster**
- Cloned tables cannot be altered to add Hash organization
- LOAD performance may be slower because the data will not be sorted in page order
- Hash key column values cannot be updated - the row must be deleted and re-inserted with the new key value
- If the fixed hash space is too small then performance may suffer

Some considerations and restrictions.

Catalog impact

Table	Column	Description
SYSCOLUMNS	HASHKEYCOLSEQ	Numeric position of column in table's hash key
SYSCOPY	STYPE	H = Hash organization altered
	TTYPE	For reorg, and stype=H, prev. value of HASHDATAPAGES
SYSINDEXES	HASH	Y = index is hash overflow index for the table
SYSINDEXSPACESTATS (RTS)	REORGINDEXACCESS	(# of times index was used) – for hash overflow indexes - # of overflow uses
SYSTABLEPART	HASHSPACE	For PBR, space (KB) override for table level
	HASHDATAPAGES	For PBR, space (pages) for part
SYSTABLES	HASHKEYCOLUMNS	# of columns in hash key
SYSTABLESPACE	ORGANIZATIONTYPE	H=hash
	HASHSPACE	Space (KB)
	HASHDATAPAGES	Space (pages)
SYSTABLESPACESTATS (RTS)	REORGHASHACCESS	# of times hash access was used
	HASHLASTUSED	Date when hash access was last used

The relevant columns in the catalog for hash access.

Where Are We?

1. Temporal tables
2. Hash access
- ➔ 3. **New SQL features**
4. Access path optimization
5. Currently committed data



In this section we will cover some of the new SQL features offered in DB2 10.

IBM Software
Information On Demand 2011
October 23-27
Marriott Marq. | Las Vegas, Nevada

Rownum, rank and dense rank – SQL example

```
SELECT ID, FIRSTNAME, LASTNAME, SALARY,  
RANK() OVER  
(ORDER BY SALARY DESC) AS SAL_RANK  
FROM Y999  
ORDER BY SAL_RANK
```

```
SELECT ID, FIRSTNAME, LASTNAME, SALARY,  
ROW_NUMBER() OVER  
(ORDER BY SALARY DESC) AS SAL_ROWNUM  
FROM Y999  
ORDER BY SAL_ROWNUM
```

**DB2 9 Feature
for completeness**

```
SELECT ID, FIRSTNAME, LASTNAME, SALARY,  
DENSE_RANK() OVER  
(ORDER BY SALARY DESC) AS SAL_DRANK  
FROM Y999  
ORDER BY SAL_DRANK
```

28

This is not new, but covered here for completeness only.

ROW_NUMBER returns the physical row number, RANK() skips a number when duplicates exist (“Olympic”) and DENSE_RANK() does not skip a number when duplicates exist (“Non-Olympic”).

Rownum, rank and dense rank - result

ID	FIRSTNAME	LASTNAME	SALARY	SAL_ RNUMB	SAL_ RANK	SAL_ DRANK
101	TERENCE	REESE	1,000	7	7	4
123	BENITO	GAROZZO	2,000	4	4	3
257	DOROTHY	HAYDEN	3,000	3	3	2
420	BORIS	SCHAPIRO	2,000	5	4	3
654	OMAR	SHARIF	4,000	1	1	1
666	HUGH	KELSEY	2,000	6	4	3
712	KATHY	WEI	4,000	2	1	1

How these 3 features work – an example.

New OLAP functions – “window”

- Partitioning
 - Specified as PARTITION BY clause
 - Similar to grouping via GROUP BY
 - Indicates where to break
- Ordering
 - Specified as ORDER BY clause
- Aggregation group
 - Indication of where to start and where to end
 - Empty result when from-boundary > to-boundary (no error, just like BETWEEN)

Supported aggregation group boundaries

To → From	UNBOUNDED PRECEDING	n PRECEDING	CURRENT ROW	n FOLLOWING	UNBOUNDED FOLLOWING
UNBOUNDED PRECEDING	✗	✓	✓	✓	✓
n PRECEDING	✗	✓	✓	✓	✓
CURRENT ROW	✗	✗	✓	✓	✓
n FOLLOWING	✗	✗	✗	✓	✓
UNBOUNDED FOLLOWING	✗	✗	✗	✗	✗

Reference material only

Moving sum – SQL example

```
SELECT CITY, YEAR, MONTH, RAINFALL,  
SUM(RAINFALL)  
OVER  
(PARTITION BY CITY  
ORDER BY YEAR ASC, MONTH ASC  
ROWS UNBOUNDED PRECEDING)  
AS SUM_RAIN  
FROM RAIN  
ORDER BY CITY, YEAR, MONTH
```

32

We will cover moving sum and moving average only, but other aggregate functions (e.g. CORRELATION, STDDEV etc. can also be used).

