Introducing DB2 10 Temporal Data Features

Peter Vanroose

abis
TRAINING & CONSULTING

GSE DB2 Belgium
Joint User Group Meeting
IBM, Brussels, 6 June 2013
“30th anniversary of DB2”

DB2 10 temporal data features

Objectives :

- Relational databases and historic (or versioned) data
- New SELECT query syntax for “temporal” requests
- Table setup for “system time” versioning
- Business time: data validity time period
- Bi-temporal tables
- ...

Relational databases and historic (or versioned) data 1

DB2 ==> very efficient *transactional data server* :

- **ACID: atomic (transactions) ==> commit / rollback**

  consistent ==> each visible DB state makes sense

  isolated ==> through locking (& isolation levels)

  durable ==> permanent changes

  BUT no notion of “keeping track of history”

Data warehouse & business intelligence :

- **often needs / wants historic data**

  (“how did the data look on 1 February?”)

  (trend analysis: “predict future sales from past trends”)

- **not typically a task for a transactional DB server**

  but can be integrated

What we miss: “what was the (ACID) state of my data on <time instant>” ?

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Reasons for “temporal” data queries in a relational DB

1.1

- **Not really meant for BI or DW**
- **Tracability of data changes for *auditing* purposes:**
  - “*What data was used in last month’s investment assessment?*”
  - “*Please re-run the tax computation of last 31 December*”
  - “*Since when are you giving a 5% price reduction to that client?*”
  - “*Please trace back* <certain business data> *over the last year.*”
- **Tracability of data changes for *business tracing* purposes:**
  - “*Where did we send that order to last week?*”

    ==> What was the customer address on May 30 at 15:43 ?
- **Storing data validity information:**
  - **Customer: “*My address as of 1 September will be ...*”**
  - **Insurance record(s): “*covered time interval: 1 January -- 30 June*”**
  - **Promotional action: “*Price will be 20% off between ... and ...*”**
  - **Product availability period(s) (possibly with retroactive effect)**

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

New SELECT query syntax for “temporal” requests 2

Example table: customers

id	name	address	telephone	amount_sold
1	Janssen	Singel 9	016/123456	1043.50
2	Dupont	A.Max 3	02/9876543	745.00
3	Thiery	Square 1	03/1234567	6100.00
8	Van Dijk	Dijk 8	0476/54321	75.25
9	Berends	Dorp 17	09/8765432	3201.43
10	Zander	Centre 4	-	123.45

```
SELECT * FROM customers WHERE id = 3 ;
```

id	name	address	telephone	amount_sold
3	Thiery	Square 1	03/1234567	6100.00

```
SELECT * FROM customers AS OF SYSTEM TIME '2013-05-30-15.45.00' WHERE id = 3 ;
```

(*)

id	name	address	telephone	amount_sold
3	Thiery	Zand 98	03/1234567	6100.00

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

New SELECT query syntax for “temporal” requests

- New ANSI / ISO ***SQL:2011 Standard*** syntax:

```
... FROM <table> AS OF SYSTEM TIME <timestamp> ...
```

- DB2 syntax (2 alternative forms):

```
... FROM <table> FOR SYSTEM_TIME AS OF <timestamp> ...
... FROM <table> AS OF TIMESTAMP <timestamp> ...
```

- DB2 for LUW extension:

```
... FROM <table> FOR SYSTEM_TIME AS OF <date>  ...
```

- Oracle syntax:

```
... FROM <table> AS OF TIMESTAMP <timestamp>  ...
```

- Examples:

```
SELECT * FROM customers FOR SYSTEM_TIME AS OF current timestamp ;
SELECT * FROM customers FOR SYSTEM_TIME AS OF current date - 3 days ;
SELECT * FROM customers AS OF TIMESTAMP current timestamp - 1 min ;
SELECT * FROM customers FOR SYSTEM_TIME AS OF :hv ;
```

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Table setup for “system time” versioning

3

But ... tables are (still) not “versioned” by default !

