DB2 11 for z/OS Transparent Archiving (aka as database versioning)

Triangle DB2 User's Group Meeting December 4, 2014

Stan Goodwin
Technical Sales Specialist
DB2 Advisor for z/OS
IBM Mid-Atlantic Region
segoodw@us.ibm.com

Disclaimer and Trademarks

Agenda

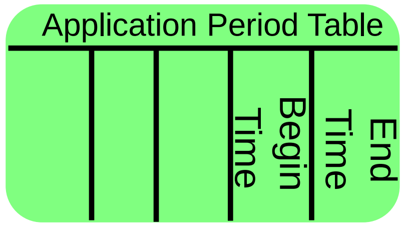
- DB2 10 Temporal Tables – review and what's new
  - New Temporal Special Registers for v11
  - Temporal predicates in DB2 views
- DB2 11 Transparent Archiving
  - What is it?
  - Archive management
  - Examples
- Summary / Q&A

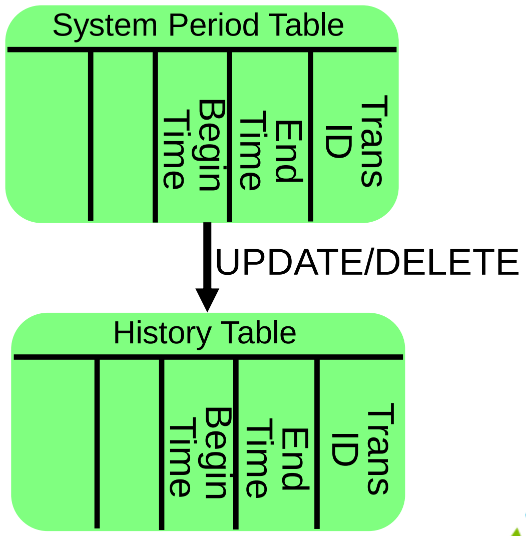
DB2 11 Objectives for Table Versioning

- Remove some restrictions for implementing Temporal Tables (introduced w/DB2 10)
- Provide easier development with temporal tables
- Ease manageability <u>by reducing the size of the active database</u>
  - Great use case for Transparent Archiving!

Temporal High Level Review

- Business Time
  - Begin & End business time columns
  - Set by the application
  - Modifications to SQL
  - Update/Delete modify periods
    - Could split rows to preserve Business Time
- System Time
  - Begin, End, and Trans time columns
  - Maintained by DB2
  - History Table for previous row versions
  - UPDATE/DELETE
    - DB2 populates the History Table
  - DML changes for SELECT only
    - Implicit UNION ALL to query History



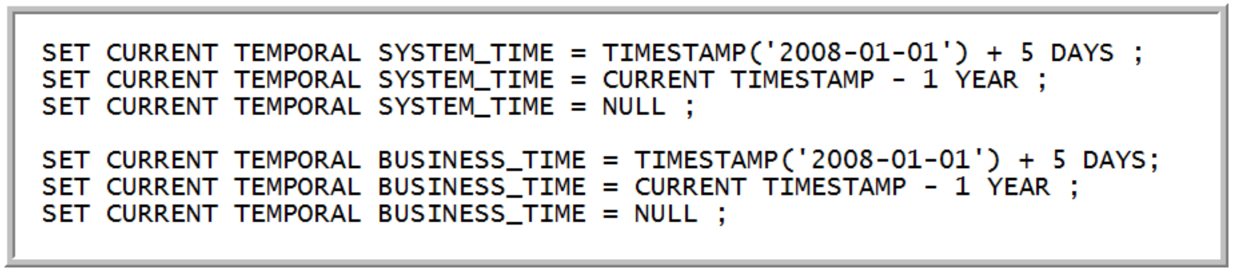


Temporal Special Registers …

- Enable customers to be able to code applications using temporal data and to be able to test the system, possibly “without changing code”
- DB2 will be able to run the same query for different times by changing the special registers
- Have the ability to run AS OF any date by changing the special register
- Provides “Time Machine” capability
  - Setting the temporal special registers to a specific point in time
  - Works for all subsequent SQL statements
  - Including those in invoked functions, stored procedures, and triggers
  - This allows the application to see data from 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
  - We add these predicates when there are direct or indirect references to Business Time or System Time tables



```
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

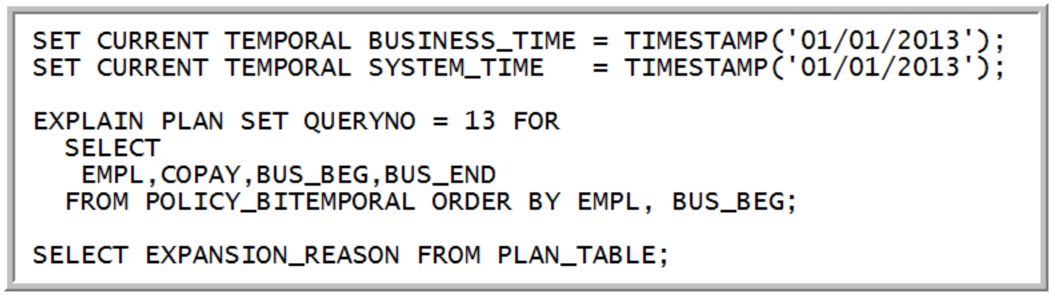
- Bind parameters determine if the Special Register will be honored when set
  - SYSTIMESENSITIVE (YES / NO)
  - BUSTIMESENSITIVE (YES / NO)
  - BIND PACKAGE
    - Default Value - YES
  - REBIND PACKAGE
    - Default Value – Existing Option
  - REBIND TRIGGER PACKAGE
    - Default Value – Existing Option
- 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 ...

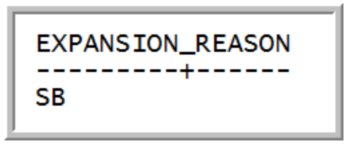
- Bind / Rebind two section implementation
  - Bind/Rebind keywords cause System Time / Bitemporal tables to bind SQL twice
  - Original section
    - Bind the original SQL for non-temporal access / no temporal predicates
    - The majority of System Time applications request for current data only
      - There is no performance degradation caused by UNION ALL query transformation
  - Extended section
    - System Time SQL uses UNION ALL to the associated History Table
    - Business Time & System Time temporal predicates, as appropriate
  - SQL accessing temporal tables does not have to be changed
- At execution time
  - If the Temporal Special Register is not set (the default), Original Section used
  - If the Temporal Special Register is set, Extended Section used
- EXPANSION_REASON
  - In several Catalog & EXPLAIN tables to indentify extended SQL

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;
```



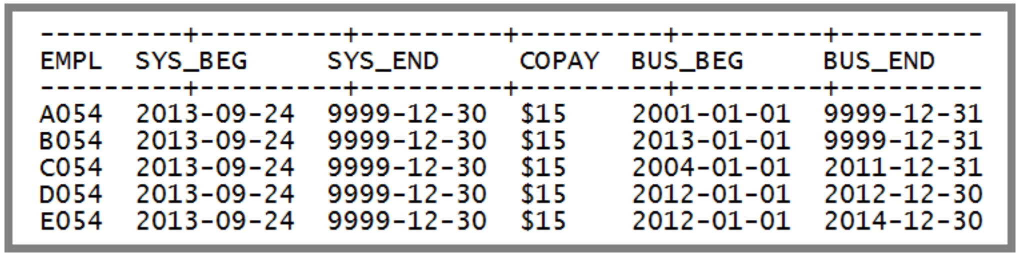
```
EXPANSION_REASON
---------+------
SB
```

Temporal Special Registers System Time ...