Choices include where to start (UNBOUNDED PRECEDING, n PRECEDING or CURRENT ROW) and where to stop (UNBOUNDED FOLLOWING or n FOLLOWING).

Moving sum - result

CITY	YEAR	MONTH	RAINFALL	SUM_RAIN
KANSAS CITY	2010	1	2	2
KANSAS CITY	2010	2	6	8
KANSAS CITY	2010	3	1	9
KANSAS CITY	2010	4	5	14
KANSAS CITY	2010	5	9	23
SEATTLE	2010	1	8	8
SEATTLE	2010	2	10	18
SEATTLE	2010	3	12	30
SEATTLE	2010	4	2	32
SEATTLE	2010	5	7	39

An example of how it works.

Moving average – SQL example

```
SELECT CITY, YEAR, MONTH, RAINFALL,  
AVG(RAINFALL)  
OVER  
(PARTITION BY CITY  
ORDER BY YEAR ASC, MONTH ASC  
ROWS 2 PRECEDING)  
AS AVG_RAIN  
FROM RAIN  
ORDER BY CITY, YEAR, MONTH
```

Choices include where to start (UNBOUNDED PRECEDING, n PRECEDING or CURRENT ROW) and where to stop (UNBOUNDED FOLLOWING or n FOLLOWING).

Moving average - result

CITY	YEAR	MONTH	RAINFALL	AVG_RAIN
KANSAS CITY	2010	1	2	2
KANSAS CITY	2010	2	6	4
KANSAS CITY	2010	3	1	3
KANSAS CITY	2010	4	5	4
KANSAS CITY	2010	5	9	5
SEATTLE	2010	1	8	8
SEATTLE	2010	2	10	9
SEATTLE	2010	3	12	10
SEATTLE	2010	4	2	8
SEATTLE	2010	5	7	7

An example of how it works.

Generic update (extended indicator variables)

Employee id: 12345

First name:	John
Last name:	Smith
Dept:	DBA
Salary:	912.87
Location:	Kansas City ..

Any or all fields may be updated

A typical problem scenario, especially for an update screen.

Business challenges

- For handling a “generic” insert/update (see previous slide), a user has three main choices:
 - Always update all columns – even when not changed
 - “Lazy” programmer’s choice – perhaps most common
 - DB2 resources wasted
 - Create static SQL for each combination
 - Efficient
 - Number of combinations grows exponentially
 - No practical
 - Use dynamic SQL
 - Complex
 - Dynamic SQL cache reuse may be limited (even with parameter markers, if large number of combinations)

Choices which exist today.

Extended NULL indicators to the rescue!

- Enabled as a bind option for static SQL (EXTENDEDINDICATOR(YES)) or by PREPARE for dynamic SQL – **default is NO!**
- -5 means:
 - For insert/merge insert: use the default value for the column
 - For update/merge update: use the default value for the column
- -7 means:
 - For insert/merge insert: use the default value for the column
 - For update/merge update: no-op (column did not change)
- Application logic sets the indicator variable for each host variable appropriately (see next slide)
- One generic insert/update/merge statement can now be used safely

A new choice now available in DB2 10.

Extended NULL indicators - Insert

Application

```
If name-entered
    :ws-name-ind = 0
    :ws-name = new value
Else
    :ws-name-ind = -5
... same for salary...
```

SQL

```
INSERT INTO EMP
VALUES
(:WS-EMPNO
, :WS-NAME:WS-NAME-IND
, :WS-SALARY:WS-SALARY-IND)
```

A coding example of how the indicators will be set by the application and used in an insert statement.

IBM Software
Information On Demand 2011
 October 23-27
 Mandalay Bay, Las Vegas, Nevada

Extended NULL indicators - Update

Application	<pre> If name-changed :ws-name-ind = 0 :ws-name = new value Else :ws-name-ind = -7 ... same for salary... </pre>
SQL	<pre> EXEC SQL UPDATE EMP SET NAME = WS-NAME:WS-NAME-IND, SALARY = :WS-SALARY:WS-SALARY-IND WHERE EMPNO = :WS-EMPNO END-EXEC. </pre>

40

A coding example of how the indicators will be set by the application and used in an update statement.

