Temporal Data

Angelo Sironi
Sironi IT Consulting s.n.c.
angelo_sironi@fastwebnet.it

These charts are part of a larger presentation on the subject of Temporal Data support in DB2 10 for z/OS and all the additional SQL and application-related enhancements of DB2 10. The material is suited for a 1.5 – 2.0 days of education for application programmers and Data Base Administrators.
Table of Content

- Time-varying data
  - Type
  - Examples
- Issues and complexities
- Temporal Data support in ISO/ANSI SQL:2011
- Temporal Data support in DB2 10 for z/OS
  - System-time period support
  - Business-time period support
  - Bi-temporal support

Time-varying data?

- A Phone Call
  - Occurred (i.e. began) on some date and time
  - Lasted for so much time
  - Was registered by the system on some timestamp
  - Never updated?
    - Not time-varying!
  - Values replaced?
    - Likely not time-varying!
  - Values updated?
    - Maybe ... time-varying...

<table>
<thead>
<tr>
<th>PhoneID</th>
<th>Call_time</th>
<th>Duration</th>
<th>Cost</th>
<th>Store_time_start</th>
<th>Store_time_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>02-123-678</td>
<td>2012-01-10 10:00.00</td>
<td>124</td>
<td>3.2</td>
<td>2012-02-01 03:02.10</td>
<td>2012-02-01 03:02.10</td>
</tr>
<tr>
<td>02-123-678</td>
<td>2012-01-10 10:00.00</td>
<td>124</td>
<td>2.3</td>
<td>2012-02-01 03:02.10</td>
<td>9999-12-31 24:00.00</td>
</tr>
</tbody>
</table>
### Time-varying data: Additional Examples

- **Products price history**
  - Valid on specific time interval
  - Valid on specific time interval

<table>
<thead>
<tr>
<th>PROD ID</th>
<th>PRICE</th>
<th>VAT</th>
<th>VALID FROM DATE</th>
<th>VALID TO DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>325</td>
<td>21%</td>
<td>02-gen-12</td>
<td>29-feb-12</td>
</tr>
<tr>
<td>P1</td>
<td>295</td>
<td>21%</td>
<td>29-feb-12</td>
<td>31-dic-99</td>
</tr>
</tbody>
</table>

- **Customer Phone Line(s) (Telco DW 1999)**
  - Valid on some time interval
    - Modifications stored on the system at some later time

<table>
<thead>
<tr>
<th>PHONE_NR</th>
<th>CUST_ID</th>
<th>BUS_STA_DT</th>
<th>BUS_END_DT</th>
<th>SYS_STA_DT</th>
<th>SYS_END_DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>02-444-111</td>
<td>C. Rossi</td>
<td>01.01.2000</td>
<td>31.12.9999</td>
<td>04.01.2000</td>
<td>31.12.9999</td>
</tr>
</tbody>
</table>

### Time in the Real World and System Time

- **Valid Time** (aka Application Time (SQL:2011) or Business Time (DB2))
  - Specifies when the facts are true with respect to the Real World
  - Useful for data that change over time and time information is relevant to applications and users
  - Time or Time Period

- **Transaction Time** (aka System Time (DB2) or System–versioned tables (SQL:2011))
  - The time when the System becomes aware of the fact
  - The information is safely stored on a system file/DB

- **Business Time vs. System Time**
  - Mr. Brown lives in Rome since 23.11.1997 (Business Time)
  - Info supplied via mail on 12.12.2002 (Business Time)
  - Info registered by the System on 23.12.2002 (System Time)
Time-varying data is pervasive

- It has been estimated that one of every 50 lines of database application code involves a date or time value
  - Data Warehouse data are, by definition, time-varying
  - Becoming more and more popular also for OLTP applications
- Often the time-oriented nature of the data is what lends its value
- Overlaying simple concepts, such as duplicate prevention, on time-varying data can be surprisingly subtle and complex

Cfr. [2]

Bi-temporal Relations: Entity Integrity

How do we identify duplicates?