Think about how you would implement “versioned data” manually:

id	name	address	telephone	amount_sold	valid_from	valid_until
1	Janssen	Singel 9	016/123456	1043.50	2013-02-02-14.02.02	-
2	Dupont	A.Max 3	02/9876543	745.00	2004-08-20-11.11.11	-
3	Thiery	Square 1	03/1234567	6100.00	2013-06-04-15.13.32	-
8	Van Dijk	Dijk 8	0476/54321	75.25	2012-01-04-12.00.00	-
9	Berends	Dorp 17	09/8765432	3201.43	2012-04-12-18.00.00	-
10	Zander	Centre 4	-	123.45	2012-11-15-09.00.00	-
1	Janssen	Singel 9	016/123456	943.50	2011-03-12-09.13.42	2013-02-02-14.02.02
1	Janssen	Singel 9	-	943.50	2004-03-30-15.13.42	2011-03-12-09.13.42
3	Thiery	Zand 98	03/1234567	6100.00	2010-01-01-00.00.00	2013-06-04-15.13.32
4	Pieters	Rand 7A	-	100.00	2010-08-31-12.21.53	2012-07-21-16.24.13
4	Pieters	Berg 71	-	100.00	2012-07-21-16.24.13	2012-12-31-23.59.59

Technical challenges:

store delta’s? duplicate PK values; query performance&complexity; triggers for update & delete; default values for hidden cols; ...

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Table setup for “system time” versioning: how DB2 wants it

3.1

```
CREATE TABLE customers
    (id             int NOT NULL
    , name          varchar(64)
    , address       varchar(128)
    , telephone     varchar(32)
    , amount_sold   dec(9,2)
    , valid_from    timestamp(12)  GENERATED ALWAYS AS ROW BEGIN  NOT NULL
    , valid_until                  GENERATED ALWAYS AS ROW END    NOT NULL
    , trans_id      timestamp(12)  GENERATED ALWAYS AS TRANSACTION START ID
    , PRIMARY KEY (id)
    , PERIOD SYSTEM_TIME (valid_from, valid_until)
    );

CREATE TABLE customers_history LIKE customers ;

ALTER TABLE customers
    ADD VERSIONING   USE HISTORY TABLE customers_history ;
```

- may also ALTER customers: ADD three columns & PERIOD spec
- the three columns could be declared as IMPLICITLY HIDDEN

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Table setup for “system time” versioning: configuration issues

3.2

- base and history table(space) *must* have byte-compatible rows:
  - same column names, same data types, same order & NOT NULL
  - exactly what “CREATE .. LIKE ..” provides
- no further similarities needed
  - may have different indexes
  - may have different check constraints and FKs
    (typically, the history table should have none)
  - may have different partitioning, buffer pool, page size, compress
  - history table *should* have all direct DML blocked
    (since it should be completely transparent to applications)
- ALTER TABLE (column alterations or additions) on base table
  automatically updates the history table definition

Table setup for “system time” versioning: sample data

3.3

customers table:

id	name	address	telephone	amount_sold	valid_from	valid_until
1	Janssen	Singel 9	016/123456	1043.50	2013-02-02-14.02.02	9999-12-30-00.00.00
2	Dupont	A.Max 3	02/9876543	745.00	2004-08-20-11.11.11	9999-12-30-00.00.00
3	Thiery	Square 1	03/1234567	6100.00	2013-06-04-15.13.32	9999-12-30-00.00.00
8	Van Dijk	Dijk 8	0476/54321	75.25	2012-01-04-12.00.00	9999-12-30-00.00.00
9	Berends	Dorp 17	09/8765432	3201.43	2012-04-12-18.00.00	9999-12-30-00.00.00
10	Zander	Centre 4	-	123.45	2012-11-15-09.00.00	9999-12-30-00.00.00

customers_history table:

id	name	address	telephone	amount_sold	valid_from	valid_until
1	Janssen	Singel 9	016/123456	943.50	2011-03-12-09.13.42	2013-02-02-14.02.02
1	Janssen	Singel 9	-	943.50	2004-03-30-15.13.42	2011-03-12-09.13.42
3	Thiery	Zand 98	03/1234567	6100.00	2010-01-01-00.00.00	2013-06-04-15.13.32
4	Pieters	Rand 7A	-	100.00	2010-08-31-12.21.53	2012-07-21-16.24.13
4	Pieters	Berg 71	-	100.00	2012-07-21-16.24.13	2012-12-31-23.59.59