Note that an attempt to use the CASE statement within the update e.g.

```

CASE WHEN :WS-OLD-NAME = :WS-NEW-NAME
      THEN :WS-NEW-NAME:WS-NULL-7
      ELSE :WS-NEW-NAME-:WS-NULL-0
END

```

does not appear to work (SQLCODE -365, SQLSTATE 22539) – “An expression that involves more than 1 host variable is invalid”.

Implicit casting

- DB2 9 implicitly casts in many cases - e.g. INTEGER to DECIMAL
- DB2 10 extends this to casting between a character or graphic string type and a numeric type
- Allows index access

Implicit casting from numeric to string

Source data type	Target data type
SMALLINT	VARCHAR(6)
INTEGER	VARCHAR(11)
BIGINT	VARCHAR(20)
NUMERIC/DECIMAL	VARCHAR(precision+2)
REAL	VARCHAR(24)
FLOAT	VARCHAR(24)
DOUBLE	VARCHAR(24)
DECFLOAT	VARCHAR(42)

Reference material only

Implicit casting from string to numeric

Source data type	Target data type
CHAR	DECFLOAT(34)
VARCHAR	DECFLOAT(34)
GRAPHIC	DECFLOAT(34)
VARGRAPHIC	DECFLOAT(34)

Reference material only

Casting from string to numeric – DECFLOAT usage

- When DB2 implicitly casts a string to a numeric value, the target data type is DECFLOAT(34), which is then compatible with other numeric data types - it is a superset of the others.
- An invalid string e.g. '12AB34' (embedded blanks) will cause an SQLCODE -420. The message may be confusing since you may be attempting to convert to DECIMAL or INTEGER.
- DECIMAL('1.9', 3, 0) = 1 (truncate always)
- An implicit cast from '1.9' to DECIMAL(3,0) = DECIMAL(DECFLOAT('1.9'), 3, 0) depends on value of CURRENT DECFLOAT ROUNDING MODE
 - ROUND_HALF_EVEN 1.9 → 2 and 1.5 or 2.5 → 2 (default, ties to even)
 - ROUND_HALF_DOWN 1.9 → 1 and 2.5 → 2
 - ROUND_HALF_UP 1.9 → 1 and 2.5 → 3
 - ROUND_DOWN 1.9 → 1 and 2.5 → 2 (recommended)

44

Beware of the possibly confusing message and the fact that ROUND_HALF_EVEN is the default which behaves contrary to what a COBOL programmer would expect.

Greater precision for timestamp

- Number of digits for the fractional second in a timestamp extended
- Range supported in DB2 10 is 0 to 12 digits
 - Maximum of `TIMESTAMP(12)`
 - String representation: `yyyy-mm-dd- hh.mm.ss.nnnnnpppppp`
- Greater precision may be useful when timestamp is used as a unique key
 - Reduces chance of duplicate
 - Still need logic to handle duplicates in application
- The DB2 9 default of 6 digits remains
 - `TIMESTAMP` is the same as `TIMESTAMP(6)`
 - String representation: `yyyy-mm-dd- hh.mm.ss.nnnnnn`