<table>
<thead>
<tr>
<th>PHONE_NR</th>
<th>CUST_ID</th>
<th>BUS_STA_DT</th>
<th>BUS_END_DT</th>
<th>SYS_STA_DT</th>
<th>SYS_END_DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>02-444-111</td>
<td>C. Rossi</td>
<td>01.01.2000</td>
<td>31.12.9999</td>
<td>01.01.2000</td>
<td>24.08.2001</td>
</tr>
<tr>
<td>02-444-111</td>
<td>C. Rossi</td>
<td>01.01.2000</td>
<td>20.08.2001</td>
<td>24.08.2001</td>
<td>31.12.9999</td>
</tr>
</tbody>
</table>

- Duplicates exist when
  - Same Natural Key (Phone Nr) AND ...
  - Overlapping Business Time periods AND ...
  - Overlapping System Time periods
Preliminary Findings – 1

- Temporal Models enforce Temporal Functional Dependencies
- If attributes depend on time, then time is Business Time or System Time

Preliminary Findings – 2

Bi-Temporal support helpful when ...

- Need to track history of changes ... AND ...
- Changes occur in the real world long enough before (or after) they are captured by the system ...
  AND ...
- Need to avoid loosing any piece of information
  - Re-build past situations
  - Justify query result differences over time
Inserting Records into a “Bi–temporal Relation”

- **Inserting a new entity record**
  - Usual SQL Insert with proper timings
    - **Business Begin Date**: Past, Present or Future date/time when the “fact” described by the record begins being true. Current Date/Timestamp for Current Insertion.
    - **Business End Date**: (Past, Future or) Now, i.e. date/time when the “fact” described by the record becomes false. Usually Null or preferably “end of era” (es. “31.12.9999”), if undefined.
    - **System Begin Date**: Current Date (or Current Timestamp)
    - **System End Date**: Usually Null or preferably “end of era” (es. “31.12.9999”)
Updating Records in a “Bi-temporal Relation” – 1

Modifications: 5+ cases

1. Current State
   State Modification
   Resulting States

2. Current State
   State Modification
   Resulting States

3. Current State
   State Modification
   Resulting States

Updating Records in a “Bi-temporal Relation” – 2

Sequenced Modifications (Update/Inserts): 5+ cases

4. Current State
   State Modification
   Resulting State

5. Current States
   State Modification
   Resulting States

- + additional extensions of first 4 cases to multiple states
- Similarly for Deletes
Updating Bi-Temporal DM: considerations

- **Tracking a change of an existing record requires (for the simplest case)**
  - One SQL update operation
  - Two SQL Insert operations

- **More complex situations exist (e.g., delayed and out-of-sequence)**

- **Additional complexities**
  - Entity & Referential Integrity
    - When using Surrogate Keys
    - When not using Surrogate Keys
  - Vacuuming
    - Removal of obsolete data

(Aside) Conclusions

- Time-variant data support is an inherent characteristic of a Data Warehouse and many OLTP applications
- Managing time-varying data can be surprisingly subtle and complex
  - Data modeling
  - Application and query design & code
- It’s very important to choose the right level of support, avoiding unnecessary complexities, when they don’t provide real business value
- Be aware!
Temporal Data Support in SQL:2011

SQL Standard

- INCITS DM32.2 (formerly X3H2) is responsible for SQL standard in US
  - International Committee for Information Technology Standards is an ANSI (American National Standards Institute) accredited standards producing organization
- ISO/IEC JTC 1/SC 32 Data Management and Interchange/WG 3 committee is responsible for SQL Standard Internationally
- Typically 3 to 5 year cycle for each new version of the standard
- SQL is a multi-part standard which currently has 9 parts
  - The highest part number is currently 14 (parts 5, 6, 7, 8 & 12) were terminated
  - Part 2: SQL/Foundation is the SQL language specification (biggest/most important part)
- 7 Versions of Standard SQL
Temporal Feature 1\textsuperscript{st} Attempt: 1995–2001

- X3H2 and WG 3 approved work on a new part of SQL standard called SQL/Temporal (1995)
- US proposal largely based on the pioneering work of Prof. Rick Snodgrass of Univ. of Arizona
  - The US proposal proved to be controversial at ISO
- UK competing proposal based on the work of Prof. Nikos Lorentzos of Univ. Athens, Greece
  - US disagreed with the ISO comments on the US proposal
  - The US also did not see the need for the UK proposal
- ANSI and ISO decided to defer further work on SQL/Temporal until SQL:99 was published
  - After the publication of SQL:99, neither US nor UK brought in any new proposals to resolve the differences
- Because of inactivity, SQL/Temporal canceled in 2001

