IBM DB2 for i
Temporal Database Support

Doug Mack
IBM Lab based Services
mackd@us.ibm.com
Use Cases of Temporal Data Management

*Point in time and period of time queries*

**Track and analyze changes in your business**
- Easily compare data from two points or periods in time
- Increased accuracy in time-based reporting

**Effectively perform and trace data corrections**
- Record when the change was made

**Auditing and compliance**
- Ability to show past data for any point in time
- Ability to show which information got changed in the same transaction
- Ability to show when it was changed
## System Time vs. Business Time

<table>
<thead>
<tr>
<th><strong>System Time</strong></th>
<th><strong>Business Time</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Captures the time when changes happened to data inside DB2</td>
<td>Captures the time when changes happen(ed) to business artifacts</td>
</tr>
<tr>
<td>DB2-generated history of updated or deleted rows</td>
<td>Application-driven changes to the time dimension of business artifacts</td>
</tr>
<tr>
<td>History based on DB2 system timestamps</td>
<td>Dates or timestamps provided by the application</td>
</tr>
<tr>
<td><strong>DB2's physical view of time</strong></td>
<td><strong>Your application's logical view of time</strong></td>
</tr>
<tr>
<td>Spans from the past to the present time</td>
<td>Spans past, present, and future time</td>
</tr>
<tr>
<td>System validity (&quot;transaction time&quot;)</td>
<td>Business validity (&quot;valid time&quot;)</td>
</tr>
<tr>
<td>Supports queries such as:</td>
<td>Supports queries such as:</td>
</tr>
<tr>
<td>&quot;Which policies were in DB on June 30?&quot;</td>
<td>&quot;Which policies were active on June 30?&quot;</td>
</tr>
</tbody>
</table>

*Not Available in IBM i 7.3*
Illustration – Without Temporal Support
Illustration – Without Temporal Support

Table

row1

t1: INSERT
Illustration – Without Temporal Support

Time

Table

row1

row1

t2: UPDATE
Illustration – Without Temporal Support

Time

Table

row1

row1

row1

\text{t3: UPDATE}
Illustration – Without Temporal Support

Table

Row 1

DELETED

Time

t4: DELETE
Illustration – Without Temporal Support

After 4 transactions, what is the user’s point of view?

Table

“Current only”
Illustration – With Temporal Support

Time

Table

Table History
Illustration – With Temporal Support

Table History

Table

Row 1

$t_1$: INSERT

Time
Illustration – With Temporal Support

Table History

Before t2

Table

Time

t2: UPDATE

row1

Table History

row1

Table

row1

Before t2

Time

t2: UPDATE

row1
Illustration – With Temporal Support

Table History

Before t2
Before t3

Table

Time

row1

row1

row1

t3: UPDATE
Illustration – With Temporal Support

Table

Table History

Time

- Row 1: Before t2
- Row 2: Before t3
- Row 3: Before t4
- Row 1: t4: DELETE

DELETED

- Row 1: Deleted row1
- Row 1: Before t4

- Row 1: Before t4

- Row 1: Before t4

- Row 1: Before t4
After 4 transactions, what is the user’s point of view?

Illustration – With Temporal Support

Table

“Current only”

“Current + History”

Table History

“History only”

All managed by DB2!
IBM i 7.3 – DB2 for i Enhancements (Temporal)

**New Generated Columns**
- ROW BEGIN (birth)
- ROW END (death)
- TRANSACTION START ID
- DATA CHANGE OPERATION

**New Catalogs**
- QSYS2/SYSPERIODS
- QSYS2/SYSHISTORYTABLES

**New SET OPTION**
- SYSTIME (*YES | *NO)

**New Special Register**
- CURRENT TEMPORAL SYSTEM_TIME

**New Query period-specification**
- FOR SYSTEM TIME AS OF <value>
- FOR SYSTEM TIME FROM <value> TO <value>
- FOR SYSTEM TIME BETWEEN <value> AND <value>
Defining a System-period Temporal Table

- The **row begin** column represents the time when the row data became current
  - This is an inclusive value for the system-time period
  - `TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN`
- The **row end** column represents the time when the row data ceased to be current
  - This is an exclusive value for the system-time period
  - `TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END`
- The **transaction start ID** column contains the unique timestamp of the first data change in the transaction that produced the row
  - `TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID`
- The **data change operation** column contains a value to represent the operation
  - `I = row was inserted, U = row was updated, D = row was deleted (shown in history)`
  - `CHAR (1) NOT NULL GENERATED ALWAYS AS (DATA CHANGE OPERATION)`
- A system-period temporal table includes a **system-time period definition** with columns that capture the row begin and row end times that indicate when the data in the row is current
  - This period is used to preserve historical versions of rows (in the history table) whenever updates or deletes occur