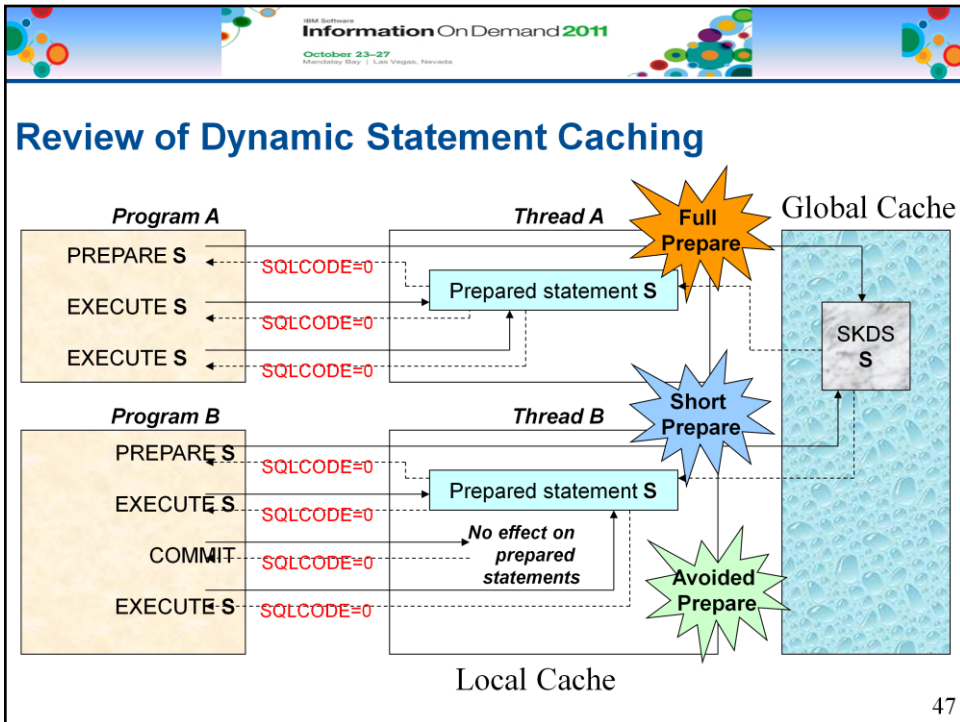
Also available is “timestamp with timezone” but I see limited use for this feature and I will not cover it here.

Where Are We?

1. Temporal tables
2. Hash access
3. New SQL features
4. **Access path optimization**
5. Currently committed data



This section will cover the huge improvements made in access path selection.



Full caching combines the two benefits (ability to issue an EXECUTE without PREPARE – small impact) + (the ability to reuse the statement prepared by another thread – big impact).

Activated by KEEP DYNAMIC(YES) bind parameter and zparm MAXKEEPD > 0 and zparm CACHEDYN=YES.

Literal replacement

- Advantages of dynamic statement cache
 - Avoids Full PREPARE
 - Requires the user id and statement to be identical
- Overcoming the problem with literals
 - WHERE EMPNO = 123 different from WHERE EMPNO = 456
 - Parameter markers preferred e.g. WHERE EMPNO = ?
 - Literals replaced with & (**like ? but different**)
 - Enables cache re-use
- Performance impact
 - Biggest gain for complex SQL (high PREPARE time) with literals which now has a cache hit
 - Beware of NUD/Correlation: REOPT will be needed – else worse performance!

How literal replacement can provide better performance (in most cases).

Literal replacement – How does it work?

- Enablement (either one of..)
 - Put CONCENTRATE STATEMENTS WITH LITERALS in the ATTRSTRING in the PREPARE – (yes, but code change!)
 - Set LITERALREPLACEMENT in the ODBC initialization file
 - Set the keyword enableLiteralReplacement='YES' in the JCC Driver
- Lookup sequence
 - Original SQL with literals is searched in the cache
 - If not found, literals replaced and searched again
 - Must have same attribute – i.e. **? does not match &**
 - If not found, new SQL is prepared and stored in the cache
- No literal replacement if a mixture of literals and parameter markers
- EXPLAIN STATEMENT CACHE ALL populates the LITERAL_REPL column

49

Details on how the replacement can be specified. The choice requiring code change seems the least attractive to me.

Predicate Transitive Closure for IN lists

- Optimizer generates and applies predicates which can be derived from other predicates specified
- Thru DB2 9, possible for equal and range predicates only (=, <>, >, <, <=, >=, BETWEEN, NOT BETWEEN) – now for IN also
- Example:

T1.C1 = T2.C1 AND T1.C1 = :hv1



T2.C1 = :hv1

- Similar closure will now occur for IN list in DB2 10

T1.C1 = T2.C1 AND T1.C1 IN (1,2,3)



T2.C1 = IN (1,2,3)

- More access path choices result due to the closure (e.g. T2 could now be accessed first, unlikely w/o PTC)

50

Simple example of how transitive closure works.

Direct table access for IN lists

- DB2 builds in-memory tables to process multiple IN lists as matching predicates
- New access type of **IN** (instead of N) for direct tables access
- Example: (Table EMP has index EMPK0 by (dept and job))

```
WHERE DEPT IN (?, ?, ?, ?)
AND    JOB  IN (?, ?, ?)
```

- results in:

PL	MT	TAB	AC	INDEX	MC	PF
1	0	DSNxxx..	IN		0	
2	1	DSNxxx..	IN		0	
3	1	EMP	I	EMPK0	2	I

An example of multiple IN lists can now be processed faster.

