# Adopting Temporal Tables in DB2 10

## Migration Scenario(s)

Prepared by Angelo Sironi
Sironi IT Consulting s.n.c.

Milan, 9 April 2013
Rome, 10 April 2013





## Agenda

- Intro to Temporal Tables
  - System-period Temporal Tables (STT)
    - Structure & Definition
    - Manipulation
  - Application-period Temporal Tables (ATT)
    - Structure & Definition
    - Manipulation
- System-period Temporal Tables (STT)
  - Migration Scenarios
  - Data Migration
  - Application migration
  - Issues
  - Concluding Remarks

# System-period Temporal Tables




## Time in the Real World and System Time

- **Valid Time** (aka Application Time 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
  - More than one Valid Time possible
- **Transaction Time** (aka **System Time** (DB2))
  - 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)

## Temporal Data in DB2

- **Table-level specification to control management of data 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 data base change
    - Useful for auditing & compliance
- **Three types of temporal tables supported**
  - Application-period temporal tables (ATTs)
  - System-period temporal tables (STTs)
  - Bitemporal tables (BTTs)
- **Time periods are (inclusive, exclusive)**
- **New FROM clause syntax 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 ALWAYS AS ROW BEGIN
       , SYS_END       TIMESTAMP(12) NOT NULL
                            GENERATED ALWAYS AS ROW END
       , PERIOD        SYSTEM_TIME(SYS_BEGIN, SYS_END)
       , TRANS_ID      TIMESTAMP(12) NOT NULL
                  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

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| DEBIT | VIP | 532 | 2011-09-21-15.10.00.549719 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.10.00.549719 |
| CREDIT | VIP | 103 | 2011-09-21-15.10.16.425539 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.10.16.425539 |

- 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)

CURRENT

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.10.00.549719 |
| CREDIT | VIP | 1,03 | 2011-09-21-15.10.16.425539 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.10.16.425539 |

HISTORY

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|

- 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)

CURRENT

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.10.00.549719 |
| CREDIT | VIP | 1,15 | 2011-09-21-15.56.26.342958 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.56.26.342958 |

HISTORY

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,03 | 2011-09-21-15.10.16.425539 | 2011-09-21-15.56.26.342958 | 2011-09-21-15.10.16.425539 |

# System Time Support: Example (cont.)

- Table content BEFORE (current & history)

CURRENT

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.10.00.549719 |
| CREDIT | VIP | 1,15 | 2011-09-21-15.56.26.342958 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.56.26.342958 |

HISTORY

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,03 | 2011-09-21-15.10.16.425539 | 2011-09-21-15.56.26.342958 | 2011-09-21-15.10.16.425539 |

- Let's DELETE some record(s)

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

- Table content AFTER (current & history)

CURRENT

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,15 | 2011-09-21-15.56.26.342958 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.56.26.342958 |

HISTORY

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,03 | 2011-09-21-15.10.16.425539 | 2011-09-21-15.56.26.342958 | 2011-09-21-15.10.16.425539 |
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 2011-09-21-16.13.59.525322 | 2011-09-21-15.10.00.549719 |



11

# System Time Support: Example (cont.)

- Table content

CURRENT

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,15 | 2011-09-21-15.56.26.342958 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.56.26.342958 |

HISTORY

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,03 | 2011-09-21-15.10.16.425539 | 2011-09-21-15.56.26.342958 | 2011-09-21-15.10.16.425539 |
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 2011-09-21-16.13.59.525322 | 2011-09-21-15.10.00.549719 |

- 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

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,15 | 2011-09-21-15.56.26.342958 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.56.26.342958 |
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 2011-09-21-16.13.59.525322 | 2011-09-21-15.10.00.549719 |



12

## System Time Support: Example (cont.)

- **Table content**

HISTORY CURRENT

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,15 | 2011-09-21-15.56.26.342958 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.56.26.342958 |

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,03 | 2011-09-21-15.10.16.425539 | 2011-09-21-15.56.26.342958 | 2011-09-21-15.10.16.425539 |
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 2011-09-21-16.13.59.525322 | 2011-09-21-15.10.00.549719 |