Temporal Feature 2\textsuperscript{nd} Attempt: 2008–2011

- INCITS DM32.2 and ISO/IEC JTC1 SC32 WG3 accepted a proposal on "system--versioned tables" (T180)
  - Not a new part of the standard
  - Temporal extensions added to SQL/Foundation
- Another temporal feature was added in 2010 in the form of "application–time period tables" (T181)
- Both "system–versioned tables" and "application–time period tables" are now part of the new version of the SQL standard (SQL:2011)
  - The proposal is largely inspired by the original one
  - The syntax is quite different
Temporal Data

- **Temporal data is data which changes over time**
  - An account interest rate changes over time
    - Referred to as valid time dimension, real world perspective or business perspective
    - SQL:2011 calls this *application time*
  - The value we have in a database for a company’s credit rating changes over time
    - This may be different than application time due to timing differences (e.g. delays or advance registration)
    - This may have nothing to do with the application time perspective (for example corrections)
    - Referred to as transaction time dimension or database perspective
    - SQL:2011 calls this *system time* (or *system versioning*)

- **Bi-temporal data is data which changes over two dimensions of time independently**
  - SQL:2011 calls this *application time with system versioning*

---

**T181 – Application–time period tables (1 of 2)**

- **Application–time period tables are tables that contain a PERIOD clause (newly–introduced) with an user–defined period name**
  - Currently restricted to temporal periods only; may be relaxed in the future

- **Application–time period tables must contain two additional columns to store**
  - The start time of the period associated with the row
  - The end time of the period
  - Values of both start and end columns are set by the users

- **Additional syntax is provided for users to specify primary key/unique constraints that ensure no two rows with the same key value have overlapping periods**
T181 – Application–time period tables (2 of 2)

- Additional syntax is provided for users to specify referential constraints
  - the period of every child row must be completely contained in the period of exactly one parent row or in the combined period of two or more consecutive parent rows
- Queries, inserts, updates and deletes on application–time period tables behave exactly like queries, inserts, updates and deletes on regular tables
- Additional syntax is provided for updating and deleting over an arbitrary period of time

Creating an application–time period table

```sql
CREATE TABLE employees
(emp_name VARCHAR(50) NOT NULL,
 dept_id VARCHAR(10),
 start_date DATE NOT NULL,
 end_date DATE NOT NULL,
 PERIOD FOR emp_period (start_date,
 end_date),
 PRIMARY KEY (emp_name, emp_period
 WITHOUT OVERLAPS),
 FOREIGN KEY (dept_id, PERIOD emp_period)
 REFERENCES departments (dept_id,
 PERIOD dept_period));
```
T180 – System-versioned tables

- System-versioned tables are tables that contain a PERIOD clause with a pre-defined period name (SYSTEM_TIME) and specify WITH SYSTEM VERSIONING.
- System-versioned tables must contain two additional columns:
  - One to store the start time of the SYSTEM_TIME period
  - One to store the end time of the SYSTEM_TIME period
  - Values of both start and end columns are set by the system
  - Users are not allowed to supply values for these columns
- Unlike regular tables, system-versioned tables preserve the old versions of rows as the table is updated.
- Rows whose periods intersect the current time are called current system rows:
  - Only current system rows can be updated or deleted
  - All constraints are enforced on current system rows only
- All others are called historical system rows

Creating a System-versioned tables

CREATE TABLE employees
    (emp_name VARCHAR(50) NOT NULL,
     dept_id VARCHAR(10),
     system_start TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
                    ALWAYS AS ROW END,
     system_end TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
PERIOD FOR SYSTEM_TIME (system_start,
                        system_end),
     PRIMARY KEY (emp_name),
     FOREIGN KEY (dept_id) REFERENCES departments (dept_id);
) WITH SYSTEM VERSIONING;
Four Different Types of Tables

<table>
<thead>
<tr>
<th>SYSTEM TIME PERIOD</th>
<th>APPLICATION TIME PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>Non Temporal</td>
</tr>
<tr>
<td>YES</td>
<td>System Versioned Table</td>
</tr>
</tbody>
</table>

**Notes:**
1. Temporal support when **time period(s)** involved
   - i.e. value(s) change over time
   - Each period limited by two timestamps
2. Basic implementation is Row–based

