Baltimore Washington DB2 User’s Group

DB2 10 for z/OS: Temporal Tables (aka Time Travel Query)

Charles Lewis
DB2 Advisor
IBM US East Extended Team
System z SW Technical Professional
lewisc@us.ibm.com
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 a 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
  - 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
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

To enable SYSTEM TIME you then alter the table:

```
CREATE TABLE POLICY
```

```
ALTER TABLE POLICY
ADD VERSIONING USE HISTORY TABLE POLICYHISTORY;
```

**Note:**

If you need to make changes to the table, you will need to alter the table to drop the versioning, make the changes, and then alter the table to add versioning.
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 current System Start Time and 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 current time (System Start Time of the associated base table row for an update)

Delete Row A @ T4
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 row 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

![Diagram showing update and delete scenarios with business time rows.](image-url)
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

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
</tr>
</thead>
<tbody>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14.745082721000</td>
<td>9999-12-31-24.00.00.000000000000</td>
</tr>
</tbody>
</table>

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 1/1/2011  
Step 2 – 09/24/2010 Update all P667 policies to a copay of $15 beginning 01/01/2011

#### Original Row

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
</tr>
</thead>
<tbody>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14.745082721000</td>
<td>9999-12-31-24.00.00.0000000000</td>
</tr>
</tbody>
</table>

#### Policy Table

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

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
</tr>
</thead>
<tbody>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22.50672497000</td>
<td>9999-12-31-24.00.00.0000000000</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22.50672497000</td>
<td>9999-12-31-24.00.00.0000000000</td>
</tr>
</tbody>
</table>

#### Policy History table

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
</tr>
</thead>
</table>
System Time / Point In Time...

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF '2010-09-22-00.00.00.0000000000000000';
```

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>
System Time / Point In Time...

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
System Time / Point In Time …

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF '2010-09-25-00.00.00.0000000000000000';
```

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>
System Time / Point In Time

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 9999-12-31-24.00.00...

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>
### System Time / Range ...

```sql
SELECT * FROM POLICY FOR SYSTEM_TIME
FROM '2010-09-22-00.00.00.0000000000000000'
TO '2010-09-24-00.00.00.0000000000000000';
```

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QBLOCKNO</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TBNAME</th>
<th>TABNO</th>
<th>ACCESTYPE</th>
<th>PREFETCH</th>
<th>OBLOCK_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>DBAO15</td>
<td>POLICYHISTORY</td>
<td>2</td>
<td>R</td>
<td>5</td>
<td>NCOSUB</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>UNIONA</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>DBAO15</td>
<td>POLICY</td>
<td>1</td>
<td>R</td>
<td>5</td>
<td>NCOSUB</td>
</tr>
</tbody>
</table>
For this query, we look for rows where:
System Start Time (SYS_BEG) <= 9/25 AND
System End Time (SYS_END) > 9/22
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

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
<tr>
<td>C054</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>9999-12-31</td>
<td>2010-09-21-21.50.14</td>
<td>2010-09-24-17.33.22</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2011-01-01</td>
<td>2010-09-24-17.33.22</td>
<td>2010-09-24-19.44.47</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$15</td>
<td>2011-01-01</td>
<td>9999-12-31</td>
<td>2010-09-24-17.33.22</td>
<td>2010-09-24-19.44.47</td>
<td>HISTORY</td>
</tr>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2010-09-24</td>
<td>2010-09-24-19.44.47</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>EMPL</th>
<th>TYPE</th>
<th>PLCY</th>
<th>COPAY</th>
<th>EFF_BEG</th>
<th>EFF_END</th>
<th>SYS_BEG</th>
<th>SYS_END</th>
<th>Which Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO54</td>
<td>HMO</td>
<td>P667</td>
<td>$10</td>
<td>2004-01-01</td>
<td>2010-09-24</td>
<td>2010-09-24-19.44.47</td>
<td>9999-12-31-24.00.00</td>
<td>BASE</td>
</tr>
</tbody>
</table>

**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

- Cannot ALTER the schema while versioning

- 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

- System Time can be ALTERed (eg. ADDed) to an existing table
  - See Information Center topic:
- QUIESCE of the Base or History
  - Will cause a quiesce against all tables in the versioning relationship, including auxiliary spaces
System Period Versioning Information

- 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 (eg ADDed) to an existing table

- 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

- FOR PORTION OF BUSINESS_TIME must specify valid Date/Time values in the FROM and TO
Comparing DB2 Temporal to RYO implementation

- Shortens time to deployment due to significant gain in productivity
- Easier and less expensive to maintain because less user code needed to solve client’s business problem
- Cost savings due to reduced CPU usage because DB2 function performs better than RYO equivalent
- Temporal model IBM has standardized through ANSI and ISO
- Consistent model enabling interoperability of client’s temporal application written to zDB2 temporal
- Some forms of business integrity supported in DB2 for z/OS removes burden from having to write in the application
Performance Study: DB2 provided system time support vs. RYO trigger solution

No History Generation
1. Following Txn mix run against in-house TPC-H database
   70% read (SELECT)
   30% write (10% INSERT + 20% UPDATE/DELETE)

RYO History Generation
2. Same Txn mix run against in-house TPC-H database enhanced with
   Update and Delete triggers creating historical rows
   Historical Rows created in separate History table

DB2 History Generation
- Same Txn mix run against enhanced TPC-H schema supporting
  SYSTEM TIME period, transaction ID,
  History table and versioning

20+% CPU reduction compared to RYO
Performance of DB2 for z/OS Business Time

UPDATE policy SET PREMIUM = 888, temp_begin = '2010-02-13', temp_end = '2010-02-15' WHERE policy_id = 1234 AND (DATE('2010-02-13') >= BUS_BEGIN AND DATE('2010-02-13') < BUS_END) OR (DATE('2010-02-15') > BUS_BEGIN AND DATE('2010-02-15') <= BUS_END) OR (DATE('2010-02-13') <= BUS_BEGIN AND DATE('2010-02-15') >= BUS_END)

UPDATE temporal_policy FOR PORTION OF BUSINESS_TIME FROM DATE('2010-02-13') TO DATE('2010-02-15') SET PREMIUM = 888 WHERE policy_id = 1234

In DB2 code path reduction

- 2-row split: -57.23%
- 3-row split: -68.44%
Customers love it

The new temporal functionality in DB2 10 for z/OS will allow us to **drastically simplify** our date-related queries. In addition, we’ll be able to **reduce our storage costs** by using cheaper storage for inactive rows and reduce our processing cost by having DB2 handle data movement more efficiently than the custom code we’ve written to do the same work in the past.

_Large Insurance Company - DB2 10 Beta Customer_

_"We are really thrilled about “Temporal Data” feature – this feature has the potential to **significantly reduce overheads**. We have estimated that **80% of our existing temporal applications** could have used “the DB2 10 temporal features” instead of application code - this feature will **drastically save developer time, testing time** – and even more importantly make applications easier to understand so **improve business efficiency and effectiveness”_
Thank You