- If the register is set to a valid not null value DB2 will add the clause
  - FOR SYSTEM TIME AS OF CURRENT TEMPORAL SYSTEM_TIME
    - To static and dynamic SQL statements referencing System time and bi-temporal tables
    - Including indirect references in a View or via a Trigger
- An explicit FOR SYSTEM_TIME period specification produces SQLCODE -20524
  - You can not use FOR or AS OF SYSTEM_TIME with the special register
- Any INSERT, UPDATE, DELETE, or MERGE (data modification) statements against System Time tables produces SQLCODE -20535
- Data modification statements are allowed against Regular Tables

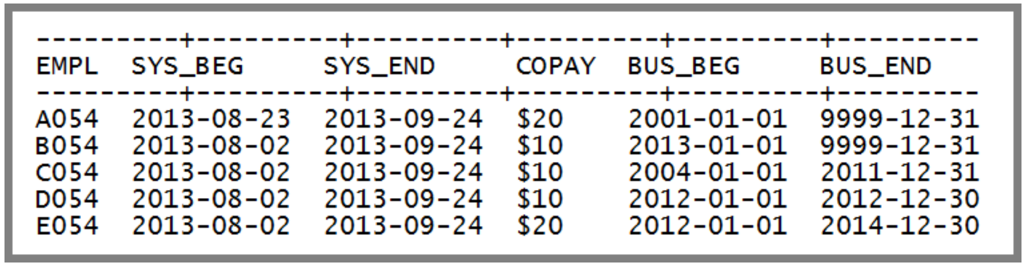
Temporal Special Registers System Time ...

- 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
```

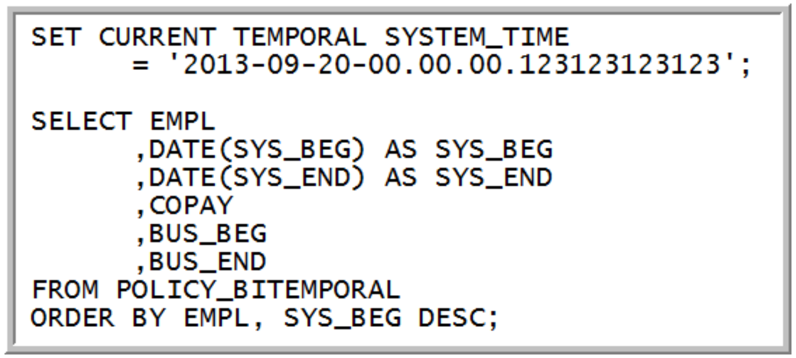
- History table
  - Copays were different values in the past
  - Recorded in History by the 9/24/2013 UPDATE



```
---------+---------+---------+---------+---------+---------
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
```

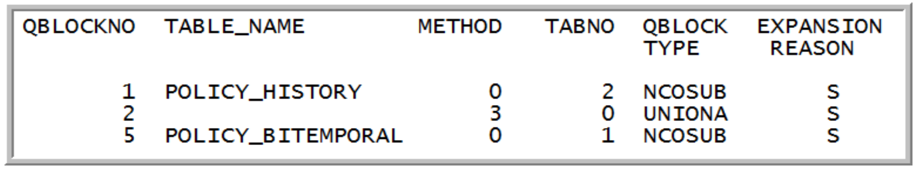
Temporal Special Registers System Time ...

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



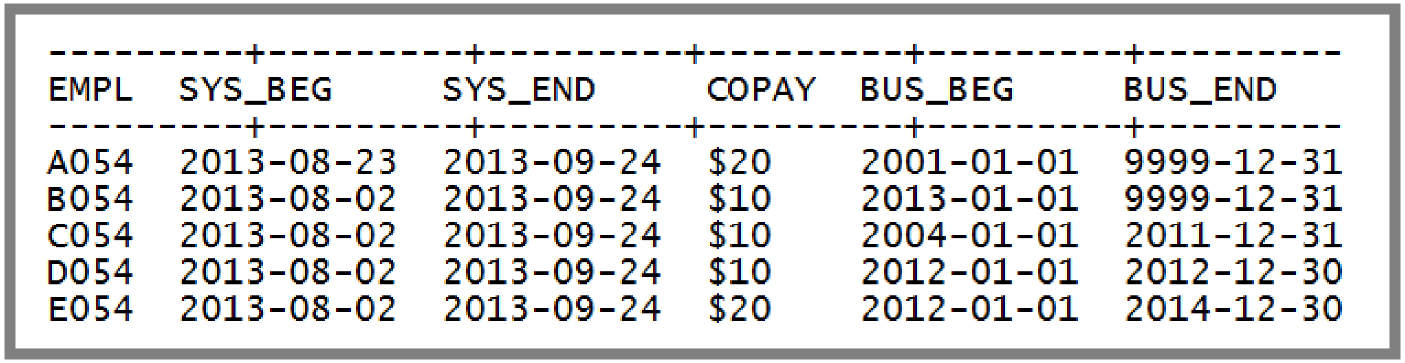
```
SET CURRENT TEMPORAL SYSTEM_TIME
      = '2013-09-20-00.00.00.123123123123';

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   EXPANSION
                                               TYPE      REASON

       1  POLICY_HISTORY            0       2  NCOSUB         S
       2                            3       0  UNIONA         S
       5  POLICY_BITEMPORAL         0       1  NCOSUB         S
```

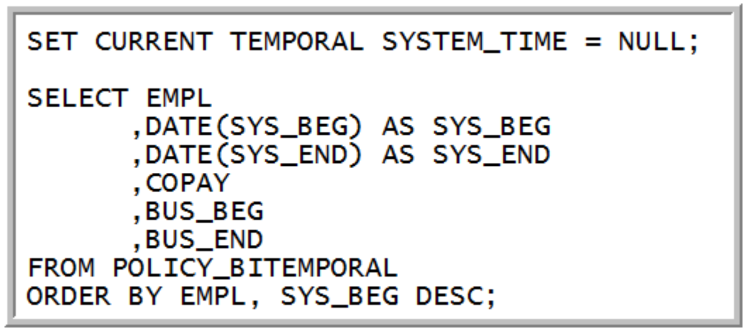
- **Rows all have the before update occurred**