(note: precision of timestamp columns: contain 12 additional fractional digits!)
(transaction start id column: will be NULL, or equal to (oldest) valid_from)

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Table setup for “system time” versioning: effects on DML

3.4

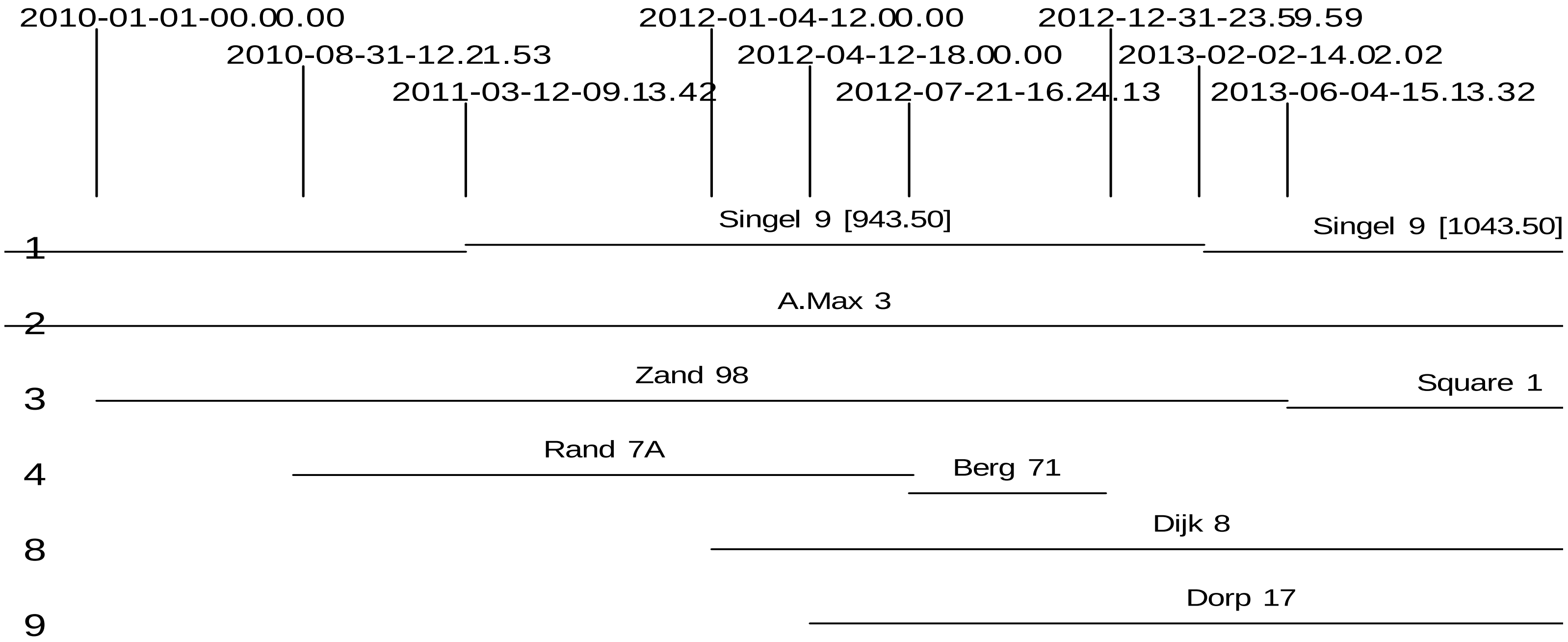
- **customer_history rows: *never* inserted/updated/deleted manually !**
- **on INSERT in customer:**
  - **the three additional columns are auto-filled by DB2:**
    - valid_from: with *current timestamp(12)*
    - valid_until: with ‘9999-12-30-00.00.00.000000000000’
      (this makes sure that the value cannot become invalid
      after time zone conversions / mappings !)
    - trans_id: with NULL (if nullable), otherwise with current timestamp
- **on UPDATE of row(s) in customer:**
  - **original (unchanged) row is “moved” to customer_history**
    - where *valid_until* is changed to *current timestamp(12)*
  - **modified row: valid_from is modified to *current timestamp(12)***
- **on DELETE of row(s) in customer:**
  - **original (old) row is “moved” to customer_history**
    - where *valid_until* is changed to *current timestamp(12)*

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Interpretation of system time validity intervals

4



PK temporal uniqueness: **for every time instant, there is at most one data row per PK value.**

Since e.g. 2011-03-12-09.13.42 is the commit timestamp of the update,

it belongs to the middle time interval, *not* the left one.

