DB2 Migration Planning Workshop

IBM

IBM Software Group

DB2 for z/OS Temporal data with Archive Transparency and V11 updates

DB2 Information Management Software

Howie Hirsch **hshirsch@us.ibm.com**

nk

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, AS/400, DATABASE 2, DB2, e-business logo, Enterprise Storage Server, ESCON, FICON, OS/390, OS/400, ES/9000, MVS/ESA, 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

Temporal Data – Time Travel Query

- **What is temporal data?**
- **Business Time & System time**
- **What are the benefits of using the database in temporal data**
- **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**

Benefits of using temporal tables …

- **Move the logic from the application layer to the database layer**
  - Consistent handling of temporal data
- **Reduce Application development time by up to 10x**
  - Application development can focus on business functions
- **Run current applications with no code change**
  - For System Time working with the current version of data
- **Preserve execution time for current queries going after current data (System Time)**
- **You probably have these types of applications running in your shop**

Benefits of using temporal tables …

- **Business Problems you can solve with temporal tables**
  - Ensure that a customer only has one financial position at a given time
  - Was an insured covered for a procedure on a specific date?
    - Was that information correct at the time the claim was processed?
  - Establish prices for a catalog ahead of time, so that they are completed before the change needs to be made
  - Answer a customer complaint about an old bill
  - … and many, many more

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
  - PM31314 (9/2011) allows the use of TIMESTAMP WITH TIMEZONE
  - 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
- **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) );
```

SYSTEM TIME columns

BUSINESS TIME columns

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)      NOT NULL,
   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;
```

- The Table structures must be the same
- The Table must be in single-table, Table Spaces
- The Table Spaces do not have to have the same attributes

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**

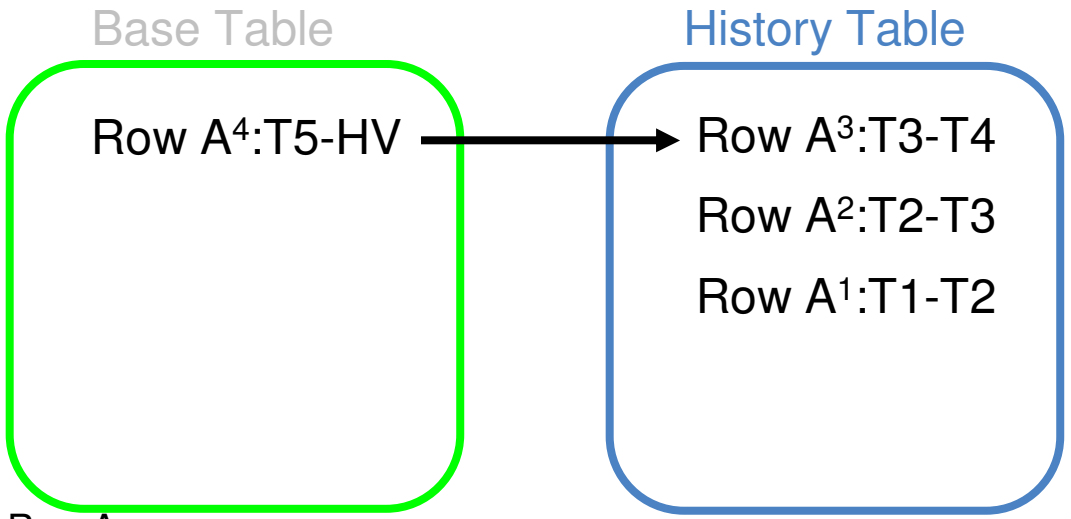
–Insert or Update	–The base table row(s) are created / updated with a UOW start time as System Start Time and a high value System End Time.
–Delete	–Remove the base table row.
–Update or Delete	–Create a “before-image” copy of all qualified base table rows in the History Table. –The newly created History row(s) are added with a System End Time equal to the UOW start time (System Start Time of the associated base table row for an update)

Update & Delete do the bulk of the System Time work.

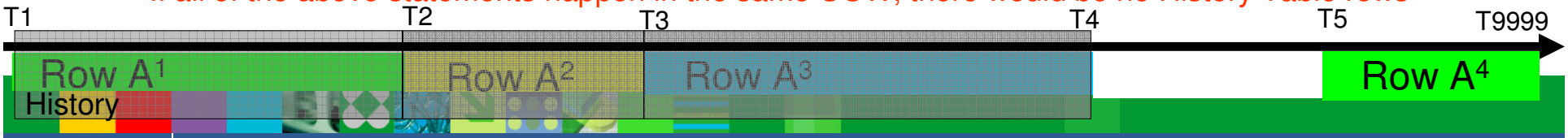
History table rows are a inserted copy of the Base table EXCEPT for the ST End.

Row Maintenance with System Time

- T1: INSERT Row A
- T2: UPDATE Row A
- T3: UPDATE Row A
- T4: DELETE Row A
- T5: INSERT Row A



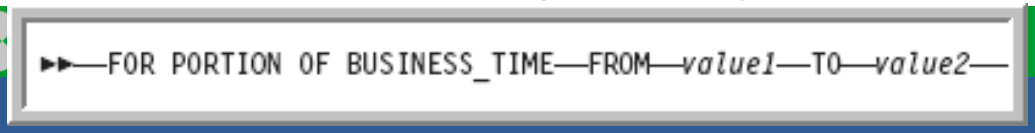
- **Notes:**
  - INSERT has no History Table impact
  - The first UPDATE begins a lineage for Row A.
    - History Table ST End = Base Table ST Begin (No gap)
    - The Base Table ST End is always High Values (HV)
  - The second UPDATE deepens the lineage
    - No gaps exist across all generations of Row A.
  - The DELETE adds to the lineage in the History Table.
    - There is no current row (Base Table) after the DELETE
  - The second INSERT begins a new row lineage
    - There is a gap between the History Table rows and the Base Table
  - If all of the above statements happen in the same UOW, there would be no History Table rows



Row Maintenance with Business Time

- **UPDATE/DELETE 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 Business Time 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



Row Maintenance with Business Time

For each row that qualifies:

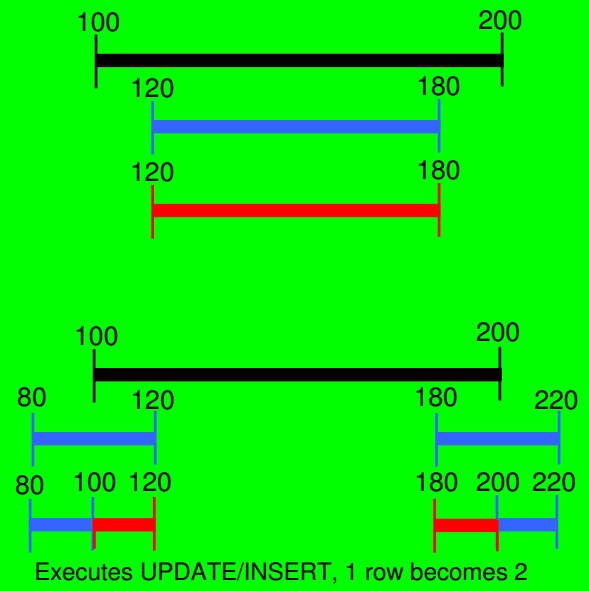
Rows Contained
FOR PORTION OF Business Time Row Result

