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

System Time	Business Time
Captures the time when changes happened to data inside DB2	Captures the time when changes happen(ed) to business artifacts
DB2-generated history of updated or deleted rows	Application-driven ch me dimension
History based on DB2 system timestamps	Dat appl
DB2's physical view of time	**Your e**
Spans from the past to the present time	Spans
System validity ("transaction time")	Busines
Supports queries such as: *"Which policies **were in DB** on June 30?"*	Supports queries such as: *"Which policies **were active** on June 30?"*

Not Available in IBM i

IBM i 7.3

Illustration – Without Temporal Support

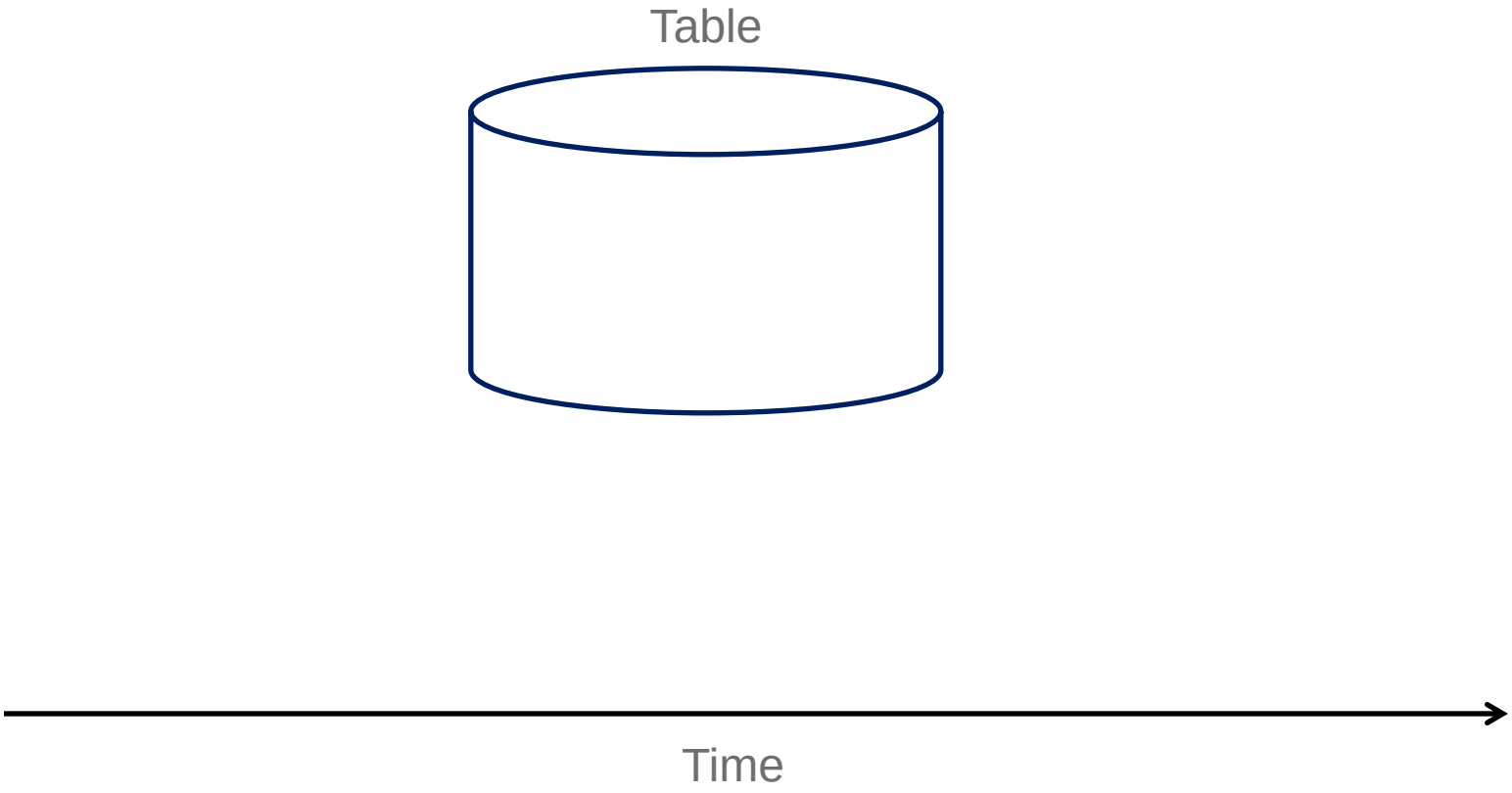


Illustration – Without Temporal Support

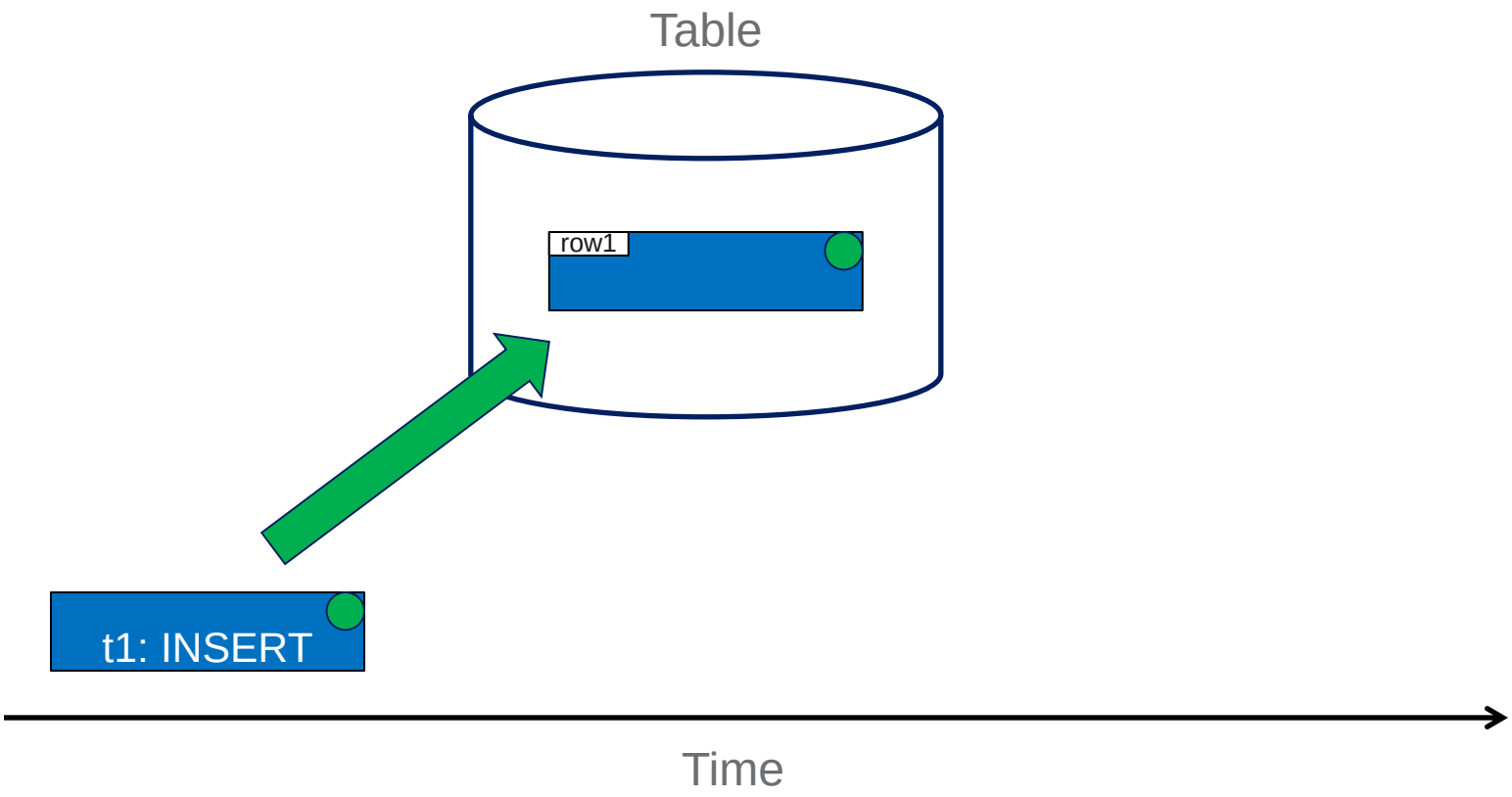


Illustration – Without Temporal Support

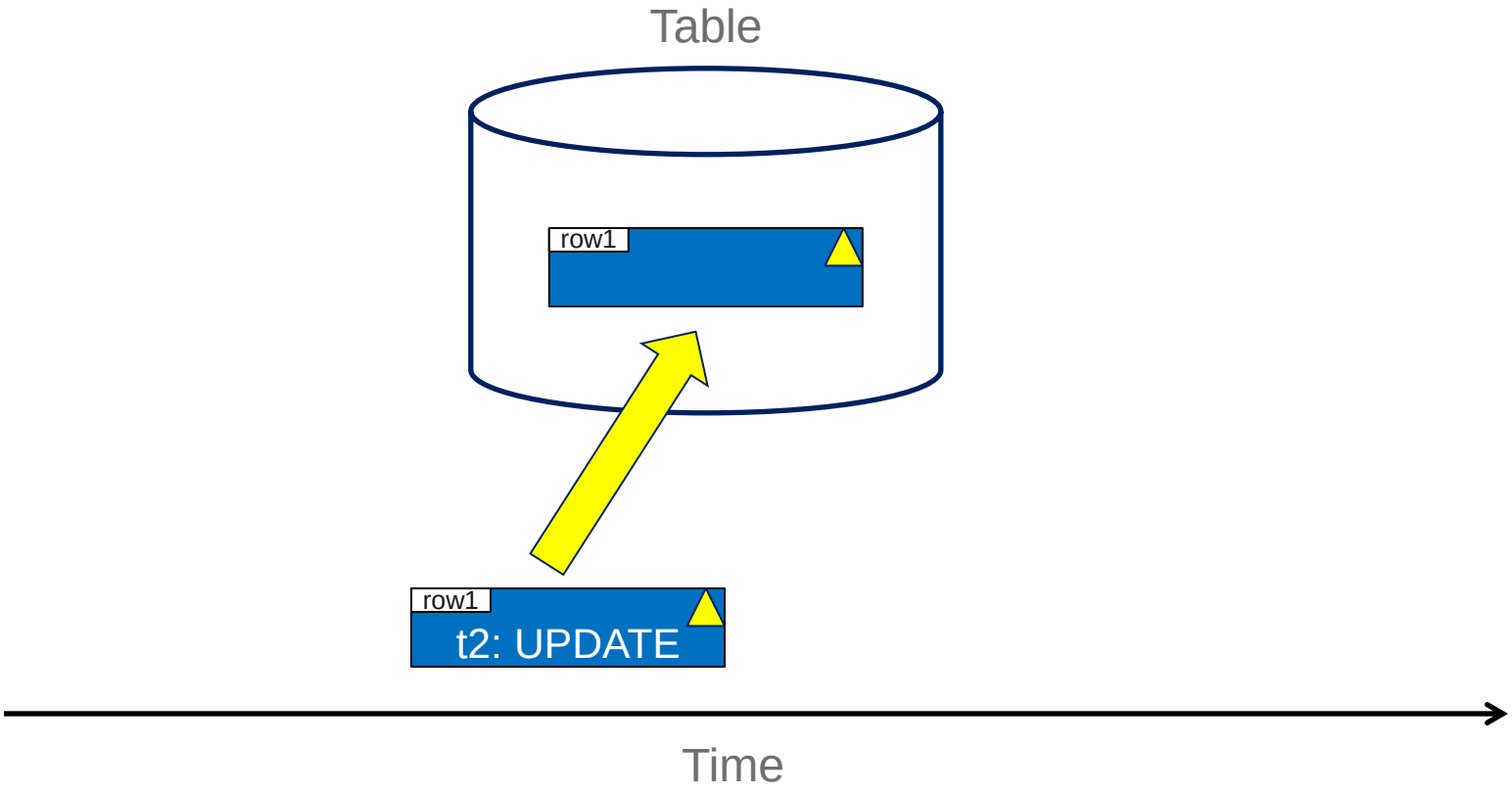


Illustration – Without Temporal Support

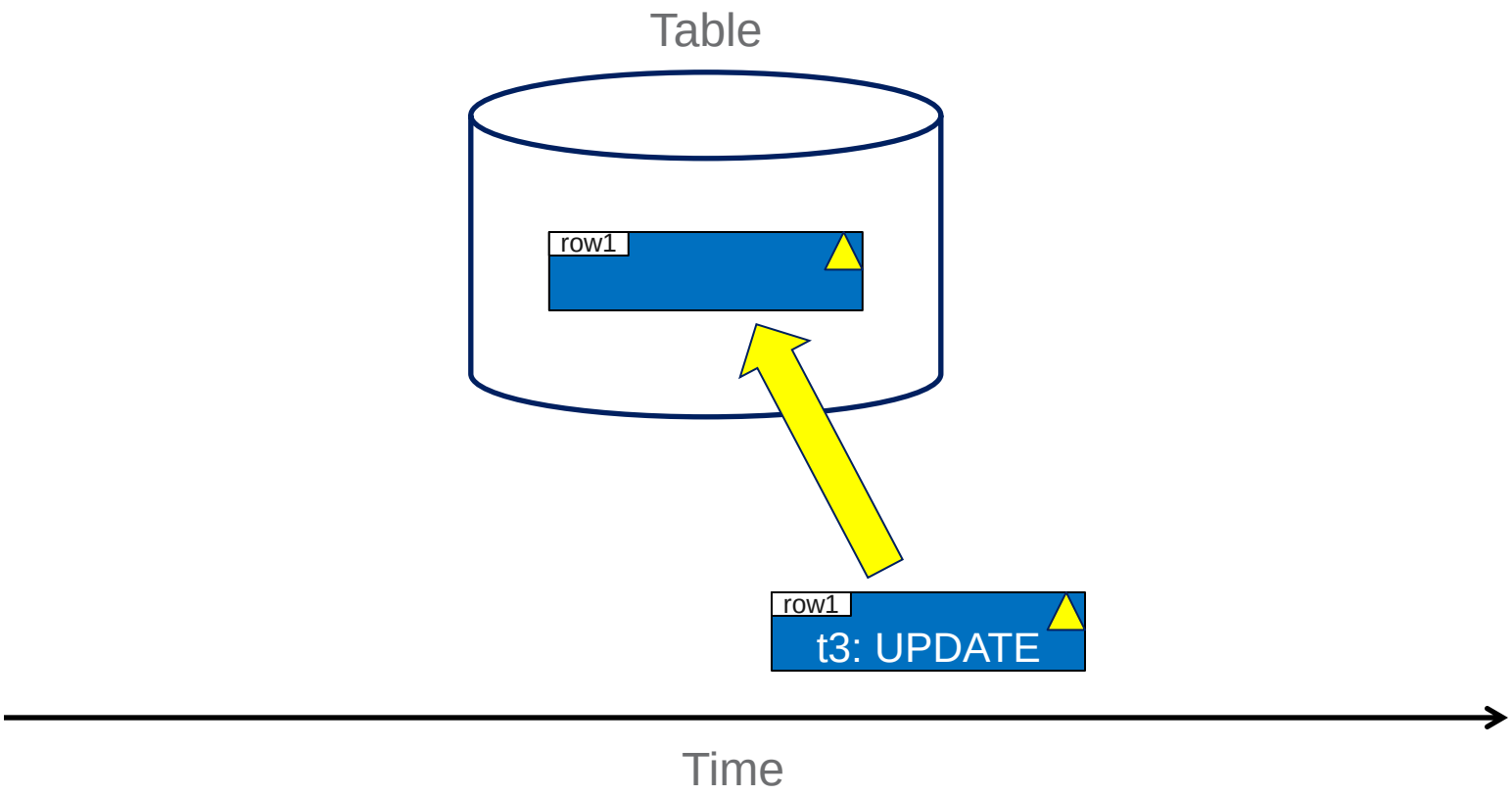


Illustration – Without Temporal Support

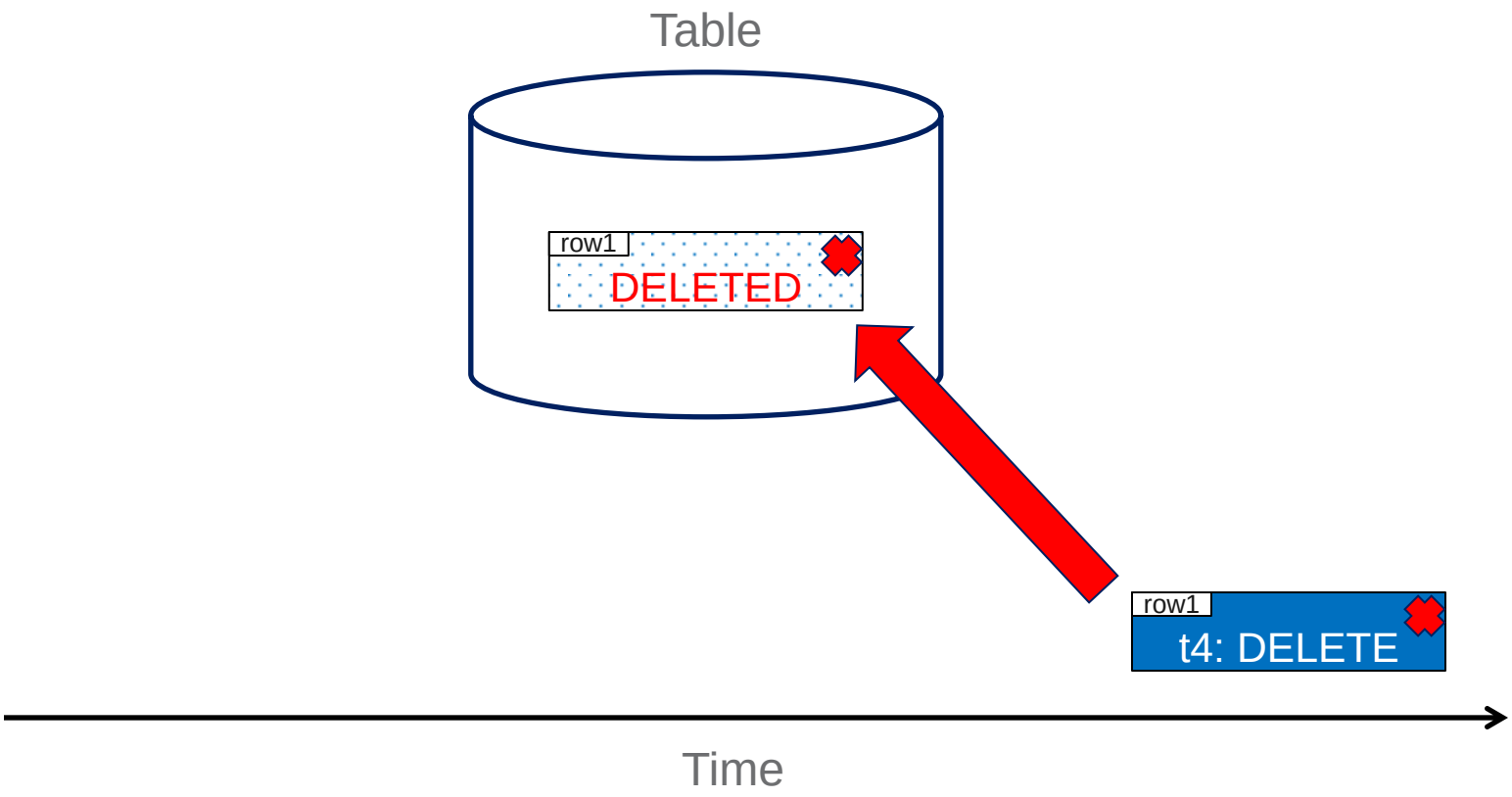


Illustration – Without Temporal Support

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