- `CREATE TABLE <history> LIKE` is used to manifest the history table
- An SQL table becomes a system-period temporal table when `ALTER TABLE ADD VERSIONING` statement is successfully executed
Defining a New System-Period Temporal Table

```sql
CREATE TABLE employees
(empID INTEGER NOT NULL PRIMARY KEY,
department VARCHAR(50),
...
system_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
system_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
trans_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID,
op_id CHAR(1) NOT NULL GENERATED ALWAYS AS (DATA CHANGE OPERATION),
PERIOD SYSTEM_TIME (system_start, system_end));

CREATE TABLE employees_history
LIKE employees;

ALTER TABLE employees ADD VERSIONING
USE HISTORY TABLE employees_history;
```
Altering an Existing Table to add System Time

Existing table *has no timestamp columns*

```sql
CREATE TABLE employees
(empID INTEGER NOT NULL PRIMARY KEY,
depart VARCHAR(50) );

ALTER TABLE employees
ADD COLUMN sys_begin TIMESTAMP(12) NOT NULL
GENERATED AS ROW BEGIN IMPLICITLY HIDDEN
ADD COLUMN sys_end TIMESTAMP(12) NOT NULL
GENERATED AS ROW END IMPLICITLY HIDDEN
ADD COLUMN trans_id TIMESTAMP(12) NOT NULL
GENERATED AS TRANSACTION START ID IMPLICITLY HIDDEN
ADD PERIOD SYSTEM_TIME (sys_begin, sys_end);

...
Temporal in the DB2 Catalogs

QSYS2.SYSPERIODS

- All temporal tables and their period columns
- The names of the associated history tables

```
SELECT table_name, period_name, begin_column_name, end_column_name, history_table_name
FROM qsys2.sysperiods
WHERE table_name = 'EMPLOYEES';
```
New in the DB2 Catalogs

QSYS2.SYSHISTORYTABLES

- The names of the associated history tables