```
---------+---------+---------+---------+---------+---------
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
```

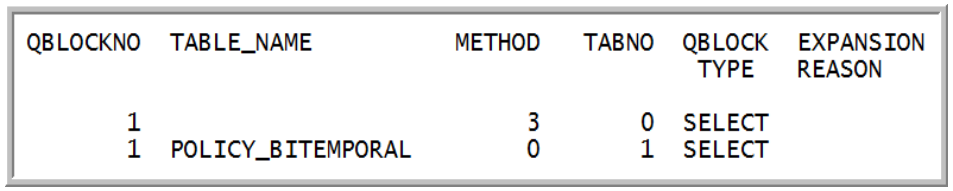
Temporal Special Registers System Time ...

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



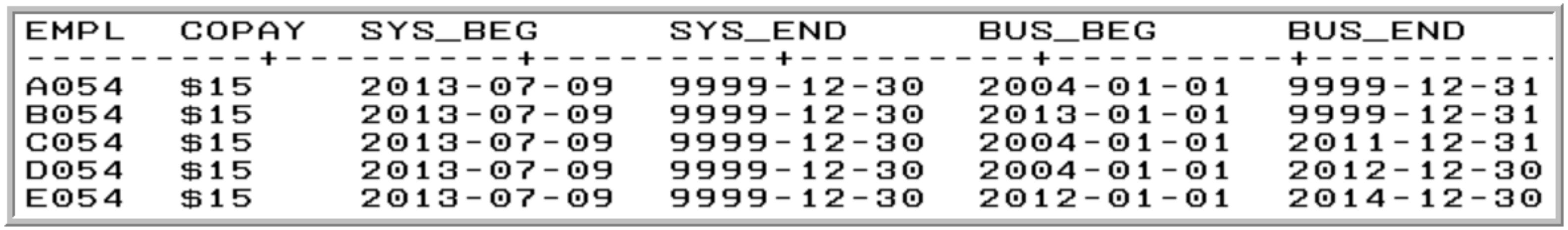
```
SET CURRENT TEMPORAL SYSTEM_TIME = NULL;

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  EXPANSION
                                                TYPE   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      COPAY     SYS_BEG      SYS_END      BUS_BEG      BUS_END
----------+----------+----------+----------+----------+----------
A054      $15       2013-07-09   9999-12-30   2004-01-01   9999-12-31
B054      $15       2013-07-09   9999-12-30   2013-01-01   9999-12-31
C054      $15       2013-07-09   9999-12-30   2004-01-01   2011-12-31
D054      $15       2013-07-09   9999-12-30   2004-01-01   2012-12-30
E054      $15       2013-07-09   9999-12-30   2012-01-01   2014-12-30
```

Temporal Special Registers Business Time ...

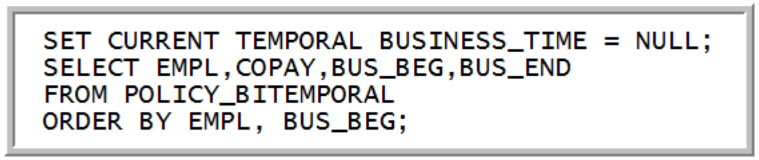
- If the register is set to a valid not null value DB2 will add the clause
  - FOR BUSINESS TIME AS OF CURRENT TEMPORAL BUSINESS_TIME
  - Static and dynamic SQL statements referencing business time and bi-temporal tables
  - Including indirect references in a View or via a Trigger
- DB2 will cast the CURRENT TEMPORAL BUSINESS_TIME to the column definition of either DATE or TIMESTAMP(6)
  - start_date <= CAST(CURRENT TEMPORAL BUSINESS_TIME AS DATE/TIMESTAMP(6))
  - end_date > CAST(CURRENT TEMPORAL BUSINESS_TIME AS DATE/TIMESTAMP(6))

Temporal Special Registers Business Time ...

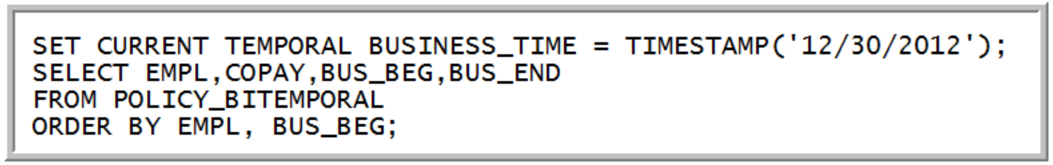
- An explicit FOR BUSINESS_TIME period specification produces an error
- For UPDATE or DELETE
  - Where CURRENT TEMPORAL BUSINESS_TIME register is not NULL and BUSTIMESENSITIVE(YES)
  - Predicates are generated as shown above
- FOR PORTION OF can also be included
  - Normal Temporal UPDATE or DELETE logic will be performed
  - Potentially splitting existing rows

Temporal Special Registers Business Time ...

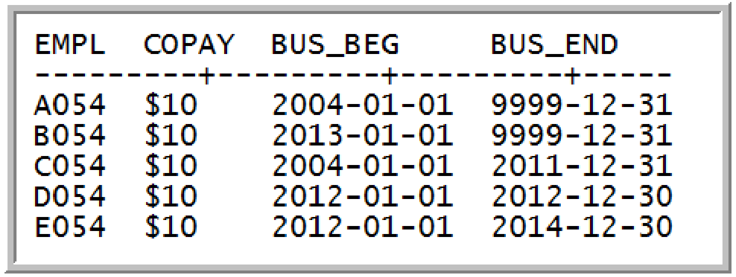
- **CURRENT TEMPORAL BUSINESS_TIME special register example**



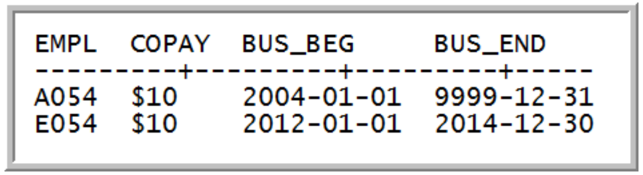
```
SET CURRENT TEMPORAL BUSINESS_TIME = NULL;
SELECT EMPL,COPAY,BUS_BEG,BUS_END
FROM POLICY_BITEMPORAL
ORDER BY EMPL, BUS_BEG;
```



```
SET CURRENT TEMPORAL BUSINESS_TIME = TIMESTAMP('12/30/2012');
SELECT EMPL,COPAY,BUS_BEG,BUS_END
FROM POLICY_BITEMPORAL
ORDER BY EMPL, BUS_BEG;
```



