

Deep Dive into Db2 Temporal Tables

Rob Bestgen
Db2 for i Consultant
bestgen@us.ibm.com



definition (from dictionary.com)

Temporal:

adjective

- of or relating to time
- enduring for a time only; temporary; transitory



From a database perspective, 'temporal' means managing and maintaining time-related data - **rows in a table**

(the temporary part is up to you 😊)

With Temporal Tables, you can answer time-based questions:

- Who was the client rep two years ago?
- Who were the client reps over the last five years?
- Produce an inventory report using a different point in time



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

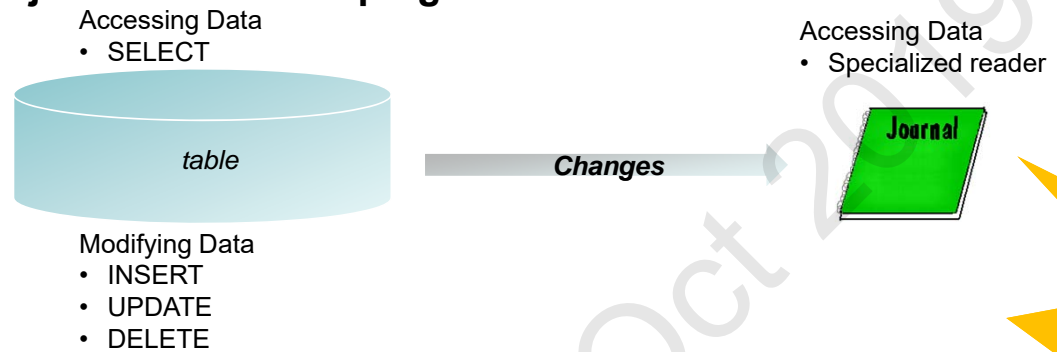
Auditing and compliance

Ability to show past data for any point in time

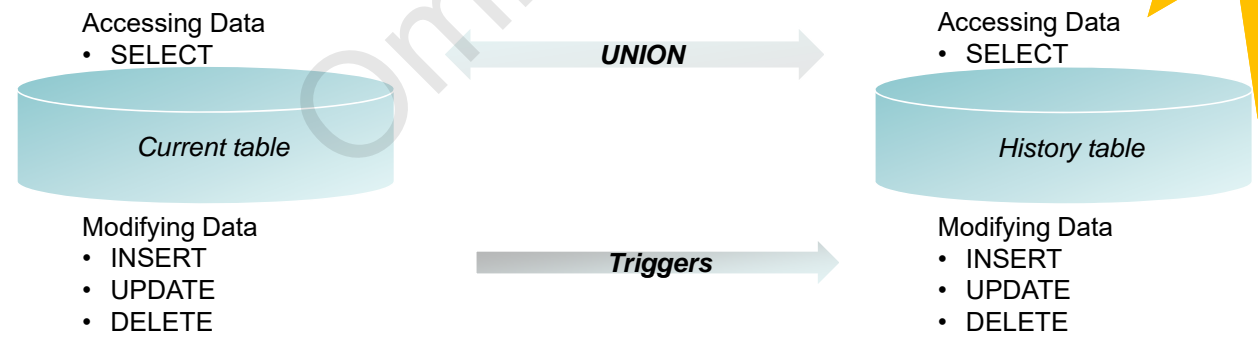
Ability to show which information got changed in the same transaction