- **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 FROM '2011-09-21-00.00.00.000000'
                   TO CURRENT TIMESTAMP
  WHERE CUST_SEGMENT = 'VIP' ;
```

- **Query result set**

| TYPE | CUST SEGMENT | INTEREST RATE | SYS BEGIN | SYS END | TRANS ID |
|---|---|---|---|---|---|
| CREDIT | VIP | 1,15 | 2011-09-21-15.56.26.342958 | 9999-12-31-24.00.00.000000 | 2011-09-21-15.56.26.342958 |
| CREDIT | VIP | 1,03 | 2011-09-21-15.10.16.425539 | 2011-09-21-15.56.26.342958 | 2011-09-21-15.10.16.425539 |
| DEBIT | VIP | 5,32 | 2011-09-21-15.10.00.549719 | 2011-09-21-16.13.59.525322 | 2011-09-21-15.10.00.549719 |



13

## Period specifications

| SYSTEM_TIME | Period BEGIN TIME = BT | Period END TIME = ET |
|---|---|---|
| AS OF TSx | TSx >= BT | TSx < ET |
| FROM TSx TO TSy | Tsy > BT | Tsx < ET |
| BETWEEN TSx AND TSy | TSy >= BT | Tsx < ET |



14

## Application-period Temporal Tables (ATTs)

- **Allow you to store time-sensitive data from a business perspective**
  - e.g. interest rates on different time periods
- **Each row has a pair of TIMESTAMP or DATE columns, with values managed by the application**
  - Begin column: represents time at which row data begins to be valid
  - End column: represents time at which row data ceases to be valid
- **When application updates the data, DB2 adds, splits, or deletes rows as needed, automatically and transparently**
- **Can be used to model data in the past, present, and future**
- **Uniqueness can be automatically enforced to disallow overlapping of application periods**
- **Unlike System-period temporal tables (STTs), no separate history table is required**




## Business Time Support: Example

- **Create a Business-period temporal table**

```
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
            (BUS_BEGIN_DATE, BUS_END_DATE)
   , PRIMARY KEY(TYPE, CUST_SEGMENT,
            BUSINESS_TIME WITHOUT OVERLAPS) );
```

- **Primary Key definition**

| IXNAME | COLNAME | COLSEQ | ORDERING | PERIOD |
|---|---|---|---|---|
| BUS_INTE_£_NTH | TYPE | 1 | A | |
| | CUST_SEGMENT | 2 | A | |
| | BUS_END_DATE | 3 | A | C |
| | BUS_BEGIN_DATE | 4 | A | B |

## Business Time Support: Example (cont.)

- Insert a couple of rows

```
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 SYS_INTEREST_RATE
   ("TYPE", CUST_SEGMENT, INTEREST_RATE,
      BUS_BEGIN_DATE, BUS_EN)
  VALUES ('CREDIT', 'VIP', 1.03,
          '18.10.2011', '31.12.9999') ;
```

- BUS_INTEREST_RATE content

| TYPE | CUST SEGMENT | INTEREST RATE | BUS BEGIN DATE | BUS END DATE |
|---|---|---|---|---|
| DEBIT | VIP | 5,32 | 2011-10-18 | 9999-12-31 |
| CREDIT | VIP | 1,03 | 2011-10-18 | 9999-12-31 |





## Business Time Support: Example (cont.)

- Table content BEFORE

| TYPE | CUST SEGMENT | INTEREST RATE | BUS BEGIN DATE | BUS END DATE |
|---|---|---|---|---|
| DEBIT | VIP | 5,32 | 2011-10-18 | 9999-12-31 |
| CREDIT | VIP | 1,03 | 2011-10-18 | 9999-12-31 |

- Let's UPDATE the interest rate

```
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

| TYPE | CUST SEGMENT | INTEREST RATE | BUS BEGIN DATE | BUS END DATE |
|---|---|---|---|---|
| DEBIT | VIP | 5,32 | 2011-10-18 | 9999-12-31 |
| CREDIT | VIP | 1,15 | 2012-10-11 | 9999-12-31 |
| CREDIT | VIP | 1,03 | 2011-10-18 | 2012-10-11 |