```
EMPL   COPAY  BUS_BEG     BUS_END
---------+---------+---------+-----
A054   $10    2004-01-01  9999-12-31
B054   $10    2013-01-01  9999-12-31
C054   $10    2004-01-01  2011-12-31
D054   $10    2012-01-01  2012-12-30
E054   $10    2012-01-01  2014-12-30
```



```
EMPL   COPAY  BUS_BEG     BUS_END
---------+---------+---------+-----
A054   $10    2004-01-01  9999-12-31
E054   $10    2012-01-01  2014-12-30
```

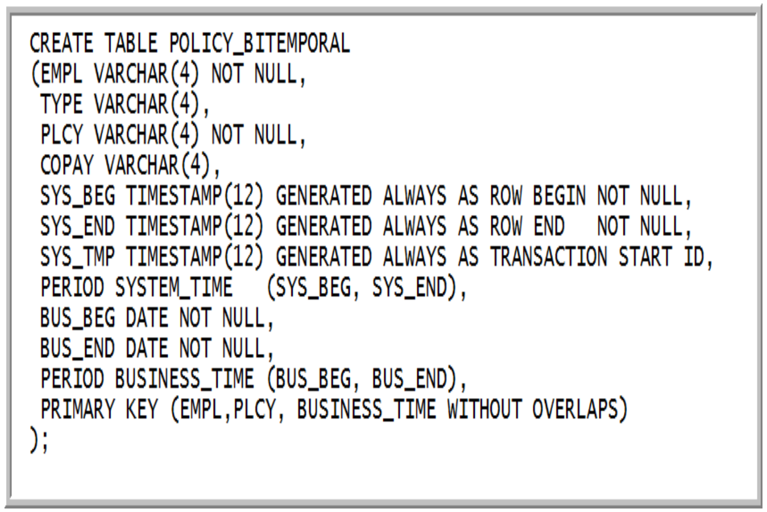
– The left side shows that when we set the CURRENT BUSINESS_TIME register to NULL
  - All qualifying rows returned

– The right side shows that when we set the CURRENT BUSINESS_TIME register to a value
  - Rows returned AS OF the specified time
  - Row D054 is not included because the BUS_END date is the same as the CURRENT BUSINESS_TIME register, and the BUS_END is exclusive, meaning that 2012-12-30 is not part of the row

Versioning & Views …

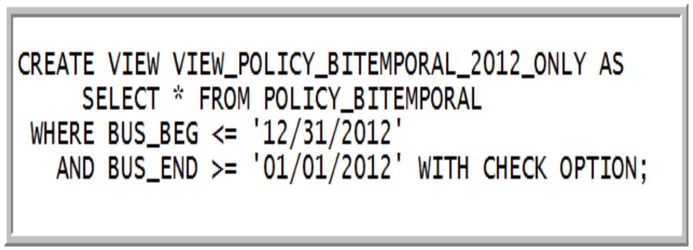
- DB2 11 - You can use temporal predicates when referring to a view
- DB2 10 & DB2 11 - You can not use temporal predicates in a view

Base Table

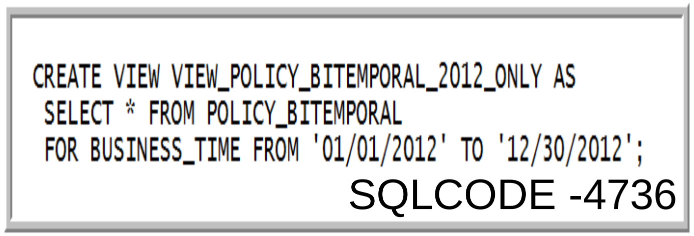


```
CREATE TABLE POLICY_BITEMPORAL
(EMPL VARCHAR(4) NOT NULL,
 TYPE VARCHAR(4),
 PLCY VARCHAR(4) NOT NULL,
 COPAY VARCHAR(4),
 SYS_BEG TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL,
 SYS_END TIMESTAMP(12) GENERATED ALWAYS AS ROW END   NOT NULL,
 SYS_TMP TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION 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;
```



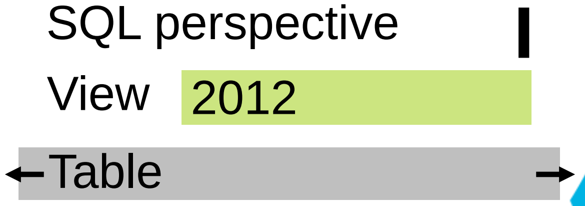
```
CREATE VIEW VIEW_POLICY_BITEMPORAL_2012_ONLY AS
 SELECT * FROM POLICY_BITEMPORAL
 FOR BUSINESS_TIME FROM '01/01/2012' TO '12/30/2012';
```

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';
```

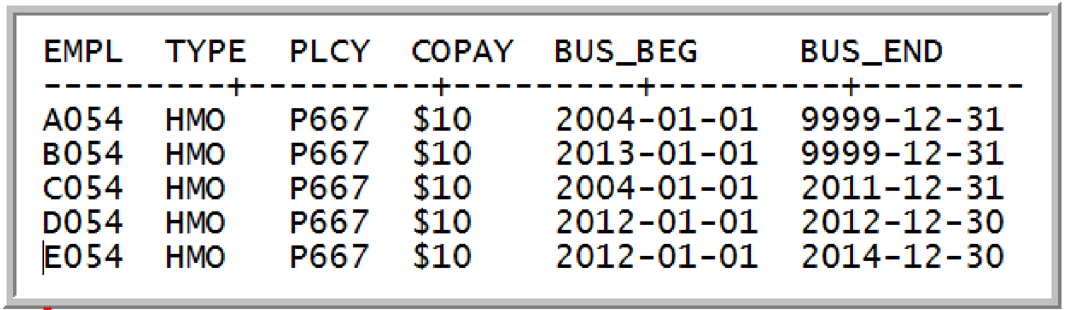
SQL perspective



Versioning & Views Example …

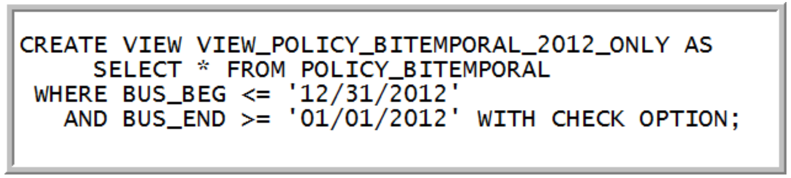
- **Show how the date predicates on the view work with the FOR BUSINESS_TIME predicate in the SQL statement**

Rows in Base Table (POLICY_BITEMPORAL)