Temporal Data Support in DB2 10
Temporal Data in DB2

- **Table-level specification to control data management based upon time**
- **Two notions of time**
  - **Business time**
    - Notes the occurrence of a real world event or business decision related to the present time, the past or the future
    - Useful for tracking business-relevant events over time
  - **System time**
    - Notes the occurrence of a database change
    - Useful for auditing & compliance
- **New syntax in FROM clause to specify a time criteria**
- **Application logic greatly simplified**

Temporal Data Support: System Time

- **Requirements addressed**
  - Managing multiple versions of relevant data
  - Auditing
- **Before DB2 V10**
  - Supported by adding one or two Date / Timestamp columns, like e.g. LOAD_TIME, INSERT_TIME, UPDATE_TIME or similar names
  - Usually, only provides the latest values of table columns
    - Record before-image lost when updating in-place
- **With DB2 10**
  - System Time supported by
    - New CREATE/ALTER TABLE options
    - New DML constructs
  - Before-image preserved by DB2
DB2 10: System Time Support

- Define table(s) with an associated System Time Period & Transaction timestamp
  - System Period indicates starting & ending points of a time interval
    - DB2 time intervals are (inclusive, exclusive)
    - Timestamp(12) columns used
    - GENERATED ALWAYS
  - Transaction timestamp used internally by DB2
    - User cannot modify values of Period & Trx. timestamp

- Define, for each temporal table, an associated History Table
  - Old versions of rows saved into it
  - Applications and users don’t need to be aware
    - Transparent access managed by DB2 as required in support of temporal queries

System Time Support: Example

- Create a System–period temporal table
  ```
  CREATE TABLE SYS_INTEREST_RATE (TYPE CHAR(6) NOT NULL,
    CUST_SEGMENT CHAR(12) NOT NULL,
    INTEREST_RATE DECIMAL(5,2) NOT NULL,
    SYS_BEGIN TIMESTAMP(12) NOT NULL
    GENERATED AS ROW BEGIN,
    SYS_END   TIMESTAMP(12) NOT NULL
    GENERATED AS ROW END,
    PERIOD    SYSTEM_TIME(SYS_BEGIN, SYS_END)
    GENERATED ALWAYS AS TRANSACTION START ID)
  );
  ```

- Create the History table
  ```
  CREATE TABLE SYS_INTEREST_RATE_HISTORY LIKE SYS_INTEREST_RATE;
  ```

- Enable system–period data versioning
  ```
  ALTER TABLE SYS_INTEREST_RATE ADD VERSIONING
  USE HISTORY TABLE SYS_INTEREST_RATE_HISTORY;
  ```
System Time Support: Example (cont.)

- **Insert a couple of rows**
  
  ```
  INSERT INTO SYS_INTEREST_RATE
  ("TYPE", CUST_SEGMENT, INTEREST_RATE)
  VALUES ('DEBIT', 'VIP', 5.32);
  INSERT INTO SYS_INTEREST_RATE
  ("TYPE", CUST_SEGMENT, INTEREST_RATE)
  VALUES ('CREDIT', 'VIP', 1.03);
  ```

- **SYS_INTEREST_RATE content**
  
<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>SYS BEGIN</th>
<th>SYS END</th>
<th>TRANS ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-09-21-15:10:00.549719</td>
<td>9999-12-31-24:00.00.000000</td>
<td>2011-09-21-15:10:00.549719</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-09-21-15:10:10.425539</td>
<td>9999-12-31-24:00.00.000000</td>
<td>2011-09-21-15:10:10.425539</td>
</tr>
</tbody>
</table>

- **SYS_INTEREST_RATE_HISTORY is empty**

  **Note:** APAR PM31314 has modified the row-end maximum value to '9999-12-30-00.00.00.000000' in accordance with java.sql.Timestamp which is based on java.util.Date (where hours have values from 0 to 23).

---

System Time Support: Example (cont.)

- **Table content BEFORE (current & history)**
  
<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>SYS BEGIN</th>
<th>SYS END</th>
<th>TRANS ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-09-21-15:10:00.549719</td>
<td>9999-12-31-24:00.00.000000</td>
<td>2011-09-21-15:10:00.549719</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-09-21-15:10:10.425539</td>
<td>9999-12-31-24:00.00.000000</td>
<td>2011-09-21-15:10:10.425539</td>
</tr>
</tbody>
</table>

- **Let's UPDATE the interest rate**
  
  ```
  UPDATE SYS_INTEREST_RATE
  SET INTEREST_RATE = INTEREST_RATE + 0.12
  WHERE TYPE = 'CREDIT'
  AND CUST_SEGMENT = 'VIP';
  ```

- **Table content AFTER (current & history)**
  
<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>SYS BEGIN</th>
<th>SYS END</th>
<th>TRANS ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.44</td>
<td>2011-09-21-15:10:00.549719</td>
<td>9999-12-31-24:00.00.000000</td>
<td>2011-09-21-15:10:00.549719</td>
</tr>
</tbody>
</table>

---
System Time Support: Example (cont.)