Scrolling/Re-positioning issues

- Reposition– in normal cursors (not scrollable cursors!)
- Most scrolling predicates provide no filtering at all and easily mislead the optimizer – example:

```
WHERE      (FUND > :WS-FUND  
OR         (FUND = :WS-FUND AND  
           ACCT > :WS-ACCT)  
OR  
           (FUND = :WS-FUND AND  
           ACCT = :WS-ACCT AND  
           DATE > :WS-DATE))  
AND       FUND >= :WS-FUND  
ORDER BY  FUND, ACCT, DATE
```

Index on: FUND
ACCT
DATE



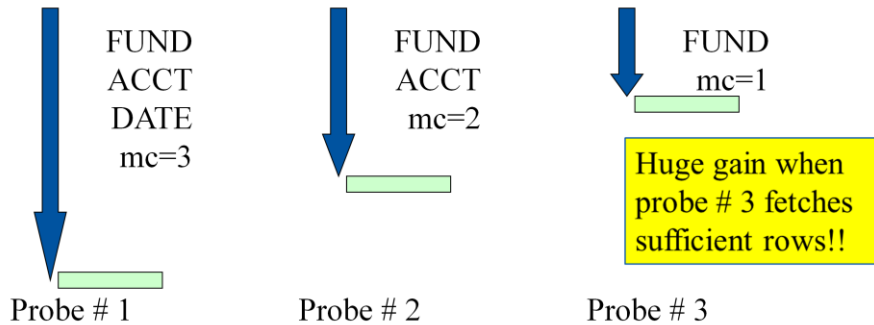
Redundant predicate
Needed in DB2 9
For matching index access

52

A common problem that haunts all restartable cursors! DB2 can match on at most 1 column due to the OR logic.

Scrolling/Re-positioning in DB2 10

- Matching index access possible in DB2 10
- 3 probes – **access type NR**
- QBLOCKNO order not relevant!



53

This feature spells R-E-L-I-E-F !!! I am expecting a huge performance gain for cursors accessing large tables (but returning few rows) in CICS programs which use such restartable cursors.

My original request was for a new SQL clause "SCROLL ON" (just like the ON clause vs. WHERE clause), to create such probes but the end result is identical to what is provided in DB2 10.

Safe query optimization

- Common reasons for bad access paths
 - Host variables or parameter markers with non uniform distribution of data – most common in our environment
 - Missing stats or stats not current
 - Unpredictable runtime resource availability – especially, RID pool usage
- Access path based purely on cost-based optimization needs a “reality check” (just like degree of parallelism)
 - Dealing with reality at run time
 - Dealing with uncertainty at bind time

Features of the “safe optimization” initiative.

Dealing with RID pool failures

- If a RID limit is reached
 - Overflow RIDs to workfile and continue processing
 - Avoids fallback to table space scan as happens DB2 9
- Default RIDPOOL size increased from 8 MB to 400 MB
Zparm MAXTEMPS_RID limits workfile usage for each list
- Work-file usage may increase - need to monitor (but not expected to increase due to larger RIDPOOL size)

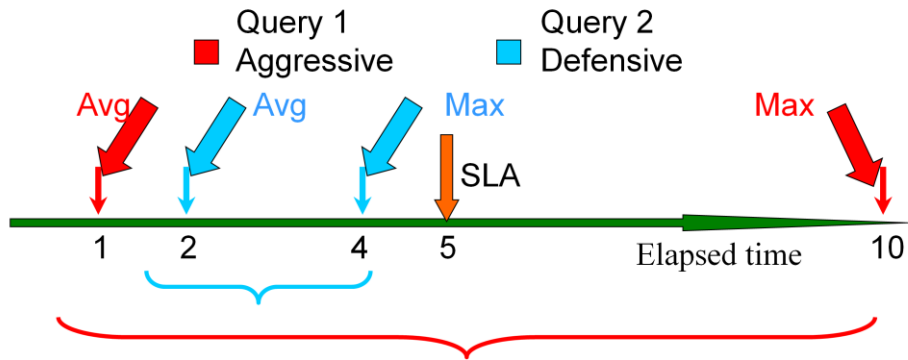
Some implications of what safe optimization means at run time.

Dealing with uncertainty

- Optimizer evaluates the “risk” associated with each predicate
- Compare access paths with close cost and chooses lowest risk plan
- Full details of “uncertainty” are **NOT** externalized!
- A good first step but not there yet (my opinion) – see next slide


Some implications of what safe optimization means at bind time.