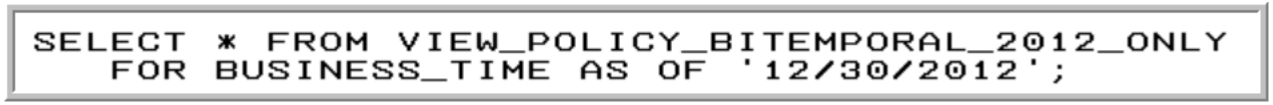
```
EMPL   TYPE   PLCY   COPAY  BUS_BEG     BUS_END
---------+---------+---------+---------+--------
A054   HMO    P667   $10    2004-01-01  9999-12-31
B054   HMO    P667   $10    2013-01-01  9999-12-31
C054   HMO    P667   $10    2004-01-01  2011-12-31
D054   HMO    P667   $10    2012-01-01  2012-12-30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

- Remember the view

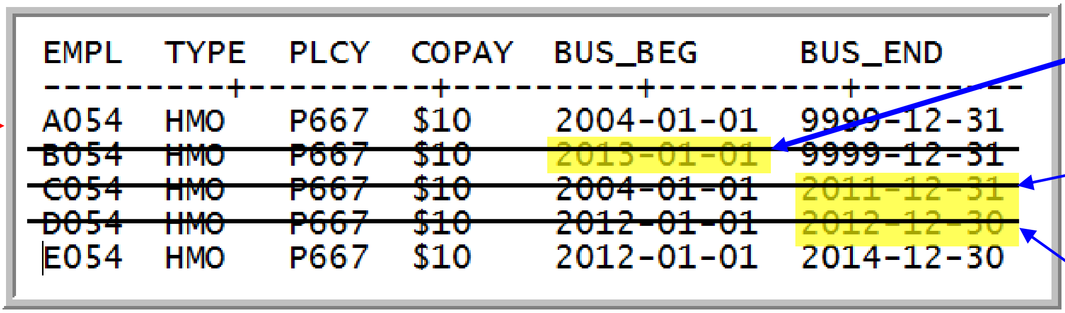


```
CREATE VIEW VIEW_POLICY_BITEMPORAL_2012_ONLY AS
     SELECT * FROM POLICY_BITEMPORAL
 WHERE BUS_BEG <= '12/31/2012'
   AND BUS_END >= '01/01/2012' WITH CHECK OPTION;
```

- 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

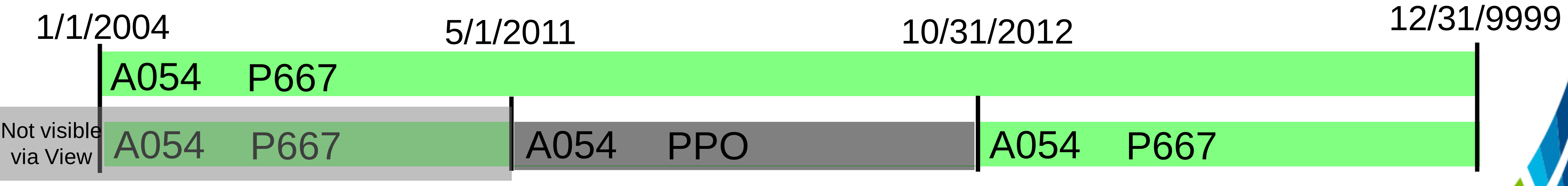
Row is in the view, but not returned because Business End time is exclusive
BUS_END = 12/30/2012

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



DB2 11 Archive Transparency

Why DB2 Archive Transparency

- In database system, querying and managing tables that contain a large amount of data is a common problem
  -> performance of maintaining large table is a key customer pain points

- One known solution is to archive the inactive/cold data to a different environment -- challenges on the ease of use and performance:
  - How to provide easy access to both current and archived data within single query
  - How to make data archiving and access “transparent” with minimum application changes

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
- Do we add extra columns for archiving like we do for system time tables?
  - You do not need extra columns to enable Archive Transparency
- Temporal and Archive Tables are mutually exclusive
  - Can not build an Archive Table on a table that has either Business Time or System Time
- Archive a large amount of data using REORG DISCARD to facilitate migration
  - User would be responsible for loading data from the DISCARD file into the archive table

Archive Transparency Compared to System Time

	System Time	Archive Transparency
Tables	Base table & History table same column #, column name, column attributes (data type, etc)	Base table & Archive table same column #, column name, column attributes (data type, etc)
Additional columns	ROW BEGIN/ROW END/TRANS ID columns	No additional columns required
Compatible with Period enabled tables	Compatible with Business Time	No
Data propagation to history/archive table	UPDATE and DELETE	DELETE SYSIBMADM.MOVE_TO_ARCHIVE Utilities can be used (REORG...DISCARD,LOAD...RESUME)
Implicit UNION ALL query transformation	Controlled by: CURRENT TEMPORAL SYSTEM_TIME special register & SYSTIMESENTIVE RE/BIND option EXPANSION_REASON=S	Controlled by built-in global variable SYSIBMADM.GET_ARCHIVE EXPANSION_REASON=A

Archive Transparency Management ...

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;
```

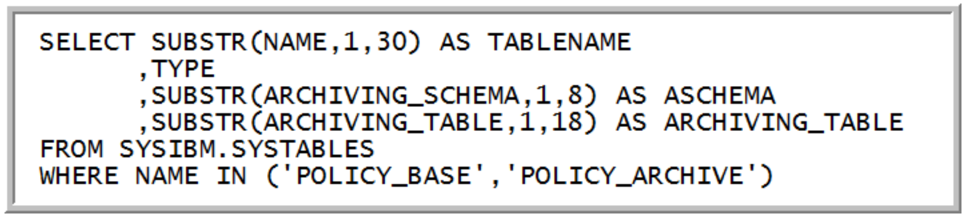
Activate archiving

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

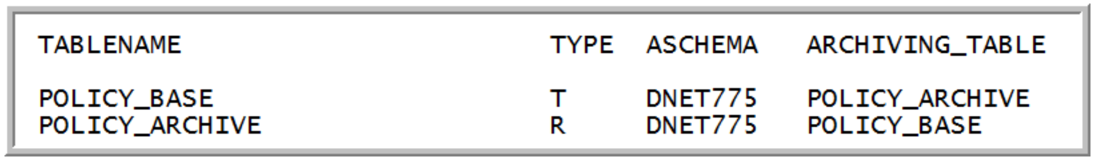
- Create the base table
- Create the archive table
- Tell DB2 to associate the base table with the archive table
- ALTER ADD COLUMN to the Base Table propagates the column to the Archive Table

Archive Transparency Management

- Use the ALTER TABLE ... DISABLE clause to remove relationship between base and archive tables
  - This may be required for Table schema ALTERs other than ADD COLUMN
- When Archive relationship is enabled
  - Archive table is TYPE is ‘R’
  - ARCHIVING_TABLE column is populated