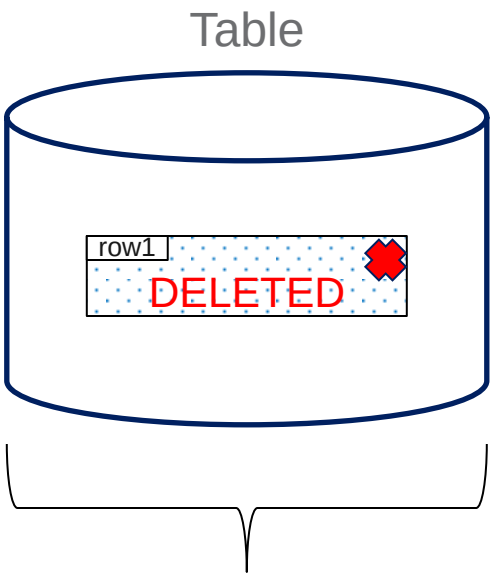


Illustration – With Temporal Support

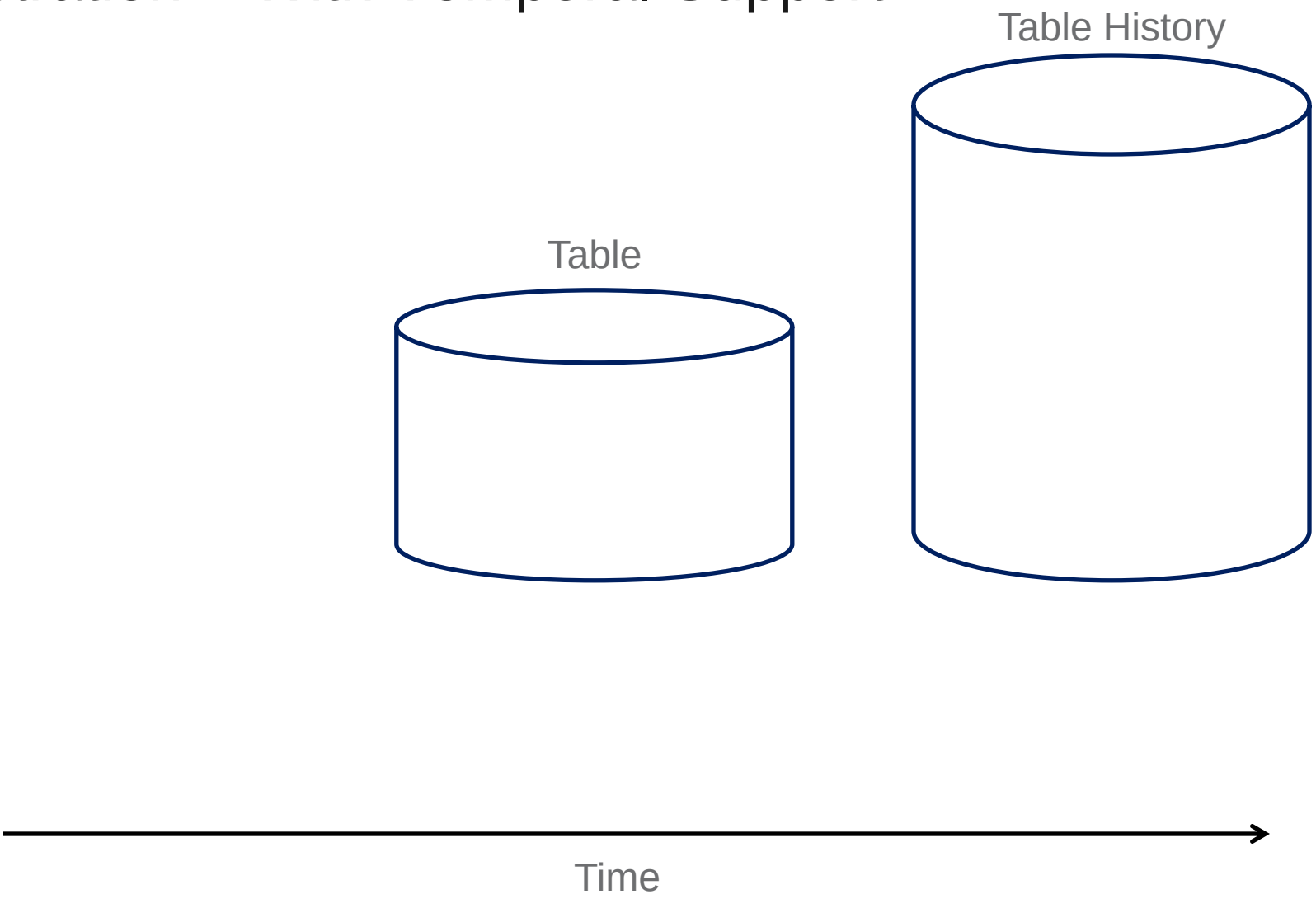


Illustration – With Temporal Support

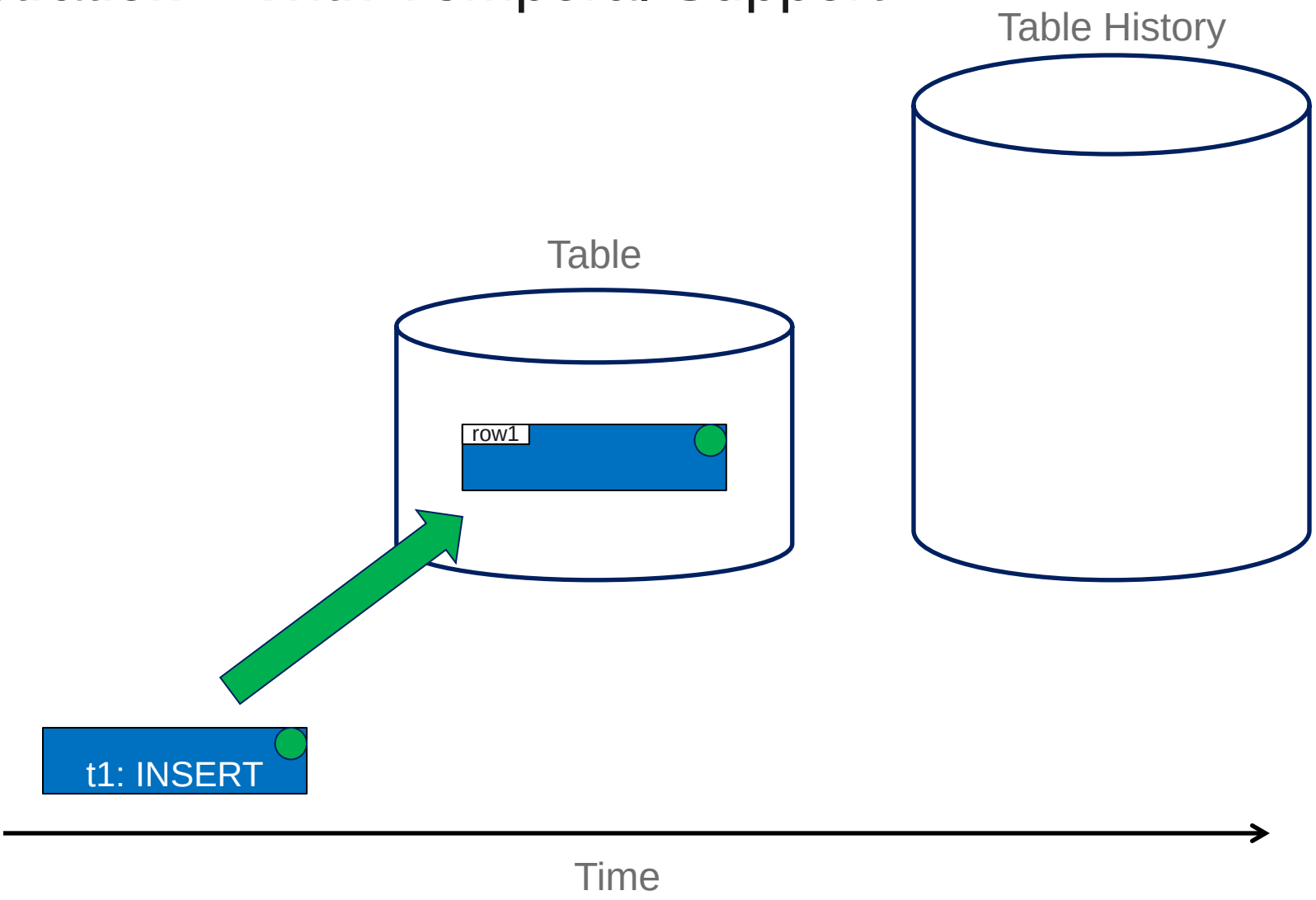


Illustration – With Temporal Support

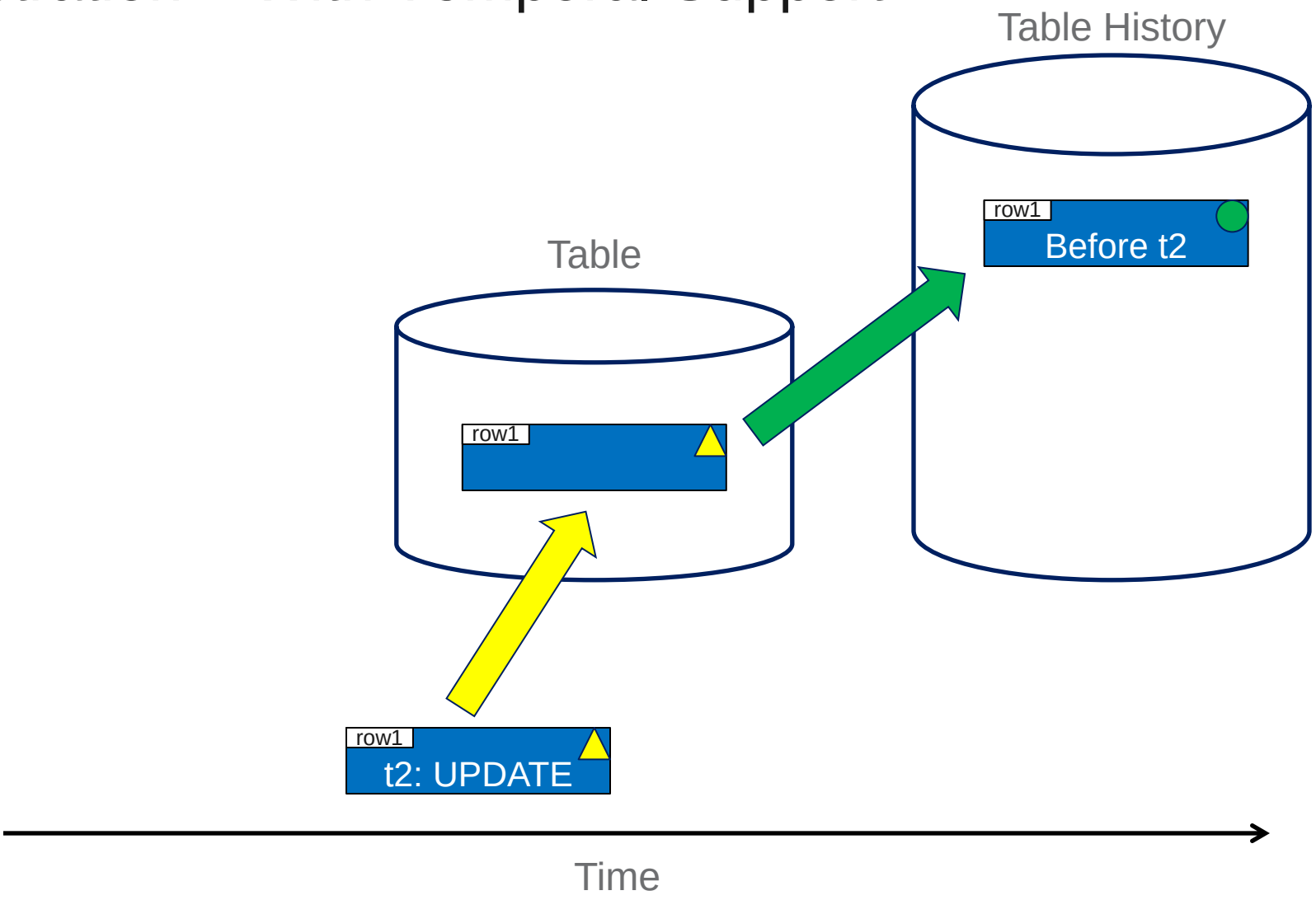


Illustration – With Temporal Support

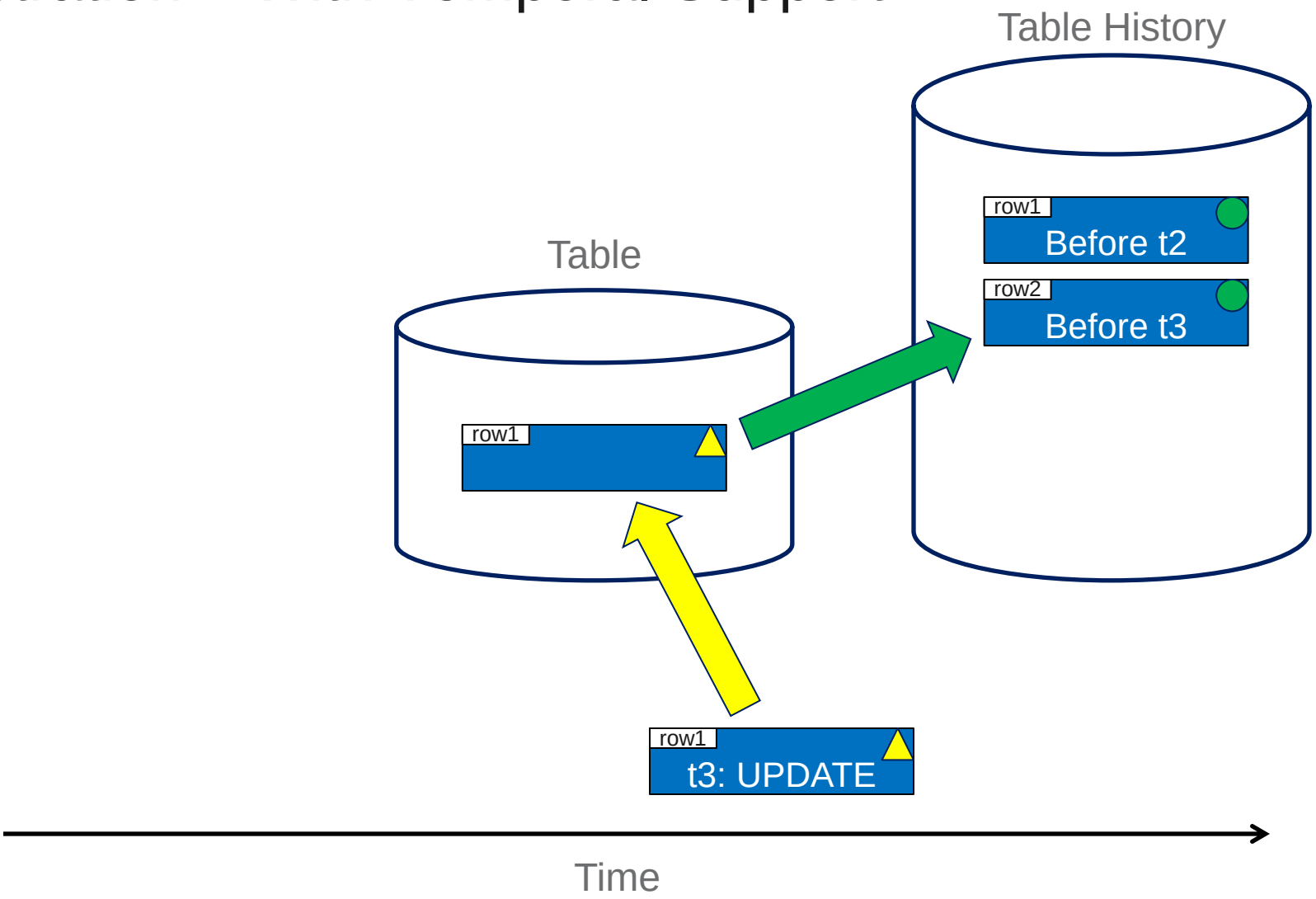


Illustration – With Temporal Support

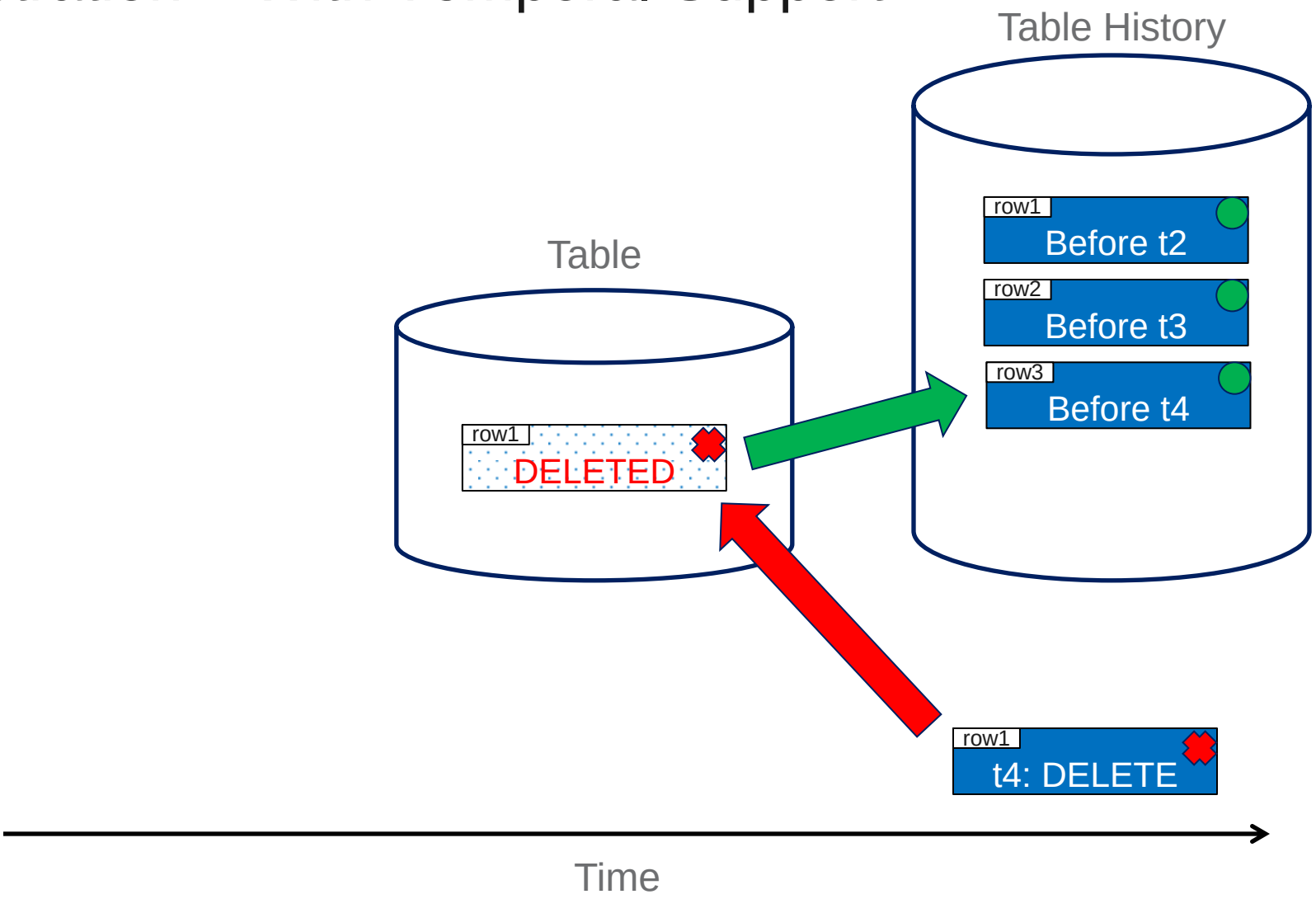
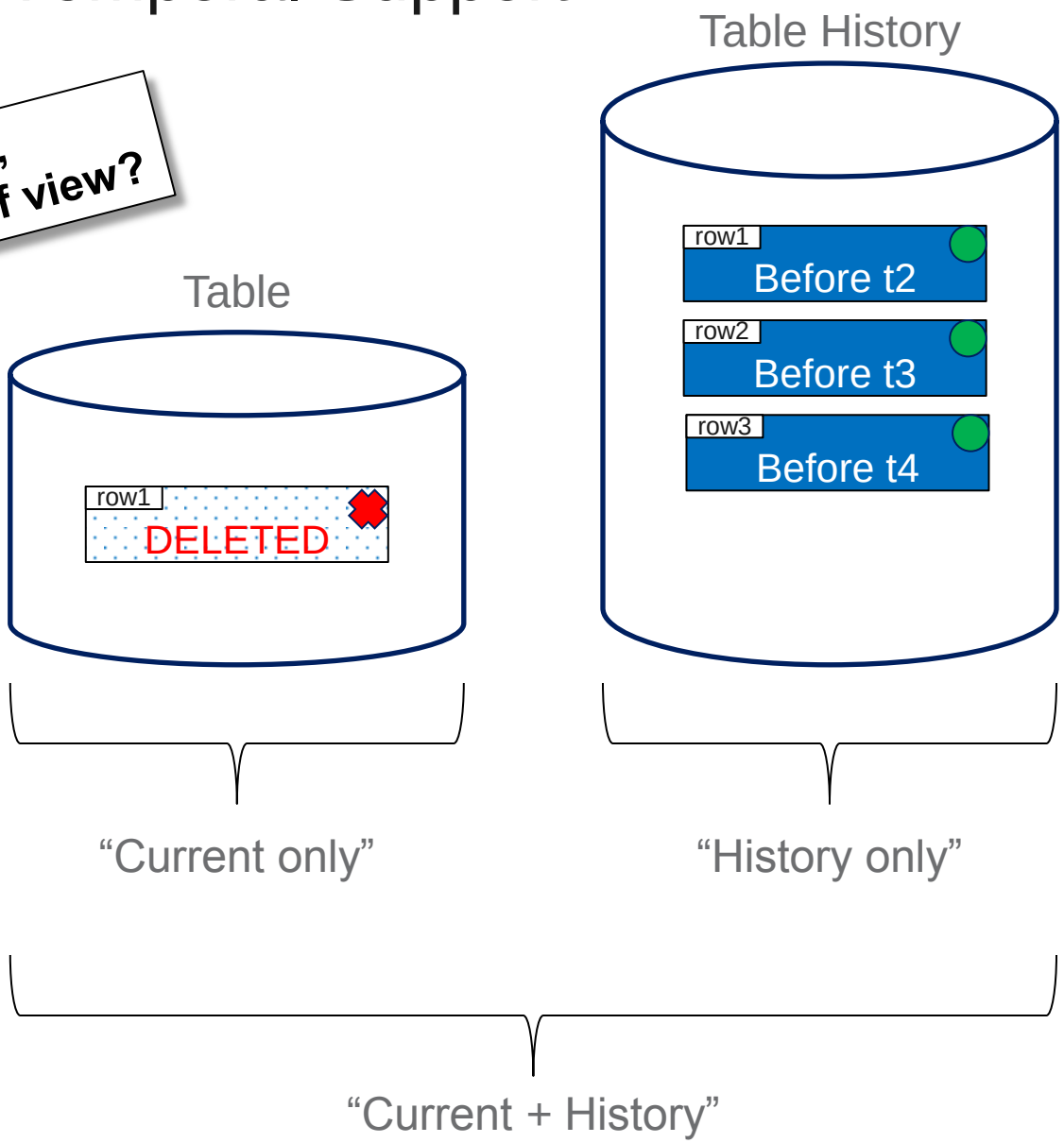


Illustration – With Temporal Support

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