In general, the "valid_from" (start) time is an *inclusive* boundary,

while the "valid_until" (end) time is an *exclusive* boundary.

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for "temporal" requests
3. Table setup for "system time" versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

System time: guarantees

4.1

- “Ordinary” SELECT queries never need to access the history table

  ==> base table looks exactly as before

  (except for additional columns)

  ==> no need to revisit existing applications

  identical access paths

- “Ordinary” INSERT/UPDATE/DELETE notice additional overhead similar to classical triggers

- History can only be forged by modifying the history table directly
  - base table ROW BEGIN & END columns are not updatable
  - BUT history table ROW BEGIN & END columns are updatable

  ==> it is possible to create inconsistent data

  (viz. violate temporal uniqueness on PK)

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

System time: additional DML possibilities

4.2

SELECT ... FROM customers AS OF TIMESTAMP current timestamp(12)
is equivalent to
SELECT ... FROM customers
and should not need to access the history table (but it does!)

SELECT ... FROM customers AS OF TIMESTAMP current date
is *NOT* equivalent to the above!
==> it's equivalent to "last midnight"

SELECT ... FROM customers AS OF TIMESTAMP current date + 1 day
is *INVALID* (as is any future date)
==> but is accepted!! (both on LUW and on z/OS)

SELECT ... FROM customers FOR SYSTEM_TIME FROM <ts1> TO <ts2>
- the time range is <ts1> *inclusive* but <ts2> *exclusive*
- might return multiple rows for the same PK
- makes sense to include (one of) the "valid_from" or "valid_until" columns in selection
- if <ts1> is larger than or equal to <ts2>, the result set is empty

SELECT ... FROM customers FOR SYSTEM_TIME BETWEEN <ts1> AND <ts2>
- the time range is <ts1> *inclusive* and also <ts2> *inclusive*

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for "temporal" requests
3. Table setup for "system time" versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

System time: additional DML possibilities

- **Use of the CURRENT TEMPORAL SYSTEM_TIME special register:**

SELECT address FROM customers WHERE id=3

```
ADDRESS
--------------------------------------------------------------------------------------
Square 1

  1 record(s) selected.
```

SELECT address FROM customers AS OF TIMESTAMP '2012-01-01' WHERE id=3

```
ADDRESS
--------------------------------------------------------------------------------------
Zand 89

  1 record(s) selected.
```

SET current temporal system_time='2012-01-01';

SELECT address FROM customers WHERE id=3;

```
ADDRESS
--------------------------------------------------------------------------------------
Zand 89

  1 record(s) selected.
```

SELECT address FROM customers AS OF TIMESTAMP '2012-01-01' WHERE id=3

```
SQL20524N  The statement failed because of an invalid period specification or
period clause for period "SYSTEM_TIME". Reason code "6".  SQLSTATE=428HY
```

UPDATE customers SET address = 'Avenue Louise 9' WHERE id=3

```
SQL20535N  The data change operation "UPDATE" is not supported for the target
object "CUSTOMERS" because of an implicit or explicit period specification
involving "SYSTEM_TIME". Reason code: "1".  SQLSTATE=51046
```

SET current temporal system_time=NULL;

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

System time: performance considerations

The query

```
SELECT ... FROM customers AS OF TIMESTAMP <ts> WHERE <cond>
```

is implemented as follows (as can be seen from EXPLAIN):

```
    SELECT ... FROM customers
        WHERE <cond>
            AND valid_from <= <ts>
UNION ALL
    SELECT ... FROM customers_history
        WHERE <cond>
            AND valid_from <= <ts>
            AND valid_until  >  <ts>
```

Could be important to create index(es)

on columns valid_from and/or valid_until,

possibly composite with other columns

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

System time: some use cases

1. Compliance & auditing:

- never need to use “AS OF” queries
- history table functions as a “change log”
- to re-run an application on the data of last month:
    - use the CURRENT TEMPORAL SYSTEM_TIME register
    - make sure that all tables are “temporal”!