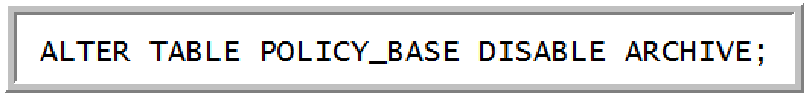


```
SELECT SUBSTR(NAME,1,30) AS TABLENAME
      ,TYPE
      ,SUBSTR(ARCHIVING_SCHEMA,1,8) AS ASCHEMA
      ,SUBSTR(ARCHIVING_TABLE,1,18) AS ARCHIVING_TABLE
FROM SYSIBM.SYSTABLES
WHERE NAME IN ('POLICY_BASE','POLICY_ARCHIVE')
```

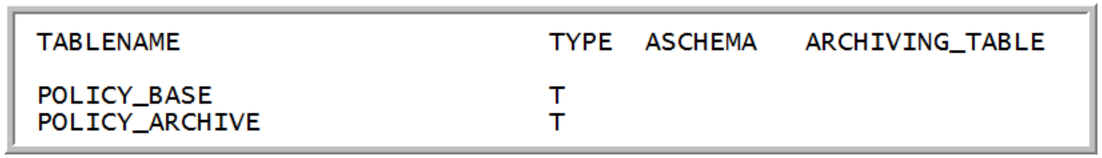


```
TABLENAME                         TYPE  ASCHEMA   ARCHIVING_TABLE

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 Global Variables ...

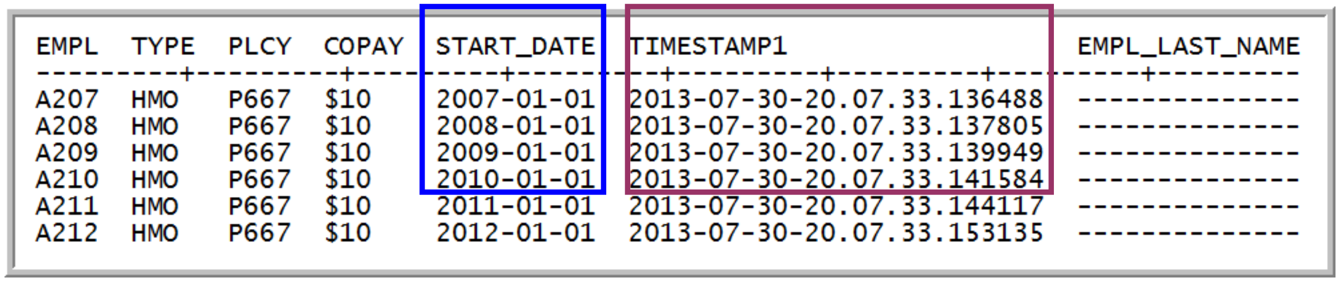
- Built-in Global Variables that impact archival tables & processing
  - Defined as CHAR(1) NOT NULL DEFAULT ‘N’
  - READ authority granted to PUBLIC
  - **SYSIBMADM.GET_ARCHIVE**
    - Determines if SELECTs against Archive Enabled (Base) Tables automatically UNION ALL the associated Archive Table
    - 'Y‘ includes the UNION ALL to Archive Tables
    - Packages must be bound ARCHIVESENSITVE(YES)
  - **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

- These settings for BIND will control 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)
- If you REBIND a package and change ARCHIVESENSITIVE, all copies of the package will be purged
- APREUSE and APCOMPARE are valid options
- You can set the EXPANSION_REASON in the Access Path repository
- DB2I Panels support ARCHIVESENSITIVE

Archive Transparency Example ...

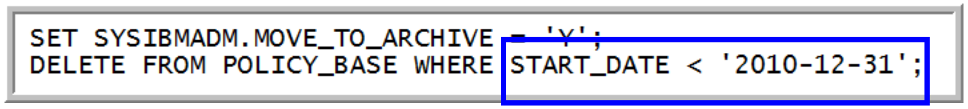
- **Archive all rows where START_DATE less than December 31, 2010**



```
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   --------------
```

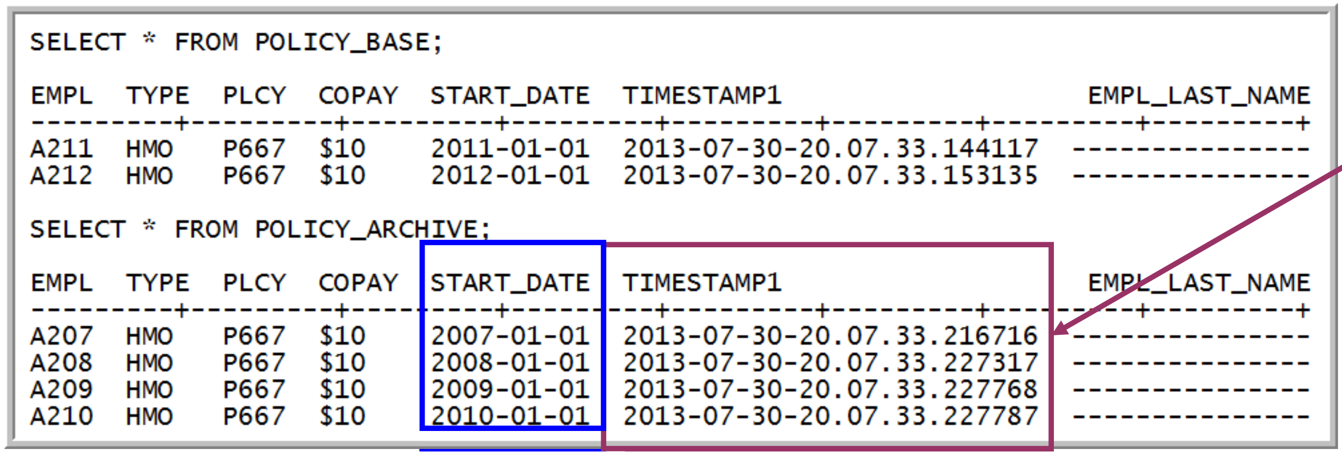
Archive-enabled table has 6 rows

- We 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

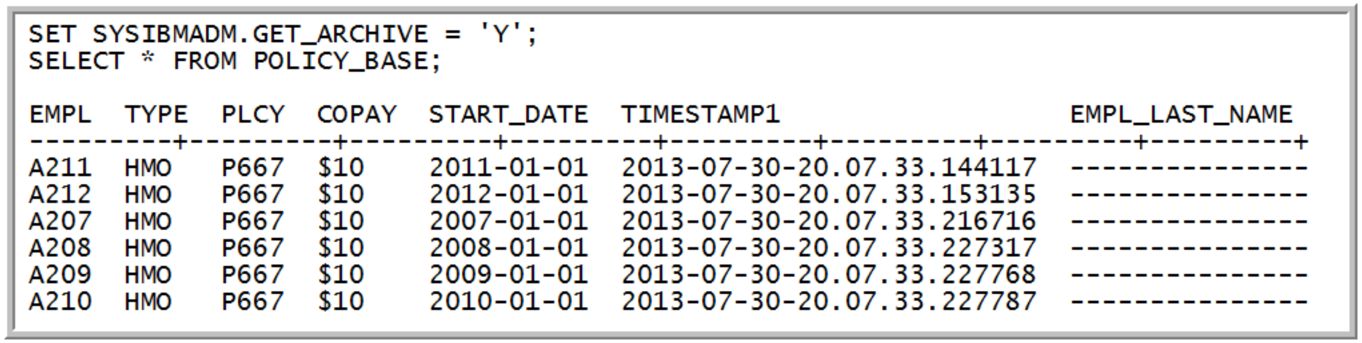