Dealing with uncertainty



Safe optimization attempts to cater to the “worst-case” as well as the “average case”. With defensive optimization, an access path with avg = 2 sec and worst case = 4 sec would be preferred over one with avg = 1 sec and worst case = 10 sec

You want minimum cost, but what cost? Average or worst-case?



How uncertainty should be dealt with

- *“..a robust query optimizer is one that generates plans that work reasonably well even when optimizer assumptions fail to hold.”*
 - Yes, DB2 10 attempts to do so.
- *“Because robustness sometimes comes at the cost of performance, users should be allowed to prioritize these competing objectives.”*
 - No, this is hidden within DB2 and user has no choice.


58

This tradeoff between performance and predictability has been discussed in various papers, most notably by Brian Babcock and Surajit Chaudhuri (see ref #3).

Obtaining EXPLAIN information for dynamic SQL

- For host-based languages (e.g. COBOL), modify the source to add the following prior to PREPARE:
 - EXEC SQL SET CURRENT EXPLAIN MODE = YES END-EXEC.
- For JDBC and SQLJ
 - Set currentExplainmode connection property
 - Automatically sets the CURRENT EXPLAIN MODE special register
- For ODBC and CLI
 - For system-wide setting: Set keyword DB2EXPLAIN in DSNAOINI file
 - For specific application: set SQL-ATTR-DB2EXPLAIN using SQLSetConnectAttr() function
- Compatible with DB2 LUW

New options to obtain EXPLAIN information.



CURRENT EXPLAIN MODE Values

- NO
 - Default
 - No Explain performed
- YES
 - Explain performed
 - Prepared statements written to cache
 - Execution permitted
- EXPLAIN
 - Explain performed
 - Prepared statements written to cache
 - Execution prohibited (SQLCODE +217)

60

The choices.

A couple of things to be aware of.

First, the user must have “explain monitored statements” privilege – not typically granted to developers.


Second, if you place an explainable statement and then select from the PLAN_TABLE, DB2 runs (and explains the selects also) – beware of the extra rows!

Where Are We?

1. Temporal tables
2. Hash access
3. New SQL features
4. Access path optimization
- ➔ 5. **Currently committed data**



In this final section, we will present a new option to provide more concurrency. It comes at a price and we will discuss all the “gotcha’s” also.



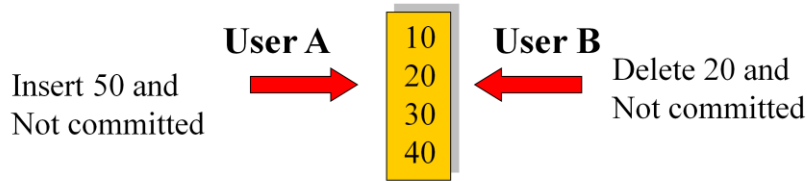
Why and How?

- Isolation UR avoids contention, but does not return committed data
- Applications ported from other DBMSs (e.g. Oracle) particularly prone to timeouts
- Returns currently committed data without waiting for locks
- Supported for uncommitted inserts or deletes
- **No support for uncommitted updates**

62

To me, the primary business driver for this feature appears to be easy migration from other databases to DB2.

Simple example (assumes row-level locking)



User C: SELECT *
 FROM..

Default (wait for outcome):	➡	(10) and wait/timeout on 20
Skip locked data:	➡	10, 30, 40
With UR:	➡	10, 30, 40, 50
Currently committed:	➡	10, 20, 30, 40

Note: UPDATE not supported - causes wait for outcome

A simple example of how each option works.

How does it work?

- Applies only when accessing UTS on DB2 10 NFM
- If contention is with uncommitted **insert**, it applies to Isolation CS or RS only
- If contention is with uncommitted **delete**, it applies to Isolation CS with CURRENTDATA(NO) only
- Statement level overrides package level which overrides plan level which overrides system level
- If lock not available & held by inserter – skip row
- If lock not available & held by deleter – return row

Some considerations and restrictions.

Syntax

- Default is “wait for outcome” behavior (no change)
- New BIND Option
 - `CONCURRENTACCESSRESOLUTION(USECURRENTLYCOMMITTED | WAITFOROUTCOME)`
- New PREPARE Attribute
 - `PREPARE ... USE CURRENTLY COMMITTED | WAIT FOR OUTCOME`
- New bind option in CREATE/ALTER of PROCEDURE, FUNCTION
 - `CONCURRENT ACCESS RESOLUTION U[SE CURRENTLY COMMITTED] / W[AIT FOR OUTCOME]`

Syntax rules.