## Migration Scenarios

- Based on DeveloperWorks Paper "**Adopting temporal tables in DB2, Part 2: Advanced migration scenarios for system-period temporal tables**"

| | Existing solution records history? | Existing history is in a separate table? | No. of timestamp columns used for versioning? | Your period model? | Article |
|---|---|---|---|---|---|
| Scenario 0 | No | N/A | None | None | Part 1 |
| Scenario 1 | Yes | Yes | Two | Inclusive-exclusive | Part 1 |
| Scenario 2 | Yes | No | Two | Inclusive-exclusive | Part 2 |
| Scenario 3 | Yes | No | One | Inclusive-exclusive | Part 2 |
| Scenario 4 | Yes | No | Two | Inclusive-inclusive | Part 2 |





## Scenario 2 — Single table for current & historical data

```
CREATE TABLE employees_s2
  ( empid BIGINT NOT NULL,
    name VARCHAR(20),
    deptid INTEGER,
    salary DECIMAL(7,2),
    system_begin TIMESTAMP(6) NOT NULL DEFAULT
                                  CURRENT TIMESTAMP,
    system_end TIMESTAMP(6) NOT NULL DEFAULT
             TIMESTAMP '9999-12-30 00:00:00.000000',
    PRIMARY KEY (empid, system_begin) );
```

**Current implementation**

| empid | name | deptid | salary | system_begin | system_end |
|---|---|---|---|---|---|
| 500 | Peter | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-06-30 09:15:45.123456 |
| 1000 | John | 1 | 5.000,00 | 2010-05-11 12:00:00.000000 | 9999-12-30 00:00:00.000000 |
| 1212 | James | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-05-11 09:30:00.100000 |
| 1212 | James | 2 | 4.500,00 | 2011-05-11 09:30:00.100000 | 9999-12-30 00:00:00.000000 |
| 4711 | Maddy | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-07-30 09:25:47.123456 |
| 4711 | Maddy | 1 | 5.250,00 | 2011-07-30 09:25:47.123456 | 9999-12-30 00:00:00.000000 |

## Scenario 2: Data Migration Details – 1

- **Modify the definition of system_begin and system_end columns**
  - TIMESTAMP(6) ➔ TIMESTAMP(12)
  - DEFAULT CURRENT TIMESTAMP etc. ➔ GENERATED ALWAYS AS ROW BEGIN/END
- **Add a TRANSACTION ID column**
- **Add PERIOD SYSTEM_TIME specification**
- **Define a new table for history records**
- **Modify Primary Key definition**
  - In a system-period temporal table, system_begin column is typically not part of the primary key
  - Details (and side-effects…!) later
- **Enable versioning**




## Scenario 2: Data Migration Details – 2

- **Table conversion & data «migration»**

1. Change TIMESTAMP type and add the transID column

```
ALTER TABLE employees_s2
    ALTER COLUMN system_begin SET DATA TYPE TIMESTAMP(12)
    ALTER COLUMN system_end   SET DATA TYPE TIMESTAMP(12)
    ADD   COLUMN trans_start TIMESTAMP(12)
          GENERATED ALWAYS AS TRANSACTION START ID
          IMPLICITLY HIDDEN;
```

2. Set auto generation of system time columns and add a system time period

```
ALTER TABLE employees_s2
    ALTER COLUMN system_begin DROP DEFAULT
    ALTER COLUMN system_end   DROP DEFAULT  ;

ALTER TABLE EMPLOYEES_S2
    ALTER COLUMN SYSTEM_BEGIN
          SET GENERATED ALWAYS AS ROW BEGIN
    ALTER COLUMN SYSTEM_END
          SET GENERATED ALWAYS AS ROW END
    ADD PERIOD SYSTEM_TIME (SYSTEM_BEGIN, SYSTEM_END);
```

## Scenario 2: Data Migration Details – 3

- **Table conversion & data «migration» (cont.)**
  3. REORG employees_s2 tablespace;
  4. Create a new history table and move history rows into it

```
CREATE TABLE employees_s2_hist LIKE employees_s2;

INSERT INTO EMPLOYEES_S2_HIST
    SELECT EMPID, NAME, DEPTID, SALARY
         , SYSTEM_BEGIN, SYSTEM_END
      FROM EMPLOYEES_S2
     WHERE SYSTEM_END <> '9999-12-30-00.00.00';

DELETE FROM EMPLOYEES_S2
 WHERE SYSTEM_END <> '9999-12-30-00.00.00';
```