```
SELECT * FROM POLICY_BASE;

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

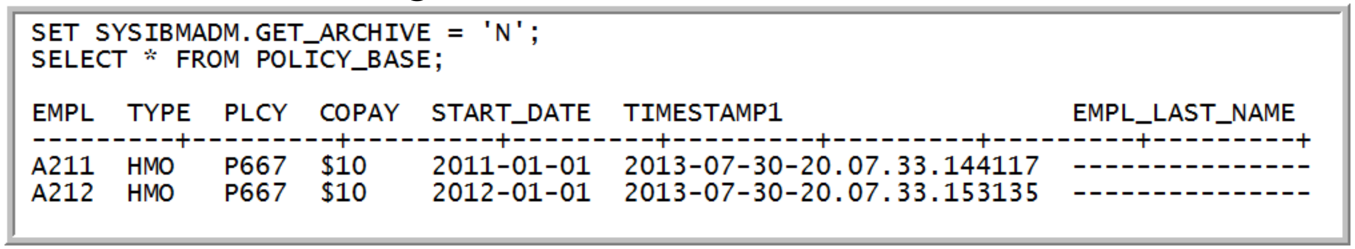
Archive Transparency Example ...

- To select data from both the base and archive tables
- Set **GET_ARCHIVE** global variable to ‘Y’ before the select statement
  - Can add an indicator column (ex. BASE_ARCHIVE CHAR(01)) to both tables to indicate from where the row was sourced
    - Base column defaults to one value (Ex. “B”)
    - INSERT Trigger on the Archive column to set a different value (Ex. “A”)



```
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   ---------------
```