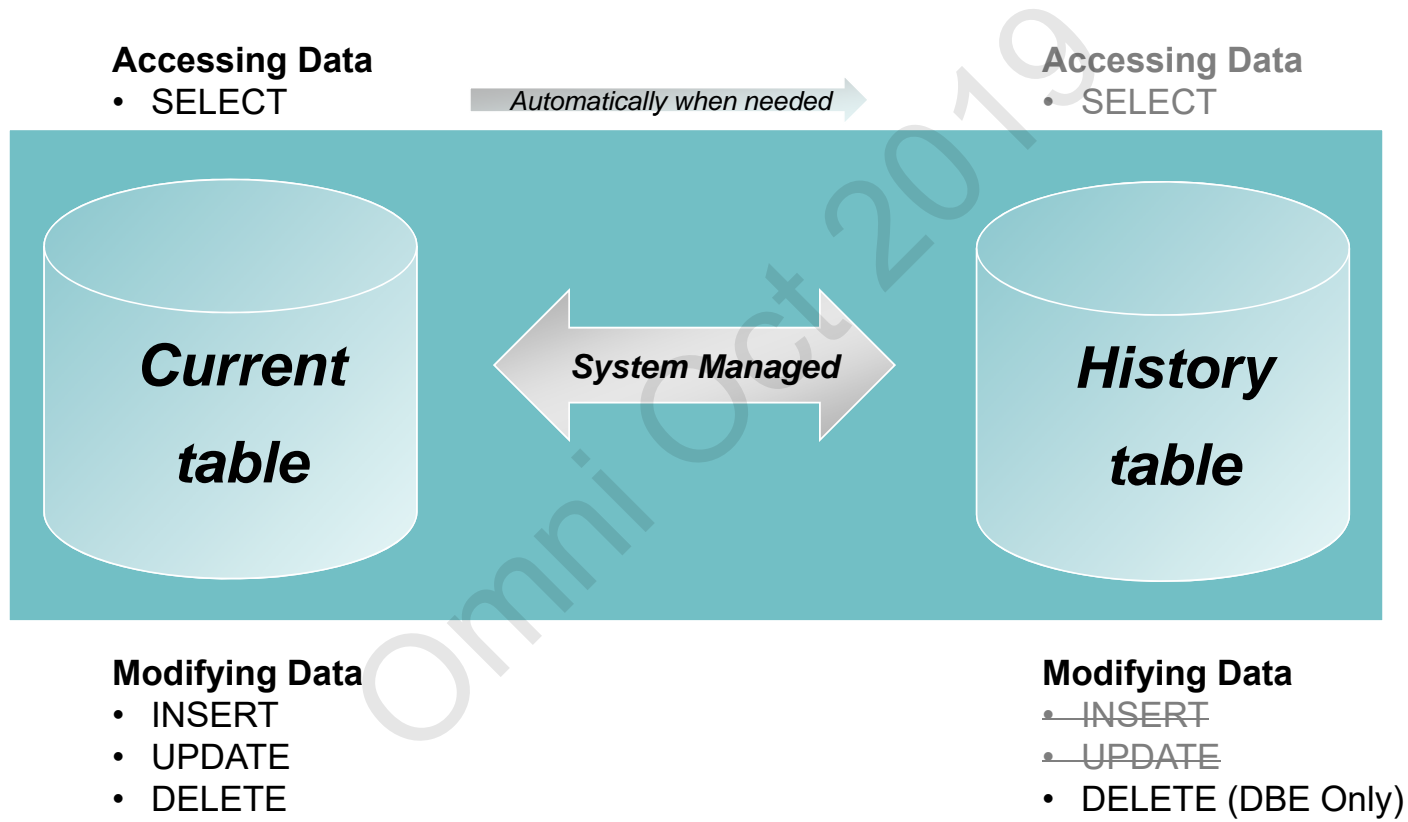
Ability to show when it was changed