2. Business Intelligence related to time evolution of data:

- “who was our best customer at the end of last month?”
- application could directly query the base + history tables
- or: take summary snapshots at several time instants: eg:

```
WITH dates(t) AS (
    SELECT date('2012-01-01') FROM sysibm.sysdummy1
    UNION ALL
    SELECT t + 1 month FROM dates WHERE t + 1 month < current date
)
SELECT SUM(amount_sold) FROM dates d, customers AS OF TIMESTAMP d.t
GROUP BY d.t                -- (although this won’t work syntactically: need host variable)
```

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

System time: some use cases

3. Compare data at two times in the past (or current)

- **detailed:**

```
SELECT a.id, a.address AS old, b.address AS new
FROM  customers AS OF TIMESTAMP :date1  a
          FULL OUTER JOIN
           customers AS OF TIMESTAMP :date2  b
          ON a.id = b.id
WHERE a.address is distinct from b.address
```

- **summaries:**

```
SELECT SUM(amount_sold), :date1
FROM  customers AS OF TIMESTAMP :date1
UNION ALL
SELECT SUM(amount_sold), :date2
FROM customers AS OF TIMESTAMP :date2
```

Resembles use case of versioning systems (git, subversion, CVS)

4. Point in time recovery

```
UPDATE customers c
SET address = (SELECT address FROM customers AS OF TIMESTAMP :x
                         WHERE id = c.id)
```

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: data validity time period

Want more control over the “valid_from” and “valid_until” values

- time instant of UPDATE is not necessarily

  time instant of when this new fact becomes valid
- example: address change should become active on 1 September

No longer about “transaction time” but about “effective” timespans.

Application should be able to insert into or update the validity dates

But still want “temporal uniqueness” guarantees from DB2

- DB2 syntax for querying a “business temporal” table:

  ... FROM <table> **FOR BUSINESS_TIME AS OF** <timestamp or date> ...
- no SQL ANSI/ISO standard (yet)

Careful: without an “AS OF”, returns the full history (all versions):

... FROM <table> ...

Table setup for “business time” versioning: how DB2 wants it 6.1

```
CREATE TABLE customers
    (id              int NOT NULL
    , name           varchar(64)
    , address        varchar(128)
    , telephone      varchar(32)
    , amount_sold    dec(9,2)
    , valid_from     timestamp(6)    NOT NULL
    , valid_until    timestamp(6)    NOT NULL
    , PERIOD BUSINESS_TIME (valid_from, valid_until)
    , PRIMARY KEY (id, BUSINESS_TIME WITHOUT OVERLAPS)
    );
```

- No history table!
- “id” could now have duplicates

  ==> need a composite primary key
- New uniqueness concept: *temporal uniqueness*
- Enforced by a new type of unique index:

```
    CREATE UNIQUE INDEX <name>
        ON <table> (<cols>, BUSINESS_TIME WITHOUT OVERLAPS) ;
```

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: inserting data 6.2

- **No defaults for “valid_from” and “valid_until”**

  ==> application *must* explicitly state the validity period

  (since these columns are NOT NULL)

==> “valid_until” could still be set to e.g. 9999-12-30 or 9999-12-31

id	name	address	telephone	amount_sold	valid_from	valid_until
1	Janssen	Singel 9	016/123456	1043.50	2013-02-02-14.02.02	9999-12-31-23.59.59
2	Dupont	A.Max 3	02/9876543	745.00	2004-08-20-11.11.11	9999-12-31-23.59.59
3	Thiery	Square 1	03/1234567	6100.00	2013-06-04-15.13.32	9999-12-31-23.59.59
8	Van Dijk	Dijk 8	0476/54321	75.25	2012-01-04-12.00.00	9999-12-31-23.59.59
9	Berends	Dorp 17	09/8765432	3201.43	2012-04-12-18.00.00	9999-12-31-23.59.59
10	Zander	Centre 4	-	123.45	2012-11-15-09.00.00	9999-12-31-23.59.59
1	Janssen	Singel 9	016/123456	943.50	2011-03-12-09.13.42	2013-02-02-14.02.02
1	Janssen	Singel 9	-	943.50	2004-03-30-15.13.42	2011-03-12-09.13.42
3	Thiery	Zand 98	03/1234567	6100.00	2010-01-01-00.00.00	2013-06-04-15.13.32
4	Pieters	Rand 7A	-	100.00	2010-08-31-12.21.53	2012-07-21-16.24.13
4	Pieters	Berg 71	-	100.00	2012-07-21-16.24.13	2012-12-31-23.59.59