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

```
CREATE TABLE employees
(empID        INTEGER NOT NULL PRIMARY KEY,
 dept         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

```
CREATE TABLE employees
(empID          INTEGER NOT NULL PRIMARY KEY,
 dept           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

Column Name
PERIOD_NAME
TABLE_SCHEMA
TABLE_NAME
BEGIN_COLUMN_NAME
END_COLUMN_NAME
PERIOD_TYPE
HISTORY_TABLE_SCHEMA
HISTORY_TABLE_NAME
ON_DELETE_ADD_EXTRA_ROW
VERSIONING_STATUS
SYSTEM_TABLE_SCHEMA
SYSTEM_TABLE_NAME
SYSTEM_HISTORY_TABLE_SCHEMA
SYSTEM_HISTORY_TABLE_NAME
SYSTEM_BEGIN_COLUMN_NAME
SYSTEM_END_COLUMN_NAME

```
SELECT table_name,
       period_name,
       begin_column_name,
       end_column_name,
       history_table_name
FROM   qsys2.sysperiods
WHERE  table_name = EMPLOYEES';
```

SELECT table_name, period_name, begin_column_name, end_column_name, history_table_name ... - Db2icoe2.rchland.ibm.com(Db2icoe2)

TABLE_NAME	PERIOD_NAME	BEGIN_COLUMN_NAME	END_COLUMN_NAME	HISTORY_TABLE_NAME
EMPLOYEES	SYSTEM_TIME	SYSTEM_START	SYSTEM_END	EMPLOYEES_HISTORY

New in the DB2 Catalogs

QSYS2.SYSHISTORYTABLES

- The names of the associated history tables

Column Name
HISTORY_TABLE_SCHEMA
HISTORY_TABLE_NAME
VERSIONING_STATUS
PERIOD_NAME
TABLE_SCHEMA
TABLE_NAME
SYSTEM_HISTORY_SCHEMA
SYSTEM_HISTORY_TABLE_NAME
SYSTEM_TABLE_SCHEMA
SYSTEM_TABLE_NAME

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

SELECT table_name, period_name, history_table_name FROM qsys2.syshistorytables WHERE t ... - Db2icoe2.rchland

TABLE_NAME	PERIOD_NAME	HISTORY_TABLE_NAME
EMPLOYEES	SYSTEM_TIME	EMPLOYEES_HISTORY

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.