Span FROM or TO
FOR PORTION OF Business Time Row Result

Span FROM and TO
FOR PORTION OF Business Time Row Result

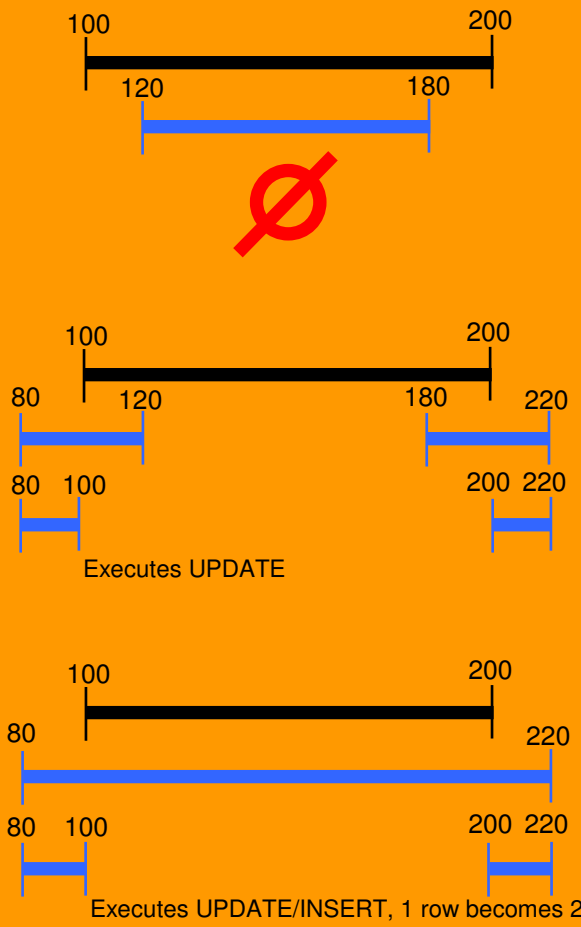
UPDATE

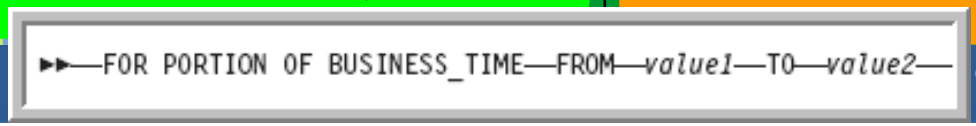
UPDATE <table...>
FOR PORTION OF BUSINESS_TIME FROM <100> TO <200>
SET....
WHERE ....



DELETE

DELETE FROM <table...>
FOR PORTION OF BUSINESS_TIME FROM <100> TO <200>
WHERE ....





AS OF Example

```
SELECT ....
FROM .....
            FOR BUSINESS_TIME AS OF x
WHERE.....
.....
;
```

Inclusive

This is a simplified version of the generated predicates, which can be viewed in the DSN_PREDICAT_TABLE

Inclusive

Exclusive

Business Time Period:
Start: DEPT_BEGIN
End: DEPT_END

Generated Predicates:
WHERE DEPT_BEGIN <= x
AND DEPT_END > x

- Same options available with FOR SYSTEM_TIME for the System Time dimension
- PM29832: "AS OF TIMESTAMP x" alternative for "FOR SYSTEM_TIME AS OF x"

FROM / TO Example

```
SELECT ....
FROM .....
                FOR BUSINESS_TIME FROM x TO y
WHERE.....
.....
;
```

Inclusive

Inclusive

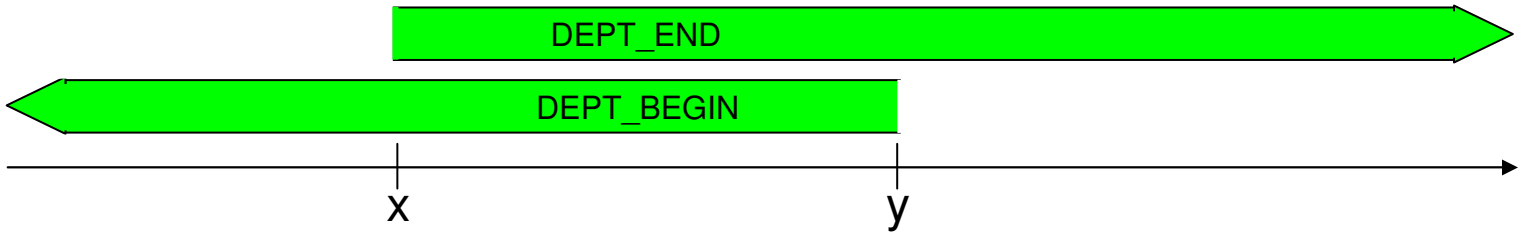
Exclusive

```
Business Time Period:
Start:        DEPT_BEGIN
End:          DEPT_END
```

Exclusive

```
Generated Predicates:
WHERE DEPT_BEGIN < y
    AND DEPT_END     > x
    AND x < y
```

- The temporal qualifiers (x, y):
  - Can use special registers and date/time arithmetic
  - Cannot reference columns / functions of the query



BETWEEN Example

```
SELECT ....
FROM .....
            FOR BUSINESS_TIME BETWEEN x AND y
WHERE.....
.....
;
```

Inclusive

Inclusive

Inclusive

Business Time Period:
Start: DEPT_BEGIN
End: DEPT_END

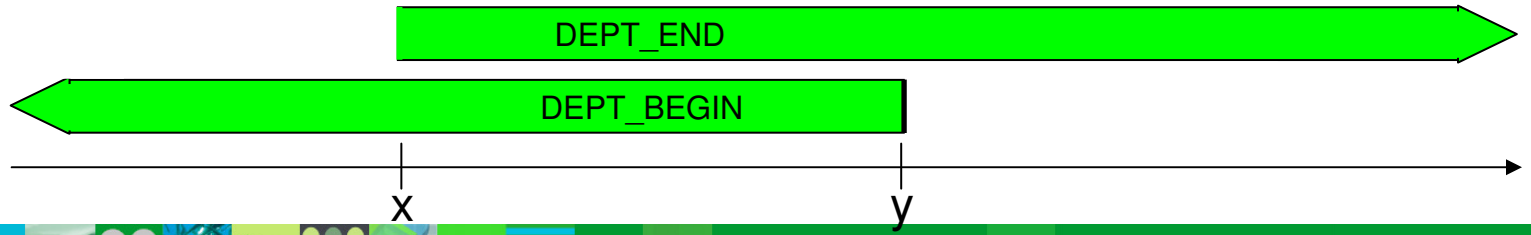
Exclusive

Generated Predicates:

```
WHERE DEPT_BEGIN <= y
    AND DEPT_END    > x
    AND x <= y
```