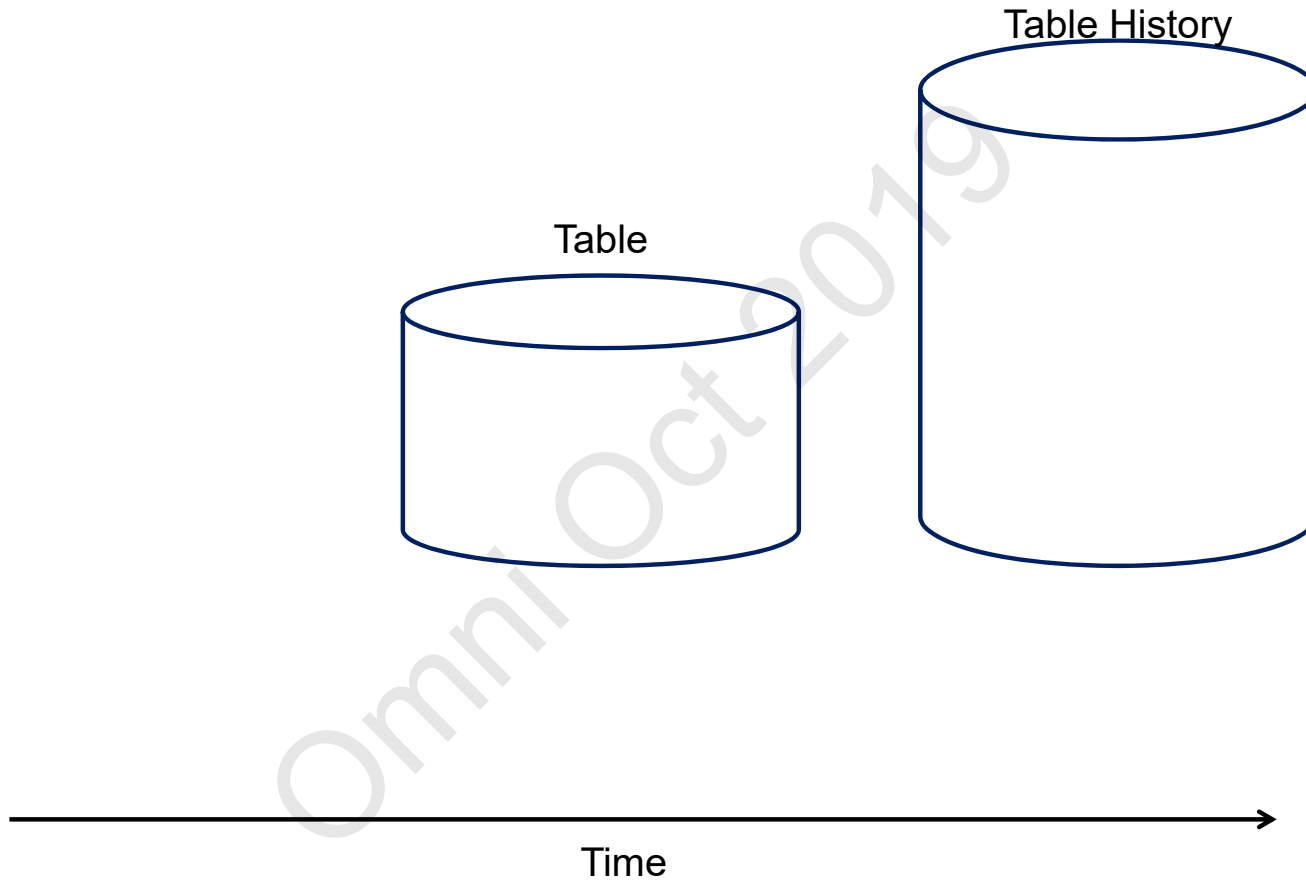
Option 1: journals and scraping

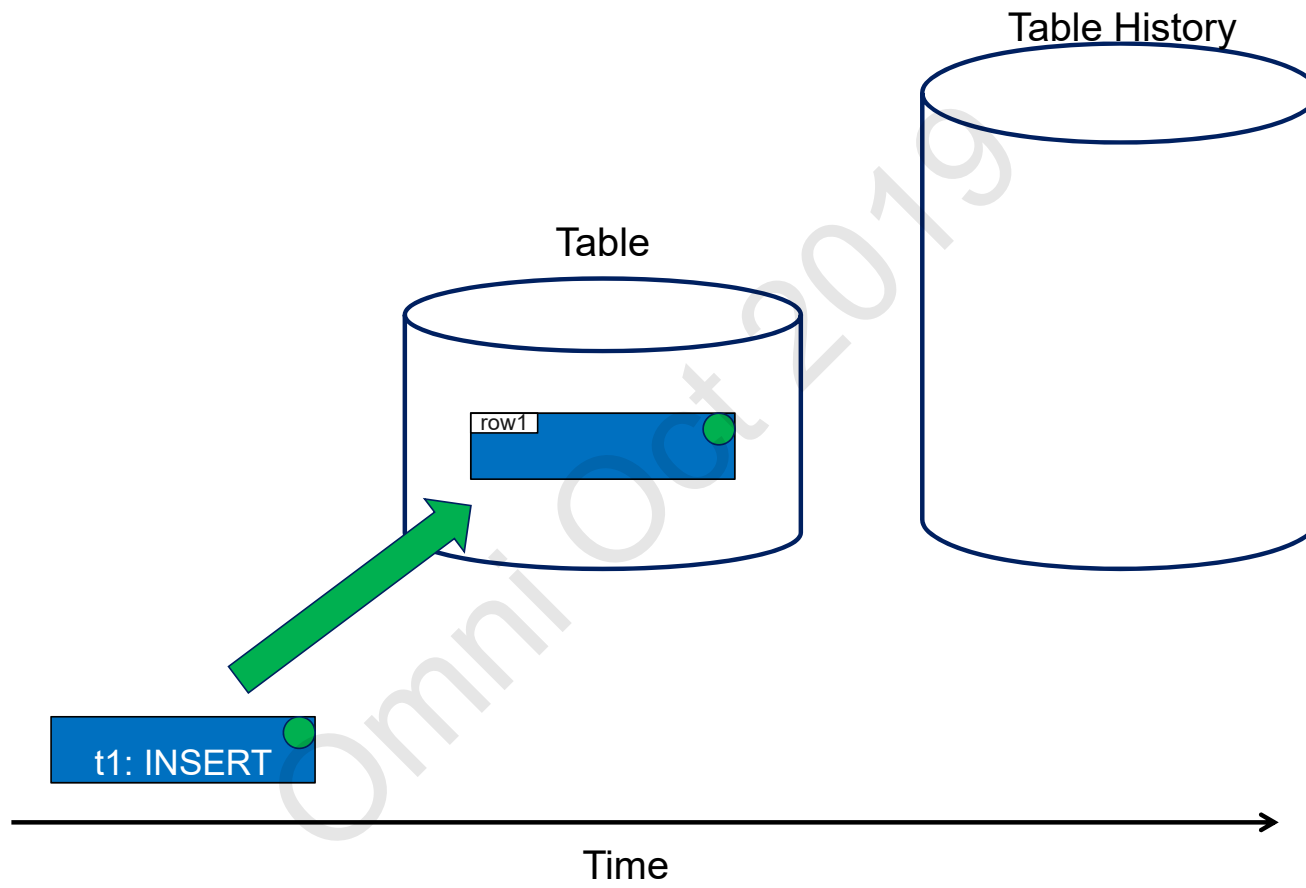