INSERT INTO employees (empID, dept) VALUES (12345,'J13'), (67890, 'K25')

Table: employees

EmpID	Dept	System_start	System_end
12345	J13	11/15/2014	12/30/9999
67890	K25	11/15/2014	12/30/9999

system_time values are always set by DB2!

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

UPDATE employees SET dept = 'M24' WHERE empID = 12345

Table: employees

EmpID	Dept	System_start	System_end
12345	M24	01/31/2015	12/30/9999
67890	K25	11/15/2014	12/30/9999

Table: employees_history

EmpID	Dept	System_start	System_end
12345	J13	11/15/2014	01/31/2015

System validity period: [inclusive, exclusive[

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.

DELETE FROM employees WHERE empID = 67890

Table: employees

EmpID	Dept	System_start	System_end
12345	M24	01/31/2015	12/30/9999

Table: employees_history

EmpID	Dept	System_start	System_end
12345	J13	11/15/2014	01/31/2015
67890	K25	11/15/2014	03/31/2016

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

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

UPDATE employees SET dept = 'M15' WHERE empID = 12345

Table: employees

EmpID	Dept	System_start	System_end
12345	M15	05/31/2016	12/30/9999

Table: employees_history

EmpID	Dept	System_start	System_end
12345	J13	11/15/2014	01/31/2015
12345	M24	01/31/2015	05/31/2016
67890	K25	11/15/2014	03/31/2016

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

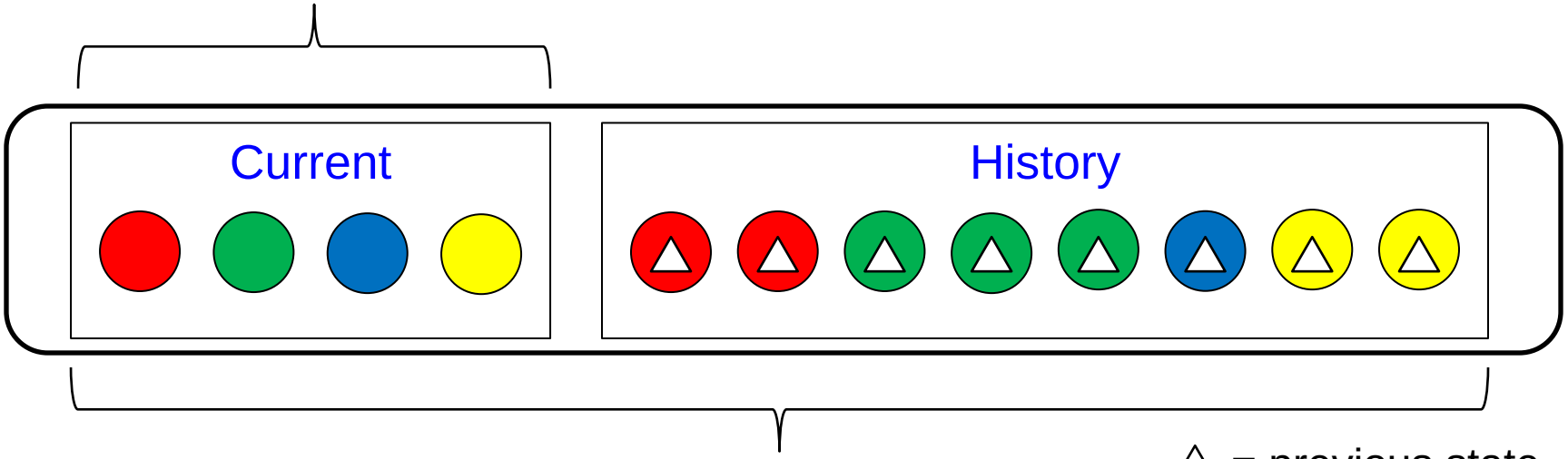
valid up to and including **2016-05-31**

EMPID	DEPT	SALARY	SYS_BEGIN	SYS_END
67890	M15	7000	2016-01-01	**2016-06-01**
67890	M15	7500	**2016-06-01**	9999-12-30

- 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



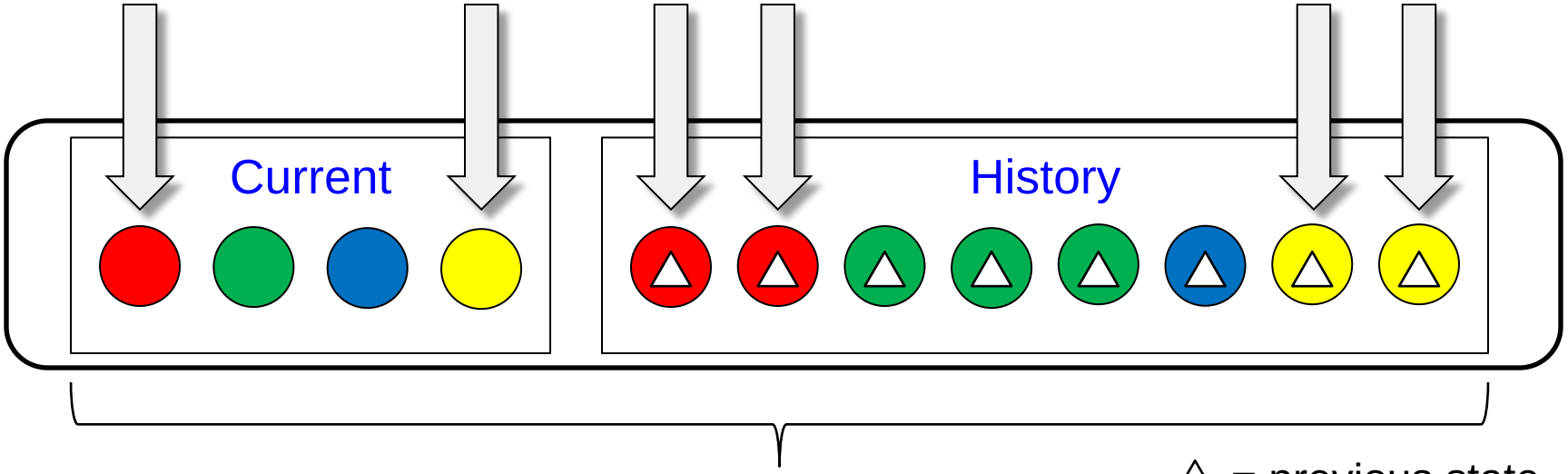
△ = previous state

Time period specification
Scope = current + history

Specifying the Time Period for Queries

Time travel

```
select COLOR, count(*)
…
where COLOR in ('RED', 'YELLOW')
group by COLOR
```



△ = previous state

Time period specification
Scope = current + history

The "same" row can show up more than once in the set

Specifying the Time Period

In the query

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

In the job or session

- **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

Querying a System-period temporal table

Table: employees

EmpID	Dept	System_start	System_end
12345	M15	05/31/2016	12/30/9999

Table: employees_history

EmpID	Dept	System_start	System_end
12345	J13	11/15/2014	01/31/2015
12345	M24	01/31/2015	05/31/2016
67890	K25	11/15/2014	03/31/2016

1. Which department *is* employee 12345 in (*right now*)?

   M15

   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?