Be cautious of using a non-temporal BETWEEN. It does not honor the exclusive nature of the End Time column and therefore can return invalid results:

WHERE x BETWEEN DEPT_BEGIN AND DEPT_END



Multiple Temporal References & Temporal Joins

- FOR BUSINESS_TIME **and** FOR SYSTEM_TIME **can be coded in the same table reference**
  - This will cause Business Time predicates to be applied against the associated Business Time columns in the History Table as well
  - Depending on the combination coded, the associated predicate rewrite and **UNION ALL** specification will be included
- **Each table in a join can include a temporal reference**
  - **SELECT ....**

    FROM TMPRL_DEPT FOR BUSINESS_TIME FROM x TO y,

    TMPRL_EMP FOR BUSINESS_TIME AS OF x

    WHERE .....
  - Make sure that the join makes “business sense”
  - Unless **AS OF** is used consistently through the join, it’s possible to join data that is not effective at the same time

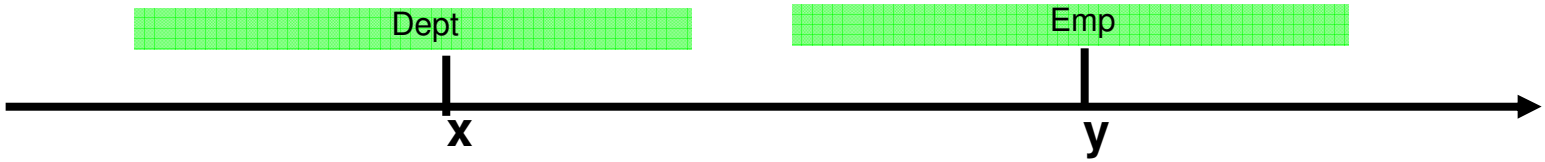
Multiple Temporal References & Temporal Joins

- **Each table in a join can include a temporal reference (continued)**
  - **SELECT ....**

    FROM TMPRL_DEPT FOR BUSINESS_TIME FROM x TO y,

    TMPRL_EMP FOR BUSINESS_TIME FROM x TO y

    WHERE .....
  - This can join rows together that are not effective at the same time



- **Non temporal predicates can be coded to ensure common affectivity**
  - **SELECT ....**

    FROM TMPRL_DEPT FOR BUSINESS_TIME FROM x TO y TD,

    TMPRL_EMP TE

    WHERE TE.EMP_END > TD.DEPT_BEGIN

    AND TE.EMP_BEGIN < TD.DEPT_END
  - This can be used in lieu of the ability to use query columns in the temporal table reference

System Period Versioning Information...

- **The Base and History table reside in single table-table spaces.**
  - Columns with the same names, data types, null attributes, CCSIDs, subtypes, hidden attributes, and field procedures as the corresponding system-period temporal table
- **A ROWID should be the only GENERATED column**
- **PM61811 (2/2013) override of recorded history and generated columns**
  - For replication products
  - Stored Procedure SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY
- **PM31313 allows for ALTER TABLE...ADD COLUMN**
  - The same column is added to the History Table
- **A History Table cannot**
  - be an MQT,
  - have a Clone Table, Column Mask, Row Permission, or Security Label column, a System Period or Application Period.
- **No temporal SELECT, UPDATE, or DELETE against the History**
- **Cannot TRUNCATE**
  - INSERT, UPDATE, DELETE, and MERGE are accepted
- **To find the Base / History Tables**

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

Bi-temporal example …

Step 1 – 9/21/2010 Employee C054 chooses HMO policy with $10 copay effective 1/1/2004

```
INSERT INTO POLICY
 (EMPL,TYPE,PLCY,COPAY,EFF_BEG,EFF_END)
  VALUES ('C054','HMO', 'P667', '$10', '1/1/2004', '12/31/9999');
```

Policy Table

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14.745082721000	9999-12-30-00.00.00.000000000000

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

POLICYHISTORY table is empty at this point

Bi-temporal example

Step 1 – 09/21/2010 Employee C054 chooses HMO policy with $10 copay effective 1/1/2004
Step 2 – 09/24/2010 Update all P667 policies to a copay of $15 beginning 01/01/2011

Original Row

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14.745082721000	9999-12-30-00.00.00.000000000000

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

Policy Table

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22.50672497000	9999-12-30-00.00.00.000000000000
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22.50672497000	9999-12-30-20000.00.000000000000

Policy History table

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14.745082721000	2010-09-24-17.33.22.50672497000

System Time / Point In Time...

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF
    '2010-09-22-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

System Time / Point In Time...

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF
    '2010-09-22-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

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

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY

System Time / Point In Time …

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF
    '2010-09-25-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

System Time / Point In Time

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF
    '2010-09-25-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

As of 09-25-2010 , the only rows that qualify are the rows from the current table, because on 09-24-2010 we updated the rows, and both rows in the current table begin as of 09-24-2010.

The Base results are differentiated from the History results by the SYS_END column. The Base will reflect 9999-12-30-00.00.00...

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00

System Time / Range …

```
SELECT * FROM POLICY FOR SYSTEM_TIME
  FROM '2010-09-22-00.00.00.000000000000'
    TO '2010-09-25-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

QBLOCKNO	PLANNO	METHOD	CREATOR	TBNAME	TABNO	ACCESSTYPE	PREFETCH	QBLOCK_TYPE
1	1	0	DBA015	POLICYHISTORY	2	R	S	NCOSUB
2	1	0			0			UNIONA
5	1	0	DBA015	POLICY	1	R	S	NCOSUB

System Time / Range

```
SELECT * FROM POLICY FOR SYSTEM_TIME
  FROM '2010-09-22-00.00.00.000000000000'
    TO '2010-09-25-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

For this query, we look for rows where:

System Start Time (SYS_BEG) < 9/25 AND

System End Time (SYS_END) > 9/22

Result of query

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

Bi-temporal example …

Step 1 - 09/21/2010 Employee C054 chooses HMO policy with $10 copay effective 1/1/2011
Step 2 - 09/24/2010 Update all policies P667 to a copay of $15 beginning 01/01/2011
Step 3 - 09/24/2010 Later on the same day(19.44) of 09/24, the customer cancelled the policy

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

```
DELETE FROM POLICY FOR PORTION OF BUSINESS_TIME
FROM CURRENT DATE TO '12/31/9999'
WHERE EMPL='C054' AND PLCY='P667';
```