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: updating data

- **Update statements without “temporal” specification will update ALL rows, not just the ones “as of current timestamp”:**

UPDATE customers SET telephone = '03/7654321' WHERE id = 3

id	name	address	telephone	amount_sold	valid_from	valid_until
1	Janssen	Singel 9	016/123456	1043.50	2013-02-02-14.02.02	9999-12-31-23.59.59
2	Dupont	A.Max 3	02/9876543	745.00	2004-08-20-11.11.11	9999-12-31-23.59.59
3	Thiery	Square 1	03/7654321	6100.00	2013-06-04-15.13.32	9999-12-31-23.59.59
8	Van Dijk	Dijk 8	0476/54321	75.25	2012-01-04-12.00.00	9999-12-31-23.59.59
9	Berends	Dorp 17	09/8765432	3201.43	2012-04-12-18.00.00	9999-12-31-23.59.59
10	Zander	Centre 4	-	123.45	2012-11-15-09.00.00	9999-12-31-23.59.59
1	Janssen	Singel 9	016/123456	943.50	2011-03-12-09.13.42	2013-02-02-14.02.02
1	Janssen	Singel 9	-	943.50	2004-03-30-15.13.42	2011-03-12-09.13.42
3	Thiery	Zand 98	03/7654321	6100.00	2010-01-01-00.00.00	2013-06-04-15.13.32
4	Pieters	Rand 7A	-	100.00	2010-08-31-12.21.53	2012-07-21-16.24.13
4	Pieters	Berg 71	-	100.00	2012-07-21-16.24.13	2012-12-31-23.59.59

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: updating data

- **Update statements with “temporal” specification:**

```
UPDATE customers FOR PORTION OF BUSINESS_TIME
                                      FROM '2013-09-01-00.00.00' TO '9999-12-31-23.59.59'
      SET telephone = '03/7654321' WHERE id = 3
```

id	name	address	telephone	amount_sold	valid_from	valid_until
1	Janssen	Singel 9	016/123456	1043.50	2013-02-02-14.02.02	9999-12-31-23.59.59
2	Dupont	A.Max 3	02/9876543	745.00	2004-08-20-11.11.11	9999-12-31-23.59.59
3	Thiery	Square 1	03/7654321	6100.00	2013-09-01-00.00.00	9999-12-31-23.59.59
3	Thiery	Square 1	03/1234567	6100.00	2013-06-04-15.13.32	2013-09-01-00.00.00
8	Van Dijk	Dijk 8	0476/54321	75.25	2012-01-04-12.00.00	9999-12-31-23.59.59
9	Berends	Dorp 17	09/8765432	3201.43	2012-04-12-18.00.00	9999-12-31-23.59.59
10	Zander	Centre 4	-	123.45	2012-11-15-09.00.00	9999-12-31-23.59.59
1	Janssen	Singel 9	016/123456	943.50	2011-03-12-09.13.42	2013-02-02-14.02.02
1	Janssen	Singel 9	-	943.50	2004-03-30-15.13.42	2011-03-12-09.13.42
3	Thiery	Zand 98	03/1234567	6100.00	2010-01-01-00.00.00	2013-06-04-15.13.32
4	Pieters	Rand 7A	-	100.00	2010-08-31-12.21.53	2012-07-21-16.24.13
4	Pieters	Berg 71	-	100.00	2012-07-21-16.24.13	2012-12-31-23.59.59

==> automatic row split when necessary!