Table content BEFORE (current & history)

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST_SEGMENT</th>
<th>INTEREST_RATE</th>
<th>SYS_BEGIN</th>
<th>SYS_END</th>
<th>TRANS_ID</th>
</tr>
</thead>
</table>

Let's DELETE some record(s)

DELETE FROM SYS_INTEREST_RATE
WHERE TYPE = 'DEBIT'
AND CUST_SEGMENT = 'VIP';

Table content AFTER (current & history)

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST_SEGMENT</th>
<th>INTEREST_RATE</th>
<th>SYS_BEGIN</th>
<th>SYS_END</th>
<th>TRANS_ID</th>
</tr>
</thead>
</table>

Let's change the way we query...

What were the interest rates for VIP customers at some time?

SELECT *
FROM SYS_INTEREST_RATE
FOR SYSTEM_TIME AS OF
TIMESTAMP '2011-09-21-16:00.00.000000'
WHERE CUST_SEGMENT = 'VIP'

Query result set

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST_SEGMENT</th>
<th>INTEREST_RATE</th>
<th>SYS_BEGIN</th>
<th>SYS_END</th>
<th>TRANS_ID</th>
</tr>
</thead>
</table>
Let’s change the way we query...
- What were the interest rates for VIP customers at some time?

```sql
SELECT * FROM SYS_INTEREST_RATE
FOR SYSTEM_TIME FROM '2011-09-21-00.00.00.000000'
TO CURRENT_TIMESTAMP
WHERE CUST_SEGMENT = 'VIP';
```

**Query result set**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST_SEGMENT</th>
<th>INTEREST_RATE</th>
<th>SYS_BEGIN</th>
<th>SYS_END</th>
<th>TRANS_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-09-21-15:10:00.549719</td>
<td>2011-09-21-15:10:00.549719</td>
<td>2011-09-21-15:10:00.549719</td>
</tr>
</tbody>
</table>

**Create a Business–period temporal table**

```sql
CREATE TABLE BUS_INTEREST_RATE
(
    TYPE          CHAR(6) NOT NULL,
    CUST_SEGMENT  CHAR(12) NOT NULL,
    INTEREST_RATE DECIMAL(5,2) NOT NULL,
    BUS_BEGIN_DATE DATE NOT NULL,
    BUS_END_DATE   DATE NOT NULL,
    PERIOD        BUSINESS_TIME,
    PRIMARY KEY (TYPE, CUST_SEGMENT, BUSINESS_TIME WITHOUT OVERLAPS)
);  
```

**Primary Key definition**

<table>
<thead>
<tr>
<th>IXNAME</th>
<th>COLNAME</th>
<th>COLSEQ</th>
<th>ORDERING</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUS_INT_E_E_NTH</td>
<td>TYPE</td>
<td>1</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CUST_SEGMENT</td>
<td>2</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BUS_END_DATE</td>
<td>3</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>BUS_BEGIN_DATE</td>
<td>4</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>
Business Time Support: Example (cont.)

- **Insert a couple of rows**

  ```sql
  INSERT INTO BUS_INTEREST_RATE
  ("TYPE", CUST_SEGMENT, INTEREST_RATE,
   BUS_BEGIN_DATE, BUS_END_DATE)
  VALUES ('DEBIT', 'VIP', 5.32,
   '18.10.2011', '31.12.9999') ;
  
  INSERT INTO BUS_INTEREST_RATE
  ("TYPE", CUST_SEGMENT, INTEREST_RATE,
   BUS_BEGIN_DATE, BUS_END)
  VALUES ('CREDIT', 'VIP', 1.03,
   '18.10.2011', '31.12.9999') ;
  ```

- **BUS_INTEREST_RATE content**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST_SEGMENT</th>
<th>INTEREST_RATE</th>
<th>BUS_BEGIN_DATE</th>
<th>BUS_END_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
</tbody>
</table>