Bi-temporal example …

Step 1 - 09/21/2010 Employee C054 chooses HMO policy with $10 copay effective 1/1/2011
Step 2 - 09/24/2010 Update all policies P667 to a copay of $15 beginning 01/01/2011
Step 3 - 09/24/2010 Later on the same day(19.44) of 09/24, the customer cancelled the policy

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

```
DELETE FROM POLICY FOR PORTION OF BUSINESS_TIME
FROM CURRENT DATE TO '12/31/9999'
WHERE EMPL='C054' AND PLCY='P667';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	2010-09-24-19.44.47	HISTORY
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	2010-09-24-19.44.47	HISTORY
CO54	HMO	P667	$10	2004-01-01	2010-09-24	2010-09-24-19.44.47	9999-12-30-00.00.00	BASE

Business time example

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	2010-09-24-19.44.47	HISTORY
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	2010-09-24-19.44.47	HISTORY
CO54	HMO	P667	$10	2004-01-01	2010-09-24	2010-09-24-19.44.47	9999-12-30-00.00.00	BASE

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

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	2010-09-24	2010-09-24-19.44.47	9999-12-30-00.00.00	BASE

```
SELECT * FROM POLICY  FOR BUSINESS_TIME AS OF
   '2011-09-25' ORDER BY EFF_BEG;
```

No rows returned.
Business time only looks at base table, not the history table

System Period Versioning Information...

- **Base and History tables must be RECOVERed as a set**
  - VERIFYSET NO can override the need to RECOVER together
- **No utility operations that deletes data from base table**
  - LOAD REPLACE
  - REORG DISCARD
  - CHECK DATA DELETE YES
- **No CHECK utilities that invalidate AUX/LOB/XML**
- **PM31313 allows for ALTER TABLE...ADD COLUMN**
  - The same column is added to the History Table
- **PM31314 changed the high value to '9999-12-30-00.00.00.000000000000'**
- **The Base and History table, table spaces must be single table**
- **No temporal SELECT, UPDATE, or DELETE against the History**
- **Cannot be an MQT**
- **Cannot have a Clone Table, Column Mask, Row Permission, or Security Label column**

System Period Versioning Information...

- **Cannot TRUNCATE**
  - INSERT, UPDATE, DELETE, and MERGE are accepted
- **To find the Base / History Tables**

```
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: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.admin/src/tpc/db2z_addingsystime.htm
- **QUIESCE of the Base or History**
  - Will cause a quiesce against all tables in the versioning relationship, including auxiliary spaces

System Period Versioning Information

- **PM61811 added DB2 Supplied Stored Procedure for System Time overrides.**
  - SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY
  - When called, the remainder of the Unit of Work
    - Does not write History Table rows
    - Allows for the override of GENERATED ALWAYS columns
  - Primarily for data movement tooling
    - Replication
    - ETL
- **REPORT TABLESPACESET identifies versioning relationships in the system-maintained temporal table space or history table space**
- **CURSORs & VIEWs referencing system temporal table with a period specification will be READ ONLY**

Business Period Versioning Information

- **Business Time can be altered on an existing table**
  - See Information Center topic: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.admin/src/tpc/db2z_alteringtableaddbusitime.htm
- **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**
- **Should adopt the same high value as System Time**
  - 9999-12-30-00.00.00.000000000000

DB2 11 - Temporal Update and Archive Transparency

- **Temporal and Views**
- **Temporal Special Registers**
- **Archive Transparency**

Versioning & Views …

- **DB2 11 - You can use temporal predicates when referring to a view**
- **DB2 10 & DB2 11 - You can not use temporal predicates in a view**

Base Table

```
CREATE TABLE POLICY_BITEMPORAL
(EMPL VARCHAR(4) NOT NULL,
 TYPE VARCHAR(4),
 PLCY VARCHAR(4) NOT NULL,
 COPAY VARCHAR(4),
 SYS_BEG TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL,
 SYS_END TIMESTAMP(12) GENERATED ALWAYS AS ROW END   NOT NULL,
 SYS_TMP TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD SYSTEM_TIME   (SYS_BEG, SYS_END),
 BUS_BEG DATE NOT NULL,
 BUS_END DATE NOT NULL,
 PERIOD BUSINESS_TIME (BUS_BEG, BUS_END),
 PRIMARY KEY (EMPL,PLCY, BUSINESS_TIME WITHOUT OVERLAPS)
);
```

VIEW

```
CREATE VIEW VIEW_POLICY_BITEMPORAL_2012_ONLY AS
     SELECT * FROM POLICY_BITEMPORAL
 WHERE BUS_BEG <= '12/31/2012'
   AND BUS_END >= '01/01/2012' WITH CHECK OPTION;
```

```
CREATE VIEW VIEW_POLICY_BITEMPORAL_2012_ONLY AS
 SELECT * FROM POLICY_BITEMPORAL
 FOR BUSINESS_TIME FROM '01/01/2012' TO '12/30/2012';
```

SQLCODE -4736

- **Temporal predicates can now be used in DML on statements referencing views**

```
SELECT EMPL,TYPE,PLCY,COPAY,BUS_BEG,BUS_END
 FROM VIEW_POLICY_BITEMPORAL_2012_ONLY
 FOR BUSINESS_TIME AS OF '12/30/2012';
```

Versioning & Views Example …

- **Show how the date predicates on the view work with the FOR BUSINESS_TIME predicate in the SQL statement**

Rows in Base Table (POLICY_BITEMPORAL)

EMPL	TYPE	PLCY	COPAY	BUS_BEG	BUS_END
A054	HMO	P667	$10	2004-01-01	9999-12-31
B054	HMO	P667	$10	2013-01-01	9999-12-31
C054	HMO	P667	$10	2004-01-01	2011-12-31
D054	HMO	P667	$10	2012-01-01	2012-12-30
E054	HMO	P667	$10	2012-01-01	2014-12-30

- Remember the view

```
CREATE VIEW VIEW_POLICY_BITEMPORAL_2012_ONLY AS
     SELECT * FROM POLICY_BITEMPORAL
 WHERE BUS_BEG <= '12/31/2012'
   AND BUS_END >= '01/01/2012' WITH CHECK OPTION;
```

- Select Rows from view using AS OF business time

```
SELECT * FROM VIEW_POLICY_BITEMPORAL_2012_ONLY
    FOR BUSINESS_TIME AS OF '12/30/2012';
```

EMPL	TYPE	PLCY	COPAY	BUS_BEG	BUS_END
A054	HMO	P667	$10	2004-01-01	9999-12-31
~~B054~~	~~HMO~~	~~P667~~	~~$10~~	~~2013-01-01~~	~~9999-12-31~~
~~C054~~	~~HMO~~	~~P667~~	~~$10~~	~~2004-01-01~~	~~2011-12-31~~
~~D054~~	~~HMO~~	~~P667~~	~~$10~~	~~2012-01-01~~	~~2012-12-30~~
E054	HMO	P667	$10	2012-01-01	2014-12-30

Row is not in the view because BUS_BEG IS > 12/31/2012

Row is not in the view because BUS_END <=01/01/2012

Row is in the view, but not returned because Business End time is exclusive
BUS_END = 12/30/2012

Versioning & Views Temporal Modifications ...