  5. Change the primary key

```
ALTER TABLE employees_s2 DROP PRIMARY KEY
                         ADD PRIMARY KEY (empid);
```




## Scenario 2: Data Migration Details – 4

- **Table conversion & data «migration» (cont.)**
  - Enable versioning

```
ALTER TABLE employees_s2
  ADD VERSIONING USE HISTORY TABLE employees_s2_hist;
```

## Scenario 2: Application Migration – 1

- **History maintenance**
  - If maintained by TRIGGERS
    - DROP Update & Delete Triggers
  - If maintained by application code
    - Remove those portions of code
- **INSERT & UPDATE statements**
  - Modify if they insert / update values in system begin / end columns
- **DELETE statements to purge history rows**
  - Must reference history table




## Scenario 2: Application Migration – 2

- **Retrieving all versions of a qualifying row**
  - **Before** migration

```
SELECT empid, name, deptid, salary
     , system_begin, system_end
  FROM employees_s2
 WHERE empid = 4711;
```

  - **After** migration

```
SELECT empid, name, deptid, salary
     , system_begin, system_end
  FROM employees_s2
   FOR SYSTEM_TIME FROM '0001-01-01-00.00.00'
                   TO '9999-12-30-00.00.00'
 WHERE empid = 4711;
```

- **Issue**
  - After migration, the "before migration" queries will work, but will provide a potentially incomplete result!
  - The above may affect all queries (static and dynamic) against the base table, synonyms/aliases and views that reference the any migrated base table(s)

## Scenario 2: Application Migration – 3

- **Retrieving some (Current or Past) Version of a Row**
  - Before migration

```
SELECT empid, name, deptid, salary
     , system_begin, system_end
  FROM employees_s2
 WHERE empid = 4711
   AND SYSTEM_BEGIN <= :somets
   AND SYSTEM_END    > :somets;
```

  - After migration

**Current Version only**

```
SELECT empid, name, deptid
     , salary, system_begin
     , system_end
  FROM employees_s2
 WHERE empid = 4711
```

**Any Version of the Row**

```
SELECT empid, name, deptid
     , salary, system_begin
     , system_end
  FROM employees_s2
 FOR SYSTEM_TIME AS OF :somets
 WHERE empid = 4711
```




## Performance Implications

**Current Version only**

```
SELECT empid, name, deptid
     , salary, system_begin
     , system_end
  FROM employees_s2
 WHERE empid = 4711
```

**Any Version of the Row**

```
SELECT empid, name, deptid
     , salary, system_begin
     , system_end
  FROM employees_s2
 FOR SYSTEM_TIME AS OF :somets
 WHERE empid = 4711
```

Access Base Table only

Access both Base Table
AND
History Table

## Scenario 2: Application Migration – 4

- **Retrieving all versions of a qualifying row – JOINS**
  - **Before** migration

```
SELECT e.empid, name, deptid, salary, e.system_begin
     , system_end, award_date, award_amount
  FROM employees_s2 e
     , awards       a
 WHERE e.empid = 4711
   AND a.empid = e.empid
   AND a.system_begin = e.system_begin;
```

  - **After** migration

```
SELECT e.empid, name, deptid, salary, e.system_begin
     , system_end, award_date, award_amount
  FROM employees_s2 e  FOR SYSTEM_TIME
                         FROM '0001-01-01-00.00.00'
                           TO '9999-12-30-00.00.00'
     , awards       a
 WHERE e.empid = 4711
   AND a.empid = e.empid
   AND a.system_begin = e.system_begin;
```




## DB2 10 for LUW Temporal Registers

- **Two new Special Registers (not available in DB2 10 for z/OS)**
  - CURRENT TEMPORAL SYSTEM_TIME
  - CURRENT TEMPORAL BUSINESS_TIME
- **However, can only be used to specify a point in time, not a period of time**
  - Only partially helpful in avoiding the mentioned issues

## What about Referential Integrity?

- **Pre-migration Primary Key**
  - (EMPID, SYSTEM_BEGIN)
- **Post-migration Primary Key**
  - EMPID for current data
  - (EMPID, SYSTEM_BEGIN) for history table
- **If Foreign Keys exists in dependent table(s), they must be modified (or removed…)**
  - Il modified, dependent rows can only reference current data…!
  - TIMESTAMP(6) ➔ TIMESTAMP(12) for system_end column, to properly support joins, unless RI not defined and join based on time inclusion
- **Comments / Issues**
  - Apparently, everything works fine as long as no record is (logically) deleted
  - However, also insertion of backward in time events won't work without modifications