- **Let's UPDATE the interest rate**

  ```sql
  UPDATE BUS_INTEREST_RATE
  FOR PORTION OF BUSINESS_TIME FROM DATE '11.10.2012'
  TO DATE '31.12.9999'
  SET INTEREST_RATE = INTEREST_RATE + 0.12
  WHERE TYPE = 'CREDIT' ;
  ```

- **Table content AFTER**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST_SEGMENT</th>
<th>INTEREST_RATE</th>
<th>BUS_BEGIN_DATE</th>
<th>BUS_END_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-10-11</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-18</td>
<td>2012-10-11</td>
</tr>
</tbody>
</table>
BUSINESS Time Support: Example (cont.)

- **Table content BEFORE**
  
<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>BUS BEGIN DATE</th>
<th>BUS END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-10-11</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-18</td>
<td>2012-10-11</td>
</tr>
</tbody>
</table>

- **Let’s UPDATE again some record(s)**

  ```
  UPDATE BUS_INTEREST_RATE
  FOR PORTION OF BUSINESS_TIME
  FROM DATE '11.08.2012' TO DATE '31.12.2012'
  SET INTEREST_RATE = 1.27
  WHERE TYPE = 'CREDIT';
  ```

- **Table content AFTER**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>BUS BEGIN DATE</th>
<th>BUS END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-18</td>
<td>2012-08-11</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-08-11</td>
<td>2012-10-11</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.27</td>
<td>2012-10-11</td>
<td>2012-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-12-31</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
</tbody>
</table>

---

Business Time Support: Example (cont.)

- **Table content**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>BUS BEGIN DATE</th>
<th>BUS END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-18</td>
<td>2012-08-11</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-08-11</td>
<td>2012-10-11</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.27</td>
<td>2012-10-11</td>
<td>2012-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-12-31</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
</tbody>
</table>

- **Let’s change the way we query...**
  
  - What the interest rate(s) for VIP customers at some time?

  ```
  SELECT *
  FROM BUS_INTEREST_RATE
  FOR BUSINESS_TIME AS OF DATE '2011-12-21'
  WHERE CUST_SEGMENT = 'VIP';
  ```

- **Query result set**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>BUS BEGIN DATE</th>
<th>BUS END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-18</td>
<td>2012-08-11</td>
</tr>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
</tbody>
</table>
BUSINES Time Support: Example (cont.)

- **Table content BEFORE**
  
<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>BUS BEGIN DATE</th>
<th>BUS END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-10</td>
<td>2012-08-11</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-08-11</td>
<td>2012-10-11</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.27</td>
<td>2012-10-11</td>
<td>2012-12-31</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-12-31</td>
<td>9999-12-31</td>
</tr>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.12</td>
<td>2011-10-18</td>
<td>9999-12-31</td>
</tr>
</tbody>
</table>

- **Let's DELETE some record(s)**
  
  ```sql
  DELETE FROM BUS_INTEREST_RATE
  FOR PORTION OF BUSINESS_TIME
  FROM DATE '11.07.2012' AND DATE '11.11.2012'
  WHERE TYPE = 'CREDIT';
  ```

- **Table content AFTER**
  
<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST SEGMENT</th>
<th>INTEREST RATE</th>
<th>BUS BEGIN DATE</th>
<th>BUS END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-10</td>
<td>2012-07-1</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.27</td>
<td>2012-10-11</td>
<td>2012-12-3</td>
</tr>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>1.15</td>
<td>2012-12-31</td>
<td>9999-12-3</td>
</tr>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>5.12</td>
<td>2011-10-18</td>
<td>9999-12-3</td>
</tr>
</tbody>
</table>

Bi–Temporal Support: Example

- **Create a System–period temporal table**
  
  ```sql
  CREATE TABLE INTEREST_RATE
  ( TYPE CHAR(6) NOT NULL,
    CUSTSEGMENT CHAR(12) NOT NULL,
    INTEREST_RATE DECIMAL(5,2) NOT NULL,
    BUS_START_DATE DATE NOT NULL,
    BUS_END_DATE   DATE NOT NULL,
    PERIOD BUSINESS_TIME ( BUS_START_DATE,
                           BUS_END_DATE )
    , SYS_BEGIN TIMESTAMP(12) NOT NULL
    GENERATED AS ROW BEGIN
    , SYS_END TIMESTAMP(12) NOT NULL
    GENERATED AS ROW END
    , PERIOD SYSTEM_TIME( SYS_BEGIN, SYS_END )
    , TRANS_ID TIMESTAMP(12) NOT NULL
    GENERATED ALWAYS AS TRANSACTION START ID )
  ```
Bi-Temporal Support: Example (cont.)