```
SELECT table_name, 
       period_name, 
       history_table_name 
FROM qsys2.syshistorytables 
WHERE table_name = 'EMPLOYEES';
```
Schema Evolution

Schema changes that cannot cause loss of history are automatically propagated from the base table to the history table:

- `ALTER TABLE employees ADD COLUMN salary(INTEGER);`
  - New column automatically also added to history table!

- `ALTER TABLE employees ALTER COLUMN dept SET DATA TYPE VARCHAR(90);`
  - No data loss!
  - Column change applied base table and history table

- `ALTER TABLE employees ALTER COLUMN dept SET DATA TYPE VARCHAR(2);`
  - Blocked due to potential data loss! (SQL0190)
  - Must stop versioning before making this change

- `ALTER TABLE employees DROP COLUMN dept;`
  - Blocked due to potential data loss! (SQL0196)
  - Must stop versioning before making this change

- `DROP TABLE employees;`
  - Both base table and history table are deleted!

- `DROP TABLE employees_history;`
  - Blocked due to potential data loss! (SQL0156)
  - Must stop versioning before making this change
Insert and Update

On 11/15/2014, Employee 12345 and 67890 were hired into the department J13 & K25.

\[\text{INSERT INTO employees (empID, dept) VALUES (12345,'J13'), (67890, 'K25')}\]

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Dept</th>
<th>System_start</th>
<th>System_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>J13</td>
<td>11/15/2014</td>
<td>12/30/9999</td>
</tr>
<tr>
<td>67890</td>
<td>K25</td>
<td>11/15/2014</td>
<td>12/30/9999</td>
</tr>
</tbody>
</table>

On 1/31/2015, Employee 12345 moved to department M24.

\[\text{UPDATE employees SET dept = 'M24' WHERE empID = 12345}\]

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Dept</th>
<th>System_start</th>
<th>System_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>M24</td>
<td>01/31/2015</td>
<td>12/30/9999</td>
</tr>
<tr>
<td>67890</td>
<td>K25</td>
<td>11/15/2014</td>
<td>12/30/9999</td>
</tr>
</tbody>
</table>

Note: only date portion of TIMESTAMP value shown in examples to simplify display
## Delete and Update

On 3/31/2016, Employee 67890 left the company.

\[
\text{DELETE FROM employees WHERE empID = 67890}
\]

---

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Dept</th>
<th>System_start</th>
<th>System_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>M24</td>
<td>01/31/2015</td>
<td>12/30/9999</td>
</tr>
</tbody>
</table>

---

67890 was in K25 from 11/15/2014 to 3/31/2016

---

On 5/31/2016, Employee 12345 joined the department M15.

\[
\text{UPDATE employees SET dept = 'M15' WHERE empID = 12345}
\]

---

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Dept</th>
<th>System_start</th>
<th>System_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>M15</td>
<td>05/31/2016</td>
<td>12/30/9999</td>
</tr>
</tbody>
</table>

---

12345 was in M24 from 1/31/2015 to 5/31/2016
Specifying the Time Period for Queries

- A period is an interval of time that is defined by two date or timestamp columns in a temporal table.
- A period contains a begin column and an end column.
- The begin column indicates the beginning of the period and the end column indicates the end of the period.
- DB2 manages all system time periods as **inclusive-exclusive periods**.
  - Using inclusive-exclusive periods makes it very easy to detect or avoid gaps between time periods.

For querying, there is the notion of: **explicit and implicit period specifications**.

Explicitly including a system-time period-specification on a table reference for a non-temporal table is an error.
Specifying the Time Period for Queries

No time period specification
Scope = current

Time period specification
Scope = current + history

△ = previous state
Specifying the Time Period for Queries

Time period specification
Scope = current + history

The “same” row can show up more than once in the set
Specifying the Time Period

- **Explicit** Period Specification
  - FOR SYSTEM_TIME
    - AS OF value
    - FROM value1 to value2
    - BETWEEN value1 AND value2

- **Implicit** Period Specification
  - An implicit period specification is affected by:
    - CURRENT TEMPORAL SYSTEM_TIME special register
    - SYSTIME bind option
  - (AS OF CURRENT TEMPORAL SYSTEM_TIME) implicitly defined

  Note: for a native HLL open of a temporal table or view based on a temporal table the CURRENT TEMPORAL SYSTEM_TIME special register does not apply and is effectively ignored, thus historical rows are not accessed
1. Which department is employee 12345 in (right now)?

   SELECT dept
   FROM employees
   WHERE empID = 12345;

   Without the FOR SYSTEM_TIME clause, query reads the current data only

2. Which department was employee 12345 in on 12/01/2014?

   SELECT dept
   FROM employees
   FOR SYSTEM_TIME AS OF '12/01/2014'
   WHERE empID = 12345;

3. How many departments has employee 12345 worked in since 2014?

   SELECT COUNT(DISTINCT dept)
   FROM employees
   FOR SYSTEM_TIME FROM '2014-01-01' TO CURRENT_TIMESTAMP
   WHERE empID = 12345;
Query Plan – The Union of Two Sets (Current and History)

Diagram:
- Final Select
- Union all
- Table Probe EXAMPLE.EMPLOYEES
- Table Scan EXAMPLE.EMPLOYEES_HISTORY
- Index Probe EXAMPLE.EMPLOYEES

The Union of Two Sets (Current and History)
System Time Special Register Considerations

When `CURRENT TEMPORAL SYSTEM_TIME` special register is set to a non-null value:

- Insert, Update, Delete operations on system-period temporal tables are blocked! (SQ20535)
- Queries will implicitly invoke the time period
  - FOR SYSTEM TIME AS OF CURRENT TEMPORAL SYSTEM_TIME
- No "stacking" of system-time specification
  - Set the desired system time *either* in the query *or* with the special register, not both!
  - The following will result in an error (SQ20524):

```
SET CURRENT TEMPORAL SYSTEM_TIME '12/01/2014';
```

```
SELECT dept
FROM employees FOR SYSTEM_TIME AS OF '01/01/2015'
WHERE empID=12345;
```
CREATE VIEW v_salary_M15 AS
SELECT empID, salary,
FROM employees
WHERE dept = 'M15';

SET CURRENT TEMPORAL SYSTEM_TIME '2015-02-15';
SELECT * FROM v_salary_M15;

SET CURRENT TEMPORAL SYSTEM_TIME '2015-11-01';
SELECT * FROM v_salary_M15;
Temporal Considerations

- Data modeling with temporal in mind – multiple instances of data
- Data integrity and transaction boundaries when base table and history table are in play
- Perspective of the data must always be clear and concise – “incorrect” output is possible
- Data life cycle must be well understood
- Increased probability of very large data sets
- Performance and scalability (UNION of 2 potentially large data sets)
- **Only SQL** query requests allow transparent inclusion and access of history
- Data governance and control – multiple instances of data must be secured

*A Database Engineer is Required!*
Summary

One Logical Set of Data

Managed by DB2

Time Travel

Current State

History Table

Historical State

Table

Summary
Any Questions?
Thank You!