## What about Referential Integrity?

- **Pre-Migration**

**employees_s2**

| empid | name | deptid | salary | system_begin | system_end |
|---|---|---|---|---|---|
| 500 | Peter | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-06-30 09:15:45.123456 |
| 1000 | John | 1 | 5.000,00 | 2010-05-11 12:00:00.000000 | 9999-12-30 00:00:00.000000 |
| 1212 | James | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-05-11 09:30:00.100000 |
| 1212 | James | 2 | 4.500,00 | 2011-05-11 09:30:00.100000 | 9999-12-30 00:00:00.000000 |
| 4711 | Maddy | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-07-30 09:25:47.123456 |
| 4711 | Maddy | 1 | 5.250,00 | 2011-07-30 09:25:47.123456 | 9999-12-30 00:00:00.000000 |

**awards**

| empid | system_begin | award_date | award_amount |
|---|---|---|---|
| 500 | 2010-05-11 12:00:00.000000 | 2011-03-01 | 400 |
| 1212 | 2010-05-11 12:00:00.000000 | 2011-03-01 | 400 |

## What about Referential Integrity?

- **Post-Migration**

**employees_s2**

| empid | name | deptid | salary | system_begin | system_end |
|---|---|---|---|---|---|
| 1000 | John | 1 | 5.000,00 | 2010-05-11 12:00:00.000000 | 9999-12-30 00:00:00.000000 |
| 1212 | James | 2 | 4.500,00 | 2011-05-11 09:30:00.100000 | 9999-12-30 00:00:00.000000 |
| 4711 | Maddy | 1 | 5.250,00 | 2011-07-30 09:25:47.123456 | 9999-12-30 00:00:00.000000 |

?

**employees_s2_hist**

| empid | name | deptid | salary | system_begin | system_end |
|---|---|---|---|---|---|
| 500 | Peter | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-06-30 09:15:45.123456 |
| 1212 | James | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-05-11 09:30:00.100000 |
| 4711 | Maddy | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | 2011-07-30 09:25:47.123456 |

**awards**

| empid | system_begin | award_date | award_amount |
|---|---|---|---|
| 500 | 2010-05-11 12:00:00.000000 | 2011-03-01 | 400 |
| 1212 | 2010-05-11 12:00:00.000000 | 2011-03-01 | 400 |

- **Note:**
  - SYSTEM_BEGIN and SYSTEM_END must be TIMESTAM(12)




## A Word on Performance

- **UPDATE & DELETE statements**
  - Might experience a cost increase
- **SELECT statements**
  - If accessing the base table only, should see a cost reduction
    - Both for CPU and elapsed time
  - Else, could experience a cost increase
    - Accessing both base and history tables
    - Additional sorts may be required
- **When retrieving current version**
  - If specifying «FOR SYSTEM_TIME AS», DB2 (for z/OS) will access both base and history table
    - Even with «FOR SYSTEM TIME AS CURRENT TIMESTAMP»
  - Else, DB2 will access the base table only!

## What about Scenario 3?

| | Existing solution records history? | Existing history is in a separate table? | No. of timestamp columns used for versioning? | Your period model? | Article |
|---|---|---|---|---|---|
| Scenario 0 | No | N/A | None | None | Part 1 |
| Scenario 1 | Yes | Yes | Two | Inclusive-exclusive | Part 1 |
| Scenario 2 | Yes | No | Two | Inclusive-exclusive | Part 2 |
| **Scenario 3** | Yes | No | One | Inclusive-exclusive | Part 2 |
| Scenario 4 | Yes | No | Two | Inclusive-inclusive | Part 2 |

- **Sample data**

| empid | name | deptid | salary | system_begin | status |
|---|---|---|---|---|---|
| 500 | Peter | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | H |
| 500 | - | - | - | 2011-06-30-09:15:45.123456 | D |
| 1000 | John | 1 | 5.000,00 | 2010-05-11 12:00:00.000000 | C |
| 1212 | James | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | H |
| 1212 | James | 2 | 4.500,00 | 2011-05-11 09:30:00.100000 | C |
| 4711 | Maddy | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 | H |
| 4711 | Maddy | 1 | 5.250,00 | 2011-07-30 09:25:47.123456 | C |