- **Create the History table**

```sql
CREATE TABLE INTEREST_RATE_HISTORY
    ( TYPE CHAR(6) NOT NULL ,
      CUST_SEGMENT CHAR(12) NOT NULL ,
      INTEREST_RATE DECIMAL(5,2) NOT NULL ,
      BUS_START_DATE DATE NOT NULL ,
      BUS_END_DATE DATE NOT NULL ,
      Sys_BEGIN TIMESTAMP(12) NOT NULL ,
      SYS_END TIMESTAMP(12) NOT NULL ,
      TRANS_ID TIMESTAMP(12) NOT NULL )
```

- **or, more simply ..**

```sql
CREATE TABLE INTEREST_RATE_HISTORY
LIKE INTEREST_RATE;
```

- **Enable system-period data versioning**

```sql
ALTER TABLE INTEREST_RATE
ADD VERSIONING USE HISTORY TABLE INTEREST_RATE_HISTORY;
```

- **Define Business-time without overlaps**

```sql
ALTER TABLE INTEREST_RATE
ADD UNIQUE (TYPE, CUST_SEGMENT,
   BUSINESS_TIME WITHOUT OVERLAPS);
```
Bi-Temporal Support: Example (cont.)

- **Enable system-period data versioning**
  ```sql
  ALTER TABLE INTEREST_RATE
  ADD VERSIONING
  USE HISTORY TABLE INTEREST_RATE_HISTORY;
  ```

- **Define Business-time without overlaps**
  ```sql
  ALTER TABLE INTEREST_RATE
  ADD UNIQUE
  (TYPE, CUST_SEGMENT, BUSINESS_TIME WITHOUT OVERLAPS);
  ```

- **Insert a couple of rows**
  ```sql
  INSERT INTO PRIM93.INTEREST_RATE ("TYPE", CUST_SEGMENT, INTEREST_RATE, BUS_START_DATE, BUS_END_DATE)
  VALUES ('DEBIT', 'VIP', 5.32, DATE '11.08.2011', DATE '31.12.9999')
  ```

- **Querying Bi-Temporal**

  ```sql
  SELECT *
  FROM INTEREST_RATE
  FOR BUSINESS_TIME AS OF '31.08.2011'
  ORDER BY 1;
  ```

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CUST_SEGMENT</th>
<th>INTEREST_RATE</th>
<th>BUS_BEGIN_DATE</th>
<th>BUS_END_DATE</th>
<th>SYS_BEGIN</th>
<th>SYS_END</th>
<th>TRANS_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDIT</td>
<td>VIP</td>
<td>5.32</td>
<td>2011-10-11</td>
<td>9999-12-31</td>
<td>2011-10-11</td>
<td>9999-12-31</td>
<td>2011-10-11</td>
</tr>
<tr>
<td>DEBIT</td>
<td>VIP</td>
<td>1.03</td>
<td>2011-10-11</td>
<td>9999-12-31</td>
<td>2011-10-11</td>
<td>9999-12-31</td>
<td>2011-10-11</td>
</tr>
</tbody>
</table>

  **Note:** SYS_BEGIN, SYS_END and TRANS_ID Timestamp values truncated for reason of space.
### Updating Bi-Temporal Data

- **Update CREDIT interest backward-in-time**
  - Date/time: 2011-10-15-18.13.19

  ```sql
  UPDATE INTEREST_RATE
  FOR PORTION OF BUSINESS TIME
  FROM '01.09.2011' TO '31.12.9999'
  SET INTEREST_RATE = 1.99
  WHERE TYPE = 'CREDIT'
  AND CUST_SEGMENT = 'VIP';
  ```

- **Valuable requests to the System**
  - According to your knowledge on 14 October 2011, which was the Credit Interest for VIP customers on that date (or any other date)?
  - According to your current (16 October 2011) knowledge, which was the Credit Interest for VIP customer on that date (or any other date)?

### Reference Material

1. SG24–7892 DB2 10 for z/OS Technical Overview, December 2010
The End