- **UPDATE or DELETE with the FOR PORTION OF clause can be applied to Views**
- **Temporal modifications can cause rows to be split**
  - Rows that are created by splitting a row through a VIEW update may not be visible in the view after the update

```
UPDATE VIEW_POLICY_BITEMPORAL_2012_ONLY
    FOR PORTION OF BUSINESS_TIME
      FROM '05/01/2011' TO '10/31/2012'
    SET PLCY = 'PPO';
```

- **Symmetric Views are Views WITH CHECK OPTION**
  - Temporal modifications are not constrained by the check option
  - Split rows that disappear from the View definition are still allowed for a complete temporal modification

Versioning & Views Restrictions

- **Views referencing temporal tables must not reference a UDF**
- **Temporal UPDATE/DELETE can not have an INSTEAD OF Trigger**
- **Untyped parameter marker is not allowed in period specification or period clause with views**
  - Untyped parameter marker (BUS_BEG = ?)
  - Typed parameter marker (BUS_BEG = ? CAST AS DATE)

Temporal Special Registers …

- **Code and test applications using temporal data, possibly "without changing code"**
- **Run the same query for different times by changing the special registers**
- **Run AS OF any date by changing the special register**
- **Provides "Time Machine" capability**
  - Set the temporal special registers to a specific point in time
  - Effective for all subsequent SQL statements; Including those in invoked functions, stored procedures, and triggers
    - Allows the application to see data for a different point in time without modifying the SQL statements

Temporal Special Registers …

- **CURRENT TEMPORAL SYSTEM_TIME**
  - TIMESTAMP(12), nullable
- **CURRENT TEMPORAL BUSINESS_TIME**
  - TIMESTAMP(12), nullable
- **SET Temporal Registers**
  - For DRDA the value of the special register is sent to remote side for implicit connect
    - When using a 3-part name
  - If you use the special registers, they continue to be used for that session until you turn them off by setting them to NULL

```
SET CURRENT TEMPORAL SYSTEM_TIME = TIMESTAMP('2008-01-01') + 5 DAYS ;
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT TIMESTAMP - 1 YEAR ;
SET CURRENT TEMPORAL SYSTEM_TIME = NULL ;

SET CURRENT TEMPORAL BUSINESS_TIME = TIMESTAMP('2008-01-01') + 5 DAYS;
SET CURRENT TEMPORAL BUSINESS_TIME = CURRENT TIMESTAMP - 1 YEAR ;
SET CURRENT TEMPORAL BUSINESS_TIME = NULL ;
```

Temporal Special Registers …

- **Bind parameters determine if the Special Register will be honored when set**
  - SYSTIMESENSITIVE (YES / NO)
  - BUSTIMESENSITIVE (YES / NO)
  - BIND PACKAGE
    - Default Value - YES
  - REBIND PACKAGE
    - Default Value – Existing Option
  - REBIND TRIGGER PACKAGE
    - Default Value – Existing Option
- **SYSTEM_TIME SENSITIVE and BUSINESS_TIME SENSITIVE for Routines**
  - Options on CREATE / ALTER SQL Scalar Procedure or Function
  - INHERIT SPECIAL REGISTERS passes set values from invoker by default
    - DEFAULT SPECIAL REGISTERS will reset to NULL
- **DB2I support**
  - Set CHANGE DEFAULTS to YES to find these options

Temporal Special Registers System Time ...

- Base table
  - All Copays are currently $15
  - All were UPDATEd on 9/24/2013

EMPL	SYS_BEG	SYS_END	COPAY	BUS_BEG	BUS_END
A054	2013-09-24	9999-12-30	$15	2001-01-01	9999-12-31
B054	2013-09-24	9999-12-30	$15	2013-01-01	9999-12-31
C054	2013-09-24	9999-12-30	$15	2004-01-01	2011-12-31
D054	2013-09-24	9999-12-30	$15	2012-01-01	2012-12-30
E054	2013-09-24	9999-12-30	$15	2012-01-01	2014-12-30

- History table
  - Copays were different values in the past
  - Recorded in History by the 9/24/2013 UPDATE

EMPL	SYS_BEG	SYS_END	COPAY	BUS_BEG	BUS_END
A054	2013-08-23	2013-09-24	$20	2001-01-01	9999-12-31
B054	2013-08-02	2013-09-24	$10	2013-01-01	9999-12-31
C054	2013-08-02	2013-09-24	$10	2004-01-01	2011-12-31
D054	2013-08-02	2013-09-24	$10	2012-01-01	2012-12-30
E054	2013-08-02	2013-09-24	$20	2012-01-01	2014-12-30

Temporal Special Registers System Time ...

- **Set CURRENT TEMPORAL SYSTEM_TIME register to before the UPDATE**
- **SELECT all rows that were in effect at that time**

```
SET CURRENT TEMPORAL SYSTEM_TIME
      = '2013-09-20-00.00.00.123123123123';

SELECT EMPL
      ,DATE(SYS_BEG) AS SYS_BEG
      ,DATE(SYS_END) AS SYS_END
      ,COPAY
      ,BUS_BEG
      ,BUS_END
FROM POLICY_BITEMPORAL
ORDER BY EMPL, SYS_BEG DESC;
```

Explain shows UNION ALL

QBLOCKNO	TABLE_NAME	METHOD	TABNO	QBLOCK TYPE	EXPANSION REASON
1	POLICY_HISTORY	0	2	NCOSUB	S
2		3	0	UNIONA	S
5	POLICY_BITEMPORAL	0	1	NCOSUB	S

- **Rows all have the before update occurred**

EMPL	SYS_BEG	SYS_END	COPAY	BUS_BEG	BUS_END
A054	2013-08-23	2013-09-24	$20	2001-01-01	9999-12-31
B054	2013-08-02	2013-09-24	$10	2013-01-01	9999-12-31
C054	2013-08-02	2013-09-24	$10	2004-01-01	2011-12-31
D054	2013-08-02	2013-09-24	$10	2012-01-01	2012-12-30
E054	2013-08-02	2013-09-24	$20	2012-01-01	2014-12-30

Temporal Special Registers System Time ...