## Scenario 3: Migration Consideration

- **Major differences vs. Scenario 2**
  - Adding column system_end and assigning a value
    - Example supplied by mentioned Paper
  - Removal of rows with status = 'D'
- **Issues**
  - Very similar to Scenario 2
  - Time-based Referential Integrity more difficult to avoid, as joins based on time-inclusion not available on original design

## Scenario 3 Variations

- **A quite common variation**

| empid | name | deptid | salary | system_begin |
|---|---|---|---|---|
| 500 | Peter | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 |
| 1000 | John | 1 | 5.000,00 | 2010-05-11 12:00:00.000000 |
| 1212 | James | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 |
| 1212 | James | 2 | 4.500,00 | 2011-05-11 09:30:00.100000 |
| 4711 | Maddy | 1 | 4.000,00 | 2010-05-11 12:00:00.000000 |
| 4711 | Maddy | 1 | 5.250,00 | 2011-07-30 09:25:47.123456 |

- **Pre-migration maintenance**
  - Just INSERTs and, possibly, DELETEs
- **Post-migration maintenance**
  - Old logic does not work!
  - Must use Searched UPDATE … or …
  - Searched DELETE + INSERT
    - **Obvious impact on Referential Integrity**




## Additional Implications (DB2 for z/OS)

- **The system-period temporal table must not have:**
  - A security label column
  - Any row permissions
  - A clone table defined on it
  - A materialized query table definition
  - Any other tables defined in the same table space
  - No utility operation is allowed that will delete data from the system-period temporal table.
    - This includes LOAD REPLACE, REORG DISCARD, and CHECK DELETE YES
  - A history table, or a column of a history table, cannot be renamed
- **For a complete list, refer to DB2 Pubs.**
  - DB2 for z/OS Admin. Guide, Restrictions for system-period data versioning
  - DB2 for LUW, Restrictions for system-period data versioning http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.dbobj.doc%2Fdoc%2Fc0058940.html

## Concluding Remarks

- **Number of Scenarios not limited to the ones covered by the two mentioned Papers**
  - A number of variations exists
- **Need to evaluate migration costs vs. advantages / savings**
  - Major advantages / savings related to future code maintenance & future enhancements to temporal support
  - Migration costs
  - Performance implications (positive / negative)
  - Utilities and operation restrictions
  - Cost of educating developer and, maybe, even end-users
- **If application is old and not subject to large or continuous costly maintenance, is it worth migrating?**
- **Personal recommendation:**
  - Apply the enhancement to new application(s) on newly created set of tables






**Not always !!!**

# Reference Material


[1] SA22-7649 **z/OS Support for Unicode: Using Unicode Services**

[2] SG24-7892 **DB2 10 for z/OS Technical Overview**, December 2010

[3] SG24-7942 **DB2 10 for z/OS Performance Topics**, June 2011

[4] SG24-7959 **Security Functions with DB2 10 for z/OS**, August 2011

[5] Cynthia M. Saracco and al., **A matter of time: Temporal data management in DB2 10**, IBM Corp., April 2012

[6] Matthias Nicola et al., **Managing time in DB2 with temporal consistency**, IBM Corp., July 2012

[7] Matthias Nicola, **Best Practices: Temporal Data Management with DB2**, IBM Silicon Valley Lab., August 2012

**Note:** applies to DB2 10 for Linux, Unix and Windows

[8] **IBM DB2 for z/OS Best Practices Web Page** at www.ibm.com/developerworks/data/bestpractices/db2zos/

[9] Jan-Eike Michels & Matthias Nicola, **Adopting temporal tables in DB2, Part 1: Basic migration scenarios for system-period tables**, http://www.ibm.com/developerworks/data/library/techarticle/dm-1210temporaltablesdb2/index.html?ca=drs-

[10] Matthias Nicola & Jan-Eike Michels, **Adopting temporal tables in DB2, Part 2: Advanced migration scenarios for system-period tables**, http://www.ibm.com/developerworks/data/library/techarticle/dm-1210temporaltablesdb2pt2/