   J13

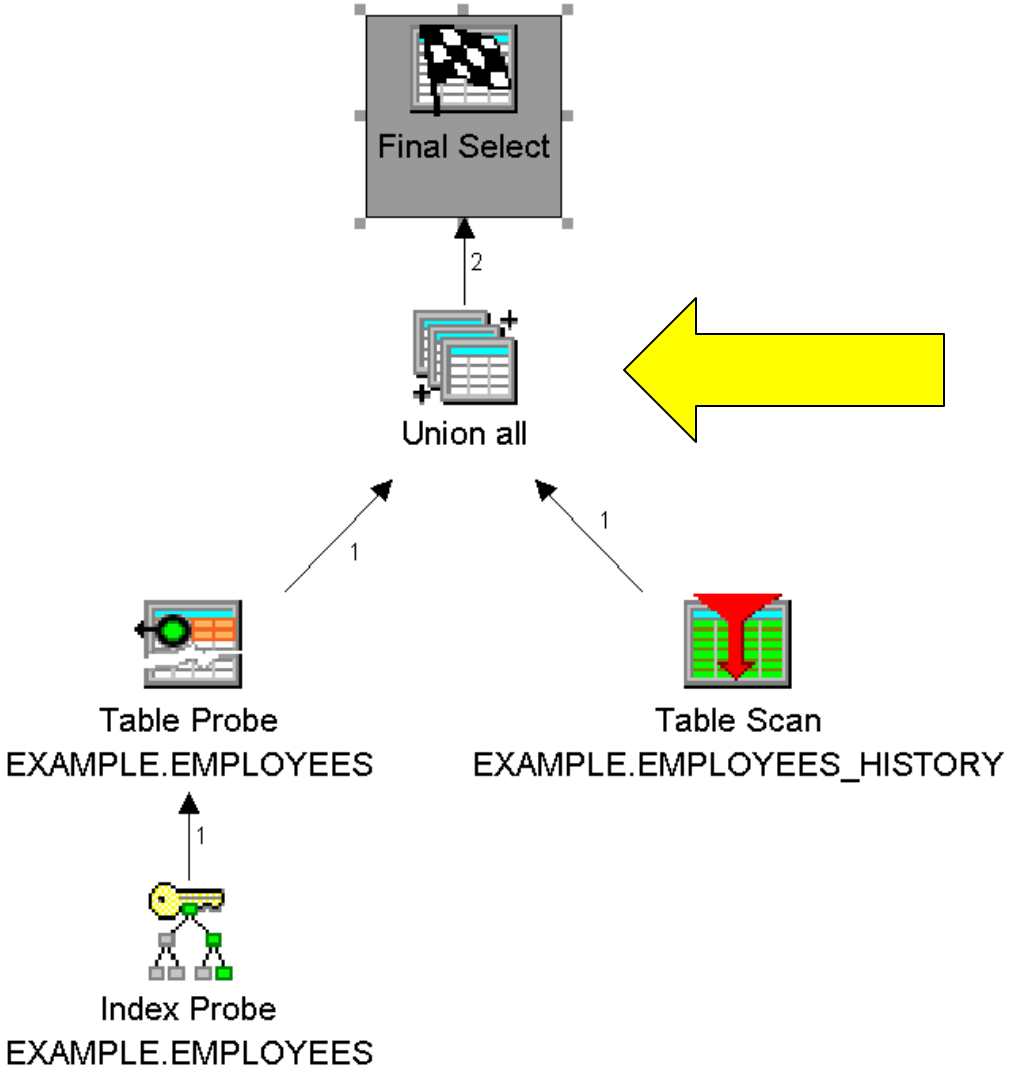
   SELECT dept
   FROM ***employees*** *FOR SYSTEM_TIME AS OF '12/01/2014'*
   WHERE empID=12345;

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

   3

   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)



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 <u>implicitly</u> 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;
```

System Time Special Register Affects Views Too

empID	dept	salary	sys_end	sys_end
12345	J13	5000	2015-01-01	2016-01-01
67890	M15	7000	2015-01-01	2015-06-01
67890	M15	7500	2015-06-01	9999-12-30

```
CREATE VIEW v_salary_M15 AS
SELECT empID, salary,
FROM employees
WHERE dept = 'M15';
```

Queries against the view:
AS OF clause or special register setting is applied to all temporal tables in the view definition.

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

67890	7000	2011-01-01	2015-06-01

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

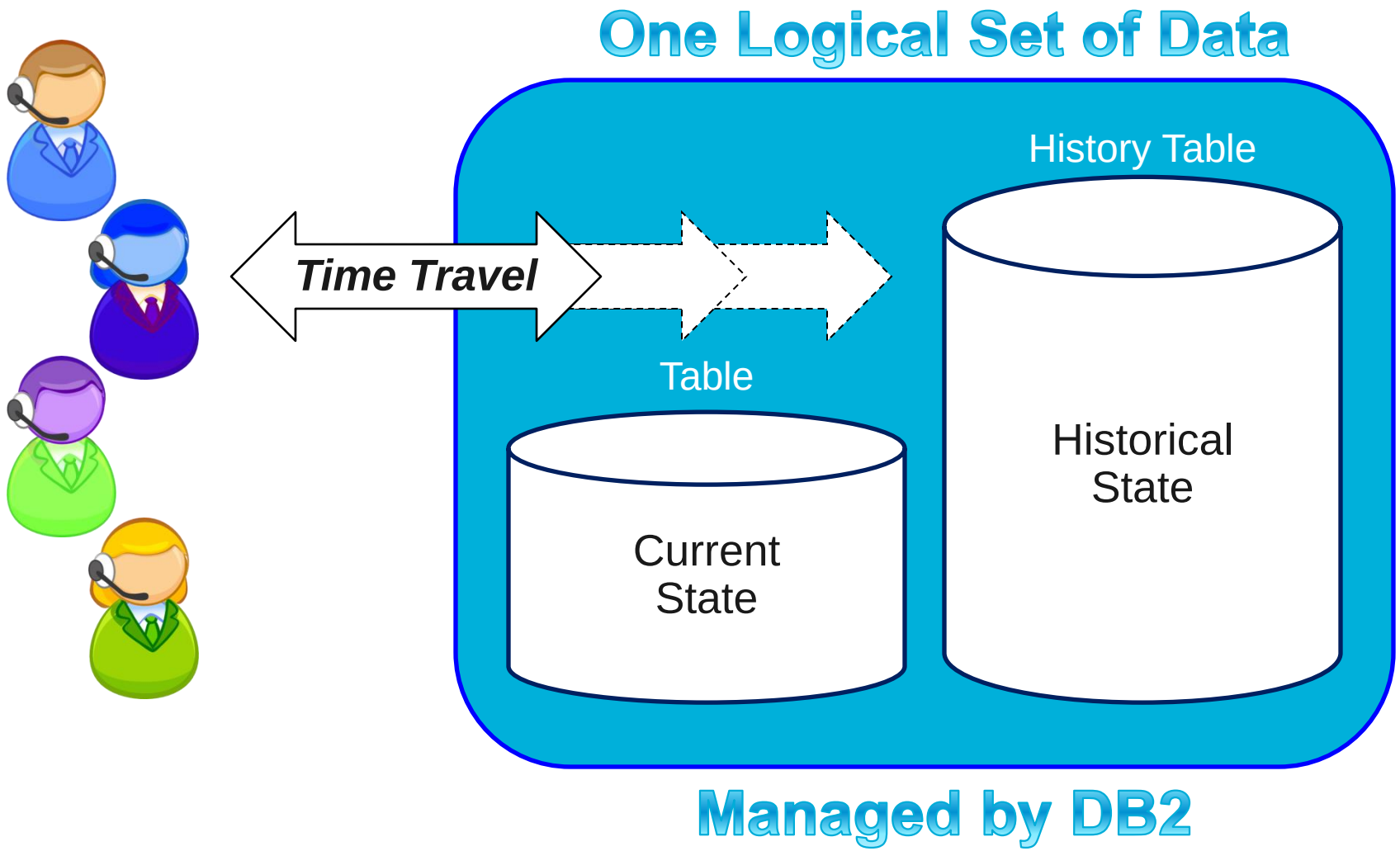
67890	7500	2015-06-01	9999-12-30

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
History Table
Time Travel
Table
Historical
State
Current
State
Managed by DB2

Any Questions?

Thank You!