When does it apply?


- Applies to row and page locking
 - Page locking
 - IRLM tracks up to 8 rows on a page
 - Page lock request for insert/delete specifies row
 - Page lock request for read specifies row
 - If IRLM knows about it then returns to say deleter/inserter
 - If IRLM doesn't know about it, then reader will wait for lock to be released
 - Does not apply to table, partition or table space locks
 - Not applicable when LOCK TABLE IN EXCLUSIVE used
 - Not applicable when lock holder is performing mass delete
 - Not applicable if lock holder has escalated

Cases when it applies and when it does not.

How do I track how often it happens?

- New counter **QISTRCCI** (part of Data Manager Statistics) - (IFCID 002)
 - Shows the number of rows skipped by read transactions using currently committed option which finds uncommitted inserts
- Similarly, new counter **QISTRCCD**
 - Shows the number of rows skipped by read transactions using currently committed option which finds uncommitted deletes

New counter to see how often the feature is activated.



IBM Software
Information On Demand 2011
October 23-27
Marriott Bay 1 Las Vegas, Nevada

Gotcha's

- Currently committed may allow committed data to be returned without waiting
- BUT – does not guarantee that DB2 will do so - in some cases DB2 may revert to unconditional locking (e.g. more than 8 rows locked on a page)
- Updates are NOT supported (where it is needed most!)

68

Some things to watch out for. In my opinion, the most troublesome area is update logic on control tables (e.g. next account number) – this is where DB2 10 does NOT support it...oh well, there will be DB2 11...

Conclusions and key take-aways

- How Temporal tables can simplify your code
- How Hash access can speed up queries against large tables
- How new SQL features can help
- How Access path determination is now smarter
- How concurrency can be improved without sacrificing integrity (not fully yet..)

I trust this session has empowered you with the knowledge to exploit the new application features of DB2 10. Good Luck!

References



1. DB2 10 for z/OS SQL Reference
2. DB2 10 for z/OS Application Programming and SQL Guide
3. "Towards a Robust Query Optimizer: A Principled and Practical Approach" - Brian Babcock and Surajit Chaudhuri - Proceedings of the 2005 ACM SIGMOD international conference on Management of data
4. IBM Redbook – DB2 10 Technical Overview – SG24-7892
5. IBM Redbook – DB2 10 Performance Topics – SG24-7942


Some of the useful references.

IBM Software
Information On Demand 2011
 October 23-27
 Mandalay Bay, Las Vegas, Nevada

About the Instructor

Suresh Sane

- Co-author-IBM Redbooks
 - SG24-6418, May 2002
 - SG24-7083, March 2004
 - SG24-7111, July 2006
 - SG24-7688, January 2009
- Seminars, courses and presentations in USA, Canada, Europe, Australia, Thailand and Israel



Member of IDUG Speaker Hall of Fame (Seattle, 2004; Denver, 2006; Las Vegas, 2011)

- IBM Information Champion (2009, 2010, 2011)

71


Suresh Sane is an IDUG Hall of Fame speaker with three Best User Speaker awards and numerous top 10 finishes. He has lectured worldwide and co-authored 4 IBM Redbooks (Dynamic SQL, Stored Procedures, Data Integrity and DB2 Packages). He was recognized as an IBM Information Champion in 2009 - 2011.

He served on the NA Conference Planning Committee 2004-2008 (Conference Chair for IDUG NA 2008) and on the IDUG Board of Directors 2009-2011.

Contact Information:

sssane@dtsystems.com or sureshsane@hotmail.com

Suresh Sane
 DST Systems, Inc.
 1055 Broadway
 Kansas City, MO 64105 USA
 (816) 435-3803



IBM Software
Information On Demand 2011
October 23-27
McCormick & Co. | Las Vegas, Nevada

Suresh Sane
DST Systems, Inc.
sssane@dstsystems.com or sureshsane@hotmail.com

Thank you and good luck with DB2 10!

Thank You!

Your Feedback is Important to Us

- Access your personal session survey list and complete via SmartSite
 - Your smart phone or web browser at: iodsmartsite.com
 - Any SmartSite kiosk onsite
 - Each completed session survey increases your chance to win an Apple iPod Touch with daily drawing sponsored by Alliance Tech