- **Reset CURRENT TEMPORAL SYSTEM_TIME register to NULL**
- **SELECT rows from base table (Same SELECT statement)**

```
SET CURRENT TEMPORAL SYSTEM_TIME = NULL;

SELECT EMPL
      ,DATE(SYS_BEG) AS SYS_BEG
      ,DATE(SYS_END) AS SYS_END
      ,COPAY
      ,BUS_BEG
      ,BUS_END
FROM POLICY_BITEMPORAL
ORDER BY EMPL, SYS_BEG DESC;
```

Explain shows no access to history table

QBLOCKNO	TABLE_NAME	METHOD	TABNO	QBLOCK TYPE	EXPANSION REASON
1		3	0	SELECT	
1	POLICY_BITEMPORAL	0	1	SELECT	

- **All rows all have COPAY value of $15 which is the current value in the table**

EMPL	SYS_BEG	SYS_END	COPAY	BUS_BEG	BUS_END
A054	2013-09-24	9999-12-30	$15	2001-01-01	9999-12-31
B054	2013-09-24	9999-12-30	$15	2013-01-01	9999-12-31
C054	2013-09-24	9999-12-30	$15	2004-01-01	2011-12-31
D054	2013-09-24	9999-12-30	$15	2012-01-01	2012-12-30
E054	2013-09-24	9999-12-30	$15	2012-01-01	2014-12-30

Temporal Special Registers System Time ...

- **Two section implementation**
  - Avoids runtime performance degradation
  - SYSTIMESENSITIVE(YES) cause system time and bi-temporal tables to bind SQL twice
  - When issuing a SELECT * FROM system time table
    - Original section
      - Bind the original SELECT * FROM system time table for current data
      - The majority of system-period temporal applications request for current data only
      - There is no performance degradation caused by UNION ALL query transformation
    - Extended section
      - Extends original SQL by adding a UNION ALL to the associated History Table
  - For the application, there is no change
- **At execution time:**
  - If the CURRENT TEMPORAL SYSTEM_TIME special register contains NULL value (the default)
    - The original section is executed
  - If the CURRENT TEMPORAL SYSTEM_TIME special register contains a Timestamp
    - The extended section is executed

Temporal Special Registers ...

- **In PLAN_TABLE, there is a new column called EXPANSION_REASON, which is populated when statements reference temporal or archive tables**
  - **A**: Query has implicit query transformation as a result of the SYSIBMADM.GET_ARCHIVE built-in global variable
  - **B**: Query has implicit query transformation as a result of the CURRENT TEMPORAL BUSINESS_TIME special register
  - **S**: Query has implicit query transformation as a result of the CURRENT TEMPORAL SYSTEM_TIME special register
  - **SB**: Query has implicit query transformation as a result of BOTH the CURRENT TEMPORAL SYSTEM_TIME special register and the CURRENT TEMPORAL BUSINESS_TIME special register
  - Blank: The query does not contain implicit query transformation

```
SET CURRENT TEMPORAL BUSINESS_TIME = TIMESTAMP('01/01/2013');
SET CURRENT TEMPORAL SYSTEM_TIME   = TIMESTAMP('01/01/2013');

EXPLAIN PLAN SET QUERYNO = 13 FOR
  SELECT
   EMPL,COPAY,BUS_BEG,BUS_END
  FROM POLICY_BITEMPORAL ORDER BY EMPL, BUS_BEG;

SELECT EXPANSION_REASON FROM PLAN_TABLE;
```

```
EXPANSION_REASON
---------+------
SB
```

Archive Transparency

- **What is the purpose of archiving?**
  - When you want to delete rows from the table, but need to keep the deleted rows for legal or business purposes
  - To move data to a cheaper storage medium
  - When you do not need to access the old data often, but need to be able to retrieve the data quickly
  - When you do not care about the lineage of a row
    - This means that you do not care about the changes to a row over time
- **No requirement for date/time columns to enable Archive Transparency**
- **Temporal and Archive Tables are mutually exclusive**
  - Not able to have Archive Table on an table with either Business Time or System Time
- **May use REORG DISCARD to Archive large amounts of data to facilitate migration**
  - User responsible for loading data from DISCARD file into the archive table

Archive Transparency Compared to System Time

	System Time	Archive Transparency
Schema -- two table approach	current table & history table same column #, column name, column attributes (data type, etc)	archive-enabled table & archive table same column #, column name, column attributes (data type, etc)
ROW BEGIN/ROW END/TRANS ID columns	mandatory	none
Period concept	yes – SYSTEM_TIME period	none, not compatible with system time
Compatible with Business Time tables	yes	no
Bind option	SYSTIMESENSITIVE	ARCHIVESENSITIVE
Implicit union all query transformation	controlled by CURRENT TEMPORAL SYSTEM_TIME special register	controlled by built-in global variable SYSIBMADM.GET_ARCHIVE
Data propagation to history/archive table	UPDATE and DELETE	DELETE SYSIBMADM.MOVE_TO_ARCHIVE
Implicit Static DMLs	Implicit two section design	Implicit two section design

Archive Transparency

- **Settings for BIND controls the sensitivity of the SYSIBMADM.GET_ARCHIVE global variable:**
  - ARCHIVESENSITIVE (default YES) – packages (No space between ARCHIVE and SENSITIVE))
    - BIND PACKAGE
    - REBIND PACKAGE
    - REBIND TRIGGER PACKAGE
    - CREATE TRIGGER (implicit trigger package)
  - ARCHIVE SENSITIVE (default YES) – UDFs and Stored Procedures (space between ARCHIVE & SENSITIVE)
    - CREATE FUNCTION (SQL scalar)
    - ALTER FUNCTION (SQL scalar)
    - CREATE PROCEDURE (SQL native)
    - ALTER PROCEDURE (SQL native)
- **REBIND a package with change ARCHIVESENSITIVE purges all copies of a package**
- **APREUSE and APCOMPARE are valid options**
- **You can set the EXPANSION_REASON in the Access Path repository**
- **DB2I Panels provide BIND/REBIND support for ARCHIVESENSITIVE**

Archive Transparency Global Variables ...

- **Built-in Global Variables which impact archival tables and processing**
  - Defined as CHAR(1) NOT NULL DEFAULT ‘N’
  - READ authority granted to PUBLIC
  - SYSIBMADM.GET_ARCHIVE
    - Determines if SELECTs against Archive Enabled (Base) Tables automatically UNION ALL the associated Archive Table
    - 'Y‘ includes the UNION ALL to Archive Tables
  - SYSIBMADM.MOVE_TO_ARCHIVE
    - Determines if deleted rows of Archive Enabled Tables are inserted into associated Archive Tables
    - ‘Y’: INSERT and UPDATE not allowed against the Archive Enabled (Base) Tables
    - ‘E’: INSERT and UPDATE allowed against the Base Tables

Archive Transparency Global Variables

- **To SELECT data from both the base and archive tables**
  - Set the built-in global variable SYSIBMADM.GET_ARCHIVE to ‘Y’
  - All subsequent SQL statements including those from invoked functions, stored procedures, and triggers will access both the base and archive table to retrieve the data via DB2 adding a UNION ALL predicate for the two tables
  - Package must be bound with ARCHIVESENSITIVE(YES)
- **To access data from only the base table**
  - Set the built-in global variable SYSIBMADM.GET_ARCHIVE to ‘N’ (Default)
  - All subsequent SQL statements including those from invoked functions, stored procedures, and triggers will access data from only the base tables

Archive Transparency Example ...