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: deleting data 6.4

- **Delete statements with “temporal” specification:**

```
DELETE FROM customers FOR PORTION OF BUSINESS_TIME
                                  FROM '2014-01-01-00.00.00' TO '9999-12-31-23.59.59'
      WHERE id = 3
```

id	name	address	telephone	amount_sold	valid_from	valid_until
1	Janssen	Singel 9	016/123456	1043.50	2013-02-02-14.02.02	9999-12-31-23.59.59
2	Dupont	A.Max 3	02/9876543	745.00	2004-08-20-11.11.11	9999-12-31-23.59.59
3	Thiery	Square 1	03/7654321	6100.00	2013-09-01-00.00.00	**2014-01-01-00.00.00**
3	Thiery	Square 1	03/1234567	6100.00	2013-06-04-15.13.32	2013-09-01-00.00.00
8	Van Dijk	Dijk 8	0476/54321	75.25	2012-01-04-12.00.00	9999-12-31-23.59.59
9	Berends	Dorp 17	09/8765432	3201.43	2012-04-12-18.00.00	9999-12-31-23.59.59
10	Zander	Centre 4	-	123.45	2012-11-15-09.00.00	9999-12-31-23.59.59
1	Janssen	Singel 9	016/123456	943.50	2011-03-12-09.13.42	2013-02-02-14.02.02
1	Janssen	Singel 9	-	943.50	2004-03-30-15.13.42	2011-03-12-09.13.42
3	Thiery	Zand 98	03/1234567	6100.00	2010-01-01-00.00.00	2013-06-04-15.13.32
4	Pieters	Rand 7A	-	100.00	2010-08-31-12.21.53	2012-07-21-16.24.13
4	Pieters	Berg 71	-	100.00	2012-07-21-16.24.13	2012-12-31-23.59.59

==> automatic row split when necessary!

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: guarantees & caveats

- “Ordinary” SELECT always accesses the full table

  ==> including history!

- “Ordinary” INSERT/UPDATE/DELETE update all versions

  ==> unless WHERE on valid_from or valid_until

- INSERTs and “temporal” UPDATEs sometimes refused:

  ==> “duplicate” error from unique index

  when time intervals would overlap

  ==> *temporal uniqueness is guaranteed by DB2*

Business time: additional DML possibilities

6.6

SELECT ... FROM customers FOR BUSINESS_TIME AS OF current timestamp

is *NOT* equivalent to

SELECT ... FROM customers

SELECT ... FROM customers FOR BUSINESS_TIME AS OF current date + 1 day

is totally *VALID* (as is any future date)

SELECT ... FROM customers FOR BUSINESS_TIME FROM <ts1> TO <ts2>

- the time range is <ts1> *inclusive* but <ts2> *exclusive*
- might return multiple rows for the same id
- if <ts1> is larger than or equal to <ts2>, or one is NULL, the result set is empty

SELECT ... FROM customers FOR BUSINESS_TIME BETWEEN <ts1> AND <ts2>

- the time range is <ts1> *inclusive* and also <ts2> *inclusive*

Business time: additional DML possibilities

- **Use of the CURRENT TEMPORAL BUSINESS_TIME special register:**

SELECT address FROM customers WHERE id=3

```
ADDRESS
---------------------------------------------------------------------------------------
Square 1

  1 record(s) selected.
```

SELECT address FROM customers FOR BUSINESS_TIME AS OF '2012-01-01' WHERE id=3

```
ADDRESS
---------------------------------------------------------------------------------------
Zand 89

  1 record(s) selected.
```

SET current temporal business_time='2012-01-01';

SELECT address FROM customers WHERE id=3;

```
ADDRESS
---------------------------------------------------------------------------------------
Zand 89

  1 record(s) selected.
```

SELECT address FROM customers FOR BUSINESS_TIME AS OF '2012-01-01' WHERE id=3

```
SQL20524N  The statement failed because of an invalid period specification or
period clause for period "BUSINESS_TIME". Reason code "7".  SQLSTATE=428HY
```

SET current temporal business_time=NULL;

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: use case 6.7

Product price & availability

```
CREATE TABLE products
( prid          INTEGER  NOT NULL
, price         DEC(9,2)
, valid_from    date     NOT NULL
, valid_until   date     NOT NULL
, PERIOD BUSINESS_TIME (valid_from, valid_until)
, PRIMARY KEY (prid, BUSINESS_TIME WITHOUT OVERLAPS)
);

CREATE UNIQUE INDEX prid            --  not necessary: is automatically created !
   ON products (prid, BUSINESS_TIME WITHOUT OVERLAPS) ;
```

prid	price	valid_from	valid_until
101	250.00	2004-01-01	9999-12-30
102	750.00	2012-01-01	9999-12-30
103	150.00	2012-01-01	2013-07-01
103	120.00	2013-07-01	2013-09-01
103	3201.43	2014-01-01	9999-12-30