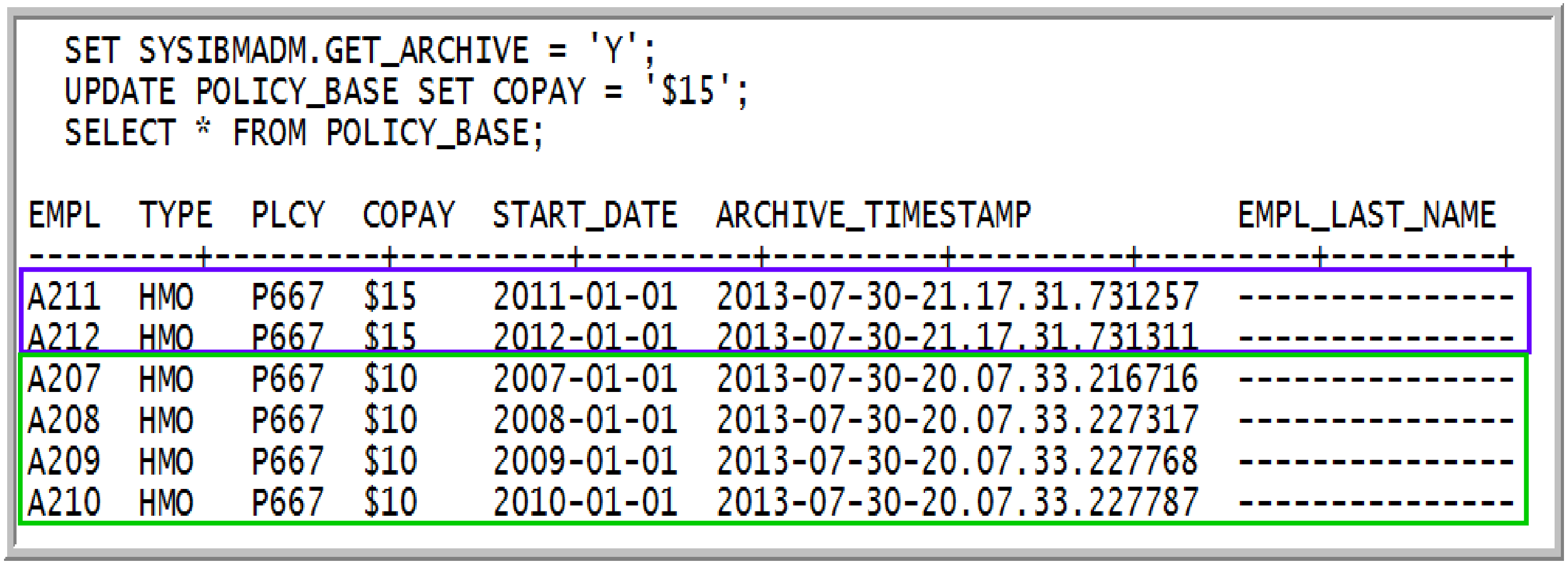
- To select data from only the base table
- Set **GET_ARCHIVE** global variable to ‘N’ before the select statement



```
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 ...

- Searched UPDATEs will only update base table rows whether the GET_ARCHIVE is set to Y or N
- In the first example, we set the GET_ARCHIVE to ‘Y’ so that the SELECT will retrieve rows from both
  - The Base and Archive tables, and you can see that only the base 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 Restrictions

- Restrictions
  - LOAD utility will not allow REPLACE option if the table is archived-enabled
  - Columns can not be ALTERed, RENAMEd or DROPped in either a base or archive table
  - If MOVE_TO_ARCHIVE = ‘Y’, INSERT, UPDATE & MERGE against the base
  - PERIOD can not be added to a base or archive table
  - A foreign key can not be defined on an archive table
  - ROTATE partitions
  - MQT
  - Row Permissions and Column Masks
  - CLONE table
  - A data change statement cannot reference temporal table in same statement as reference to archive table
  - Positioned UPDATEs and Positioned/Searched DELETEs against base tables not allowed when
    - GET_ARCHIVE = ‘Y’ and
    - Bound ARCHIVESENSITIVE YES
  - If the target of the INSERT is a view that is defined with the WITH CHECK OPTION, the definition of the view must not reference an archive-enabled table
  - If you use a DYNAMIC SENSITIVE SCROLLable cursor, you can run with GET_ARCHIVE=‘N’,

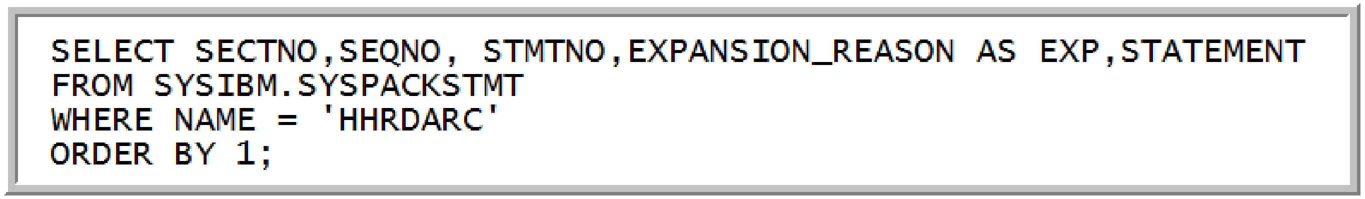
    but if you run with GET_ARCHIVE=‘Y’ you get -525 SQL error

Archive Transparency ...

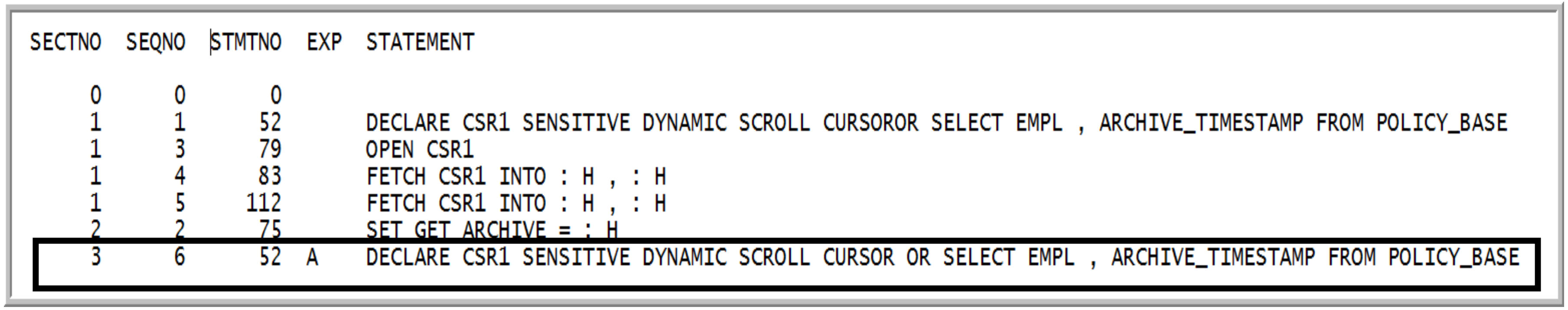
- When the BIND is performed with an Archive Enabled Table
  - We create two sections in the package when **ARCHIVESENSITIVE** is YES
    - First section - Base table only
    - Second section - Base table and archive table UNION’ed ALL together
  - **ARCHIVESENSITIVE** only refers to **GET_ARCHIVE** 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’



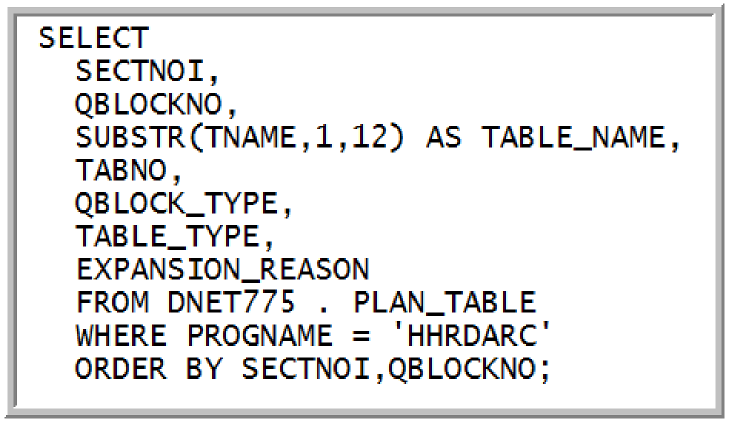
```
SELECT SECTNO,SEQNO, STMTNO,EXPANSION_REASON AS EXP,STATEMENT
FROM SYSIBM.SYSPACKSTMT
WHERE NAME = 'HHRDARC'
ORDER BY 1;
```



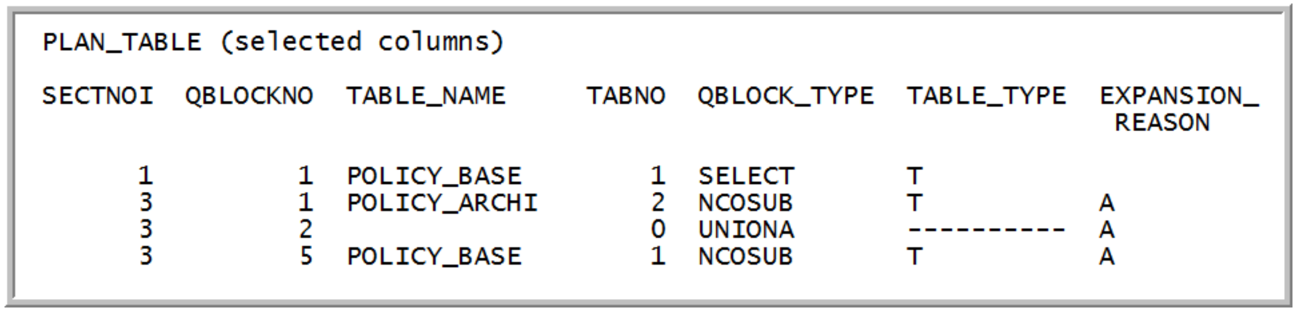
```
SECTNO  SEQNO  STMTNO  EXP  STATEMENT

     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 EXPLAIN

- EXPANSION_REASON added to the following tables
  - **DSN_COLDIST_TABLE**
  - **DSN_DETCOST_TABLE**
  - **DSN_FILTER_TABLE**
  - **DSN_FUNCTION_TABLE**
  - **DSN_KEYTGTDIST_TABLE**
  - **DSN_PGRANGE_TABLE**
  - **DSN_PGROUP_TABLE**
  - **DSN_PREDICATE_SELECTIVITY**
  - **DSN_PREDICAT_TABLE**
  - **DSN_PTASK_TABLE**
  - **DSN_QUERYINFO_TABLE**
  - **DSN_QUERY_TABLE**
  - **DSN_SORTKEY_TABLE**
  - **DSN_SORT_TABLE**
  - **DSN_STATEMENT_CACHE_TABLE**
  - **DSN_STATEMNT_TABLE**
  - **DSN_STRUCT_TABLE**
  - **DSN_VIEWREF_TABLE**
  - **PLAN_TABLE**

Archive Transparency Comparison

- Archive Transparency
  - Works one 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

Thank You