Base Table

```
CREATE TABLE POLICY_BASE
(EMPL VARCHAR(4) NOT NULL,
 TYPE VARCHAR(4),
 PLCY VARCHAR(4) NOT NULL,
 COPAY VARCHAR(4),
 START_DATE   DATE NOT NULL,
 TIMESTAMP1   TIMESTAMP NOT NULL GENERATED ALWAYS
   FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP,
 PRIMARY KEY (EMPL,PLCY));
```

Archive Table

```
CREATE TABLE POLICY_ARCHIVE
(EMPL VARCHAR(4) NOT NULL,
 TYPE VARCHAR(4),
 PLCY VARCHAR(4) NOT NULL,
 COPAY VARCHAR(4),
 START_DATE   DATE NOT NULL,
 TIMESTAMP1 TIMESTAMP NOT NULL GENERATED ALWAYS
   FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP,
 PRIMARY KEY (EMPL,PLCY));
```

OR

```
CREATE TABLE POLICY_ARCHIVE
  LIKE POLICY_BASE
  INCLUDING ROW CHANGE TIMESTAMP;
```

Activate archiving

```
ALTER TABLE POLICY_BASE ENABLE ARCHIVE USE POLICY_ARCHIVE;
```

- Create the base table
- Create the archive table
- Tell DB2 to associate the base table with the archive table
- ALTER ADD COLUMN to the Base Table propagates the column to the Archive Table

Archive Transparency Example ...

- Archive all rows where START_DATE less than December 31, 2010

```
EMPL   TYPE   PLCY   COPAY   START_DATE   TIMESTAMP1                    EMPL_LAST_NAME
---------+--------+--------+--------+--------+--------+--------+--------+---------
A207   HMO    P667   $10     2007-01-01   2013-07-30-20.07.33.136488   --------------
A208   HMO    P667   $10     2008-01-01   2013-07-30-20.07.33.137805   --------------
A209   HMO    P667   $10     2009-01-01   2013-07-30-20.07.33.139949   --------------
A210   HMO    P667   $10     2010-01-01   2013-07-30-20.07.33.141584   --------------
A211   HMO    P667   $10     2011-01-01   2013-07-30-20.07.33.144117   --------------
A212   HMO    P667   $10     2012-01-01   2013-07-30-20.07.33.153135   --------------
```

Archive-enabled table has 6 rows

- Set the Global variable MOVE_TO_ARCHIVE to 'Y' and then issue the DELETE command where the START_DATE is prior to December 31, 2010

```
SET SYSIBMADM.MOVE_TO_ARCHIVE = 'Y';
DELETE FROM POLICY_BASE WHERE START_DATE < '2010-12-31';
```

- The rows that were deleted from the base table are inserted into the archive table
- The Timestamp in the Archive Table has the time the row was archived, not the time in the base table

```
SELECT * FROM POLICY_BASE;

EMPL   TYPE   PLCY   COPAY   START_DATE   TIMESTAMP1                    EMPL_LAST_NAME
---------+--------+--------+--------+--------+--------+--------+--------+---------
A211   HMO    P667   $10     2011-01-01   2013-07-30-20.07.33.144117   --------------
A212   HMO    P667   $10     2012-01-01   2013-07-30-20.07.33.153135   --------------

SELECT * FROM POLICY_ARCHIVE;

EMPL   TYPE   PLCY   COPAY   START_DATE   TIMESTAMP1                    EMPL_LAST_NAME
---------+--------+--------+--------+--------+--------+--------+--------+---------
A207   HMO    P667   $10     2007-01-01   2013-07-30-20.07.33.216716   --------------
A208   HMO    P667   $10     2008-01-01   2013-07-30-20.07.33.227317   --------------
A209   HMO    P667   $10     2009-01-01   2013-07-30-20.07.33.227768   --------------
A210   HMO    P667   $10     2010-01-01   2013-07-30-20.07.33.227787   --------------
```

Microseconds are greater in the archive table than the base (archive-enabled) table

Archive Transparency Example ...

- To select data from both the base and archive tables
- Set GET_ARCHIVE global variable to ‘Y’ before the select statement

```
SET SYSIBMADM.GET_ARCHIVE = 'Y';
SELECT * FROM POLICY_BASE;

EMPL  TYPE  PLCY  COPAY  START_DATE  TIMESTAMP1                  EMPL_LAST_NAME
----------+---------+---------+---------+---------+---------+---------+---------+
A211  HMO   P667  $10    2011-01-01  2013-07-30-20.07.33.144117  ---------------
A212  HMO   P667  $10    2012-01-01  2013-07-30-20.07.33.153135  ---------------
A207  HMO   P667  $10    2007-01-01  2013-07-30-20.07.33.216716  ---------------
A208  HMO   P667  $10    2008-01-01  2013-07-30-20.07.33.227317  ---------------
A209  HMO   P667  $10    2009-01-01  2013-07-30-20.07.33.227768  ---------------
A210  HMO   P667  $10    2010-01-01  2013-07-30-20.07.33.227787  ---------------
```

- To select data from only the base table
- Set GET_ARCHIVE global variable to ‘N’ before the select statement

```
SET SYSIBMADM.GET_ARCHIVE = 'N';
SELECT * FROM POLICY_BASE;

EMPL  TYPE  PLCY  COPAY  START_DATE  TIMESTAMP1                  EMPL_LAST_NAME
----------+---------+---------+---------+---------+---------+---------+---------+
A211  HMO   P667  $10    2011-01-01  2013-07-30-20.07.33.144117  ---------------
A212  HMO   P667  $10    2012-01-01  2013-07-30-20.07.33.153135  ---------------
```

Archive Transparency Example ...

- **UPDATE SQL statements will only update base table rows**
  - Regardless of whether the GET_ARCHIVE is set to Y or N
- **In this example, we set the GET_ARCHIVE to 'Y' so see if the SELECT will retrieve rows from both Base and Archive tables**
  - Note that only the **Base** table rows were updated

```
SET SYSIBMADM.GET_ARCHIVE = 'Y';
UPDATE POLICY_BASE SET COPAY = '$15';
SELECT * FROM POLICY_BASE;

EMPL  TYPE  PLCY  COPAY  START_DATE  ARCHIVE_TIMESTAMP          EMPL_LAST_NAME
--------+-------+-------+-------+--------+--------+--------+--------+
A211  HMO   P667  $15    2011-01-01  2013-07-30-21.17.31.731257  ---------------
A212  HMO   P667  $15    2012-01-01  2013-07-30-21.17.31.731311  ---------------
A207  HMO   P667  $10    2007-01-01  2013-07-30-20.07.33.216716  ---------------
A208  HMO   P667  $10    2008-01-01  2013-07-30-20.07.33.227317  ---------------
A209  HMO   P667  $10    2009-01-01  2013-07-30-20.07.33.227768  ---------------
A210  HMO   P667  $10    2010-01-01  2013-07-30-20.07.33.227787  ---------------
```

Archive Transparency ...