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Business time: use case

select * from products where prid=103;

```
PRID        PRICE       VALID_FROM VALID_UNTIL
----------- ----------- ---------- -----------
        103      150.00 01/01/2012 07/01/2013
        103      120.00 07/01/2013 09/01/2013
        103     3201.43 01/01/2014 12/30/9999

  3 record(s) selected.
```

select * from products for business_time as of current date where prid=103;

```
PRID        PRICE       VALID_FROM VALID_UNTIL
----------- ----------- ---------- -----------
        103      150.00 01/01/2012 07/01/2013

  1 record(s) selected.
```

select * from products for business_time from '01.01.2013' to '01.01.2014' where prid=103;

```
PRID        PRICE       VALID_FROM VALID_UNTIL
----------- ----------- ---------- -----------
        103      150.00 01/01/2012 07/01/2013
        103      120.00 07/01/2013 09/01/2013

  2 record(s) selected.
```

select * from products for business_time between '01.01.2013' and '01.07.2013' where prid=103;

```
PRID        PRICE       VALID_FROM VALID_UNTIL
----------- ----------- ---------- -----------
        103      150.00 01/01/2012 07/01/2013
        103      120.00 07/01/2013 09/01/2013
```

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Bi-temporal tables

7

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Contain both a system time indication (system maintained)

and a business time indication (application maintained)

“What is valid at time instant X, and when did we know that?”

Necessary to answer questions like:

- **When did we decide on the 20% off promotional price?**
- **What prices did our customers see last week?**

```
ALTER TABLE products
  ADD start       GENERATED ALWAYS AS ROW BEGIN  NOT NULL  implicitly hidden
  ADD end         GENERATED ALWAYS AS ROW END    NOT NULL  implicitly hidden
  ADD trans_id    GENERATED ALWAYS AS TRANSACTION START ID implicitly hidden
;
ALTER TABLE products    ADD PERIOD SYSTEM_TIME(start,end) ;
CREATE TABLE products_history LIKE products ;
ALTER TABLE products  ADD VERSIONING USE HISTORY TABLE products_history;
```

Bi-temporal tables: use cases

What was the 20% reduction timespan, as seen last week?

```
SELECT  prid, valid_from, valid_until
FROM    products AS OF TIMESTAMP current timestamp - 7 days  p
WHERE   price = ( SELECT 0.8*price FROM products WHERE prid = p.prid) ;
```

What price(s) did we announce last week for the summer months?

```
SELECT  prid, price
FROM    products AS OF TIMESTAMP current timestamp - 7 days
                  FOR BUSINESS_TIME FROM '2013-07-01' TO '2013-09-01' ;
```

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Further reading

Very good summary in Chapter 4 of the eFlashBook

“DB2 10 for Linux, UNIX, and Windows New Features”

(Paul Zikopoulos et al.), see
`http://public.dhe.ibm.com/common/ssi/ecm/en/imm14091usen/IMM14091USEN.PDF`

Syntax details: to be found in the SQL reference manuals:

(search for “period-specification” and “row-transaction” in the syntax diagrams)

LUW: documents SC27-3885 and SC27-3886,
`http://public.dhe.ibm.com/ps/products/db2/info/vr101/pdf/en_US/DB2SQLRefVol1-db2s1e1011.pdf`
and `DB2SQLRefVol2-db2s2e1011.pdf`

z/OS: document SC19-2983, `http://publib.boulder.ibm.com/epubs/pdf/dsnsqm08.pdf`

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading

Questions, remarks, feedback, ... ?

TRAINING & CONSULTING

Thank you!

Peter Vanroose

ABIS Training & Consulting

pvanroose@abis.be

DB2 10 temporal data features

1. Relational databases and historic (or versioned) data
2. New SELECT query syntax for “temporal” requests
3. Table setup for “system time” versioning
4. Interpretation of system time validity intervals
5. System time: some use cases
6. Business time: data validity time period
7. Bi-temporal tables
8. Further reading