- **MERGE works like INSERT or UPDATE**
- **When the BIND is performed with an archive enabled table**
  - We create two sections in the package when ARCHIVESENSITIVE is YES
    - First section - Base table only
    - Second section - Base table and archive table UNION'ed ALL together
  - ARCHIVESENSITIVE only refers to GET_ARCHIVE sensitivity
    - When the GET_ARCHIVE global variable is set to 'N'
      - DB2 will use the base table only section
    - When the GET_ARCHIVE global variable is set to 'Y'
      - DB2 will use the base and archive table section
  - When MOVE_TO_ARCHIVE is set to 'Y'
    - DB2 will move rows to archive table on a DELETE even if ARCHIVESENSITIVE BIND option is set to NO
    - This prevents data not being archived if the program tells it to archive
  - When archiving data, you would usually set GET_ARCHIVE to 'N' and MOVE_TO_ARCHIVE as 'Y'

Archive Transparency EXPLAIN ...

- **In our example, Data in SYSPACKSTMT has**
  - SECTNO = 1 for archive-enabled table only and
  - SECTNO = 3 for archive-enabled table UNIONed with archive table
  - STATEMENT stored as the original statement with EXPANSION_REASON of 'A'

```
SELECT SECTNO,SEQNO, STMTNO,EXPANSION_REASON AS EXP,STATEMENT
FROM SYSIBM.SYSPACKSTMT
WHERE NAME = 'HHRDARC'
ORDER BY 1;
```

SECTNO	SEQNO	STMTNO	EXP	STATEMENT
0	0	0		
1	1	52		DECLARE CSR1 SENSITIVE DYNAMIC SCROLL CURSOROR SELECT EMPL , ARCHIVE_TIMESTAMP FROM POLICY_BASE
1	3	79		OPEN CSR1
1	4	83		FETCH CSR1 INTO : H , : H
1	5	112		FETCH CSR1 INTO : H , : H
2	2	75		SET GET ARCHIVE = : H
3	6	52	A	DECLARE CSR1 SENSITIVE DYNAMIC SCROLL CURSOR OR SELECT EMPL , ARCHIVE_TIMESTAMP FROM POLICY_BASE

Archive Transparency EXPLAIN ...

- Here you can see that there are two sections in the package in PLAN_TABLE
- Section 1 is the base section and will be used when GET_ARCHIVE='N'
- Section 3 is the expanded section and will be used when GET_ARCHIVE = 'Y'
- More information is available in
  - DSN_STATEMNT_TABLE
  - DSN_STAT_FEEDBACK
  - DSN_STRUCT_TABLE
  - DSN_DETCOST_TABLE

```
SELECT
  SECTNOI,
  QBLOCKNO,
  SUBSTR(TNAME,1,12) AS TABLE_NAME,
  TABNO,
  QBLOCK_TYPE,
  TABLE_TYPE,
  EXPANSION_REASON
  FROM DNET775 . PLAN_TABLE
  WHERE PROGNAME = 'HHRDARC'
  ORDER BY SECTNOI,QBLOCKNO;
```

PLAN_TABLE (selected columns)

SECTNOI	QBLOCKNO	TABLE_NAME	TABNO	QBLOCK_TYPE	TABLE_TYPE	EXPANSION_REASON
1	1	POLICY_BASE	1	SELECT	T	
3	1	POLICY_ARCHI	2	NCOSUB	T	A
3	2		0	UNIONA	----------	A
3	5	POLICY_BASE	1	NCOSUB	T	A

Archive Transparency Management ...

- **LOAD RESUME can be used to archive data**
  - You can add your own data into the archive table
  - Use this when doing a REORG with DISCARD to load the DISCARD file into the ARCHIVE table

Archive Transparency Management

- **Use the ALTER TABLE ... DISABLE clause to remove relationship between base and archive tables**
- **When Archive relationship is enabled, archive table is type 'R' and there are values in the ARCHIVING_TABLE column for both tables**
- **When Archive relationship is removed, table that was the archive table is a regular table type = 'T' and there are no entries in the ARCHIVING_TABLE column of either table**

```
SELECT SUBSTR(NAME,1,30) AS TABLENAME
      ,TYPE
      ,SUBSTR(ARCHIVING_SCHEMA,1,8) AS ASCHEMA
      ,SUBSTR(ARCHIVING_TABLE,1,18) AS ARCHIVING_TABLE
FROM SYSIBM.SYSTABLES
WHERE NAME IN ('POLICY_BASE','POLICY_ARCHIVE')
```

```
TABLENAME                      TYPE  ASCHEMA   ARCHIVING_TABLE

POLICY_BASE                    T     DNET775   POLICY_ARCHIVE
POLICY_ARCHIVE                 R     DNET775   POLICY_BASE
```

Before Disable Archive

```
ALTER TABLE POLICY_BASE DISABLE ARCHIVE;
```

```
TABLENAME                      TYPE  ASCHEMA   ARCHIVING_TABLE

POLICY_BASE                    T
POLICY_ARCHIVE                 T
```

After Disable Archive

Archive Transparency Comparison

- **Archive Transparency**
  - Works on a single table
  - Deletes the entire row from the base table
  - Inserts the deleted row into a DB2 archive table
  - May not satisfy legal archival requirements
- **IBM InfoSphere Optim Data Growth Solution**
  - Works on business objects
  - Can delete selected rows (keep customer, delete orders) from the base table
  - Writes row to a non updateable extract file
  - Satisfies legal archival requirements

Additional Temporal Topics

- **DB2 for z/OS Temporal Proof of Technology**
  - One day learning opportunity
  - Brief lecture / followed by detailed labs
  - Temporal Labs
    - Implementing Business Time
    - Implementing System Time
    - Working with Business Time (showing System Time impacts)
    - Working with System Time (discover row lineage)
    - Temporal Simulation

Temporal Data – IBM DB2 Tools Support

- **DB2 Administration Tool and Object Compare**
  - Alter, Migrate, Compare, Create
- **DB2 Table Editor**
  - Insert, Update, Delete
- **SQL Performance Analyzer**
  - Business Time & System time
- **Log Analysis Tool**
  - Externalized Business and/or System Time
- **Data Studio (3.1.1)**
  - Versioning properties, SQL assist, etc.

Summary of DB2 Temporal Data and transparent Archiving

- **Support for time related data**
- **Improve developer productivity**
- **Ensure consistent handling of time related data**
- **Ability to archive data and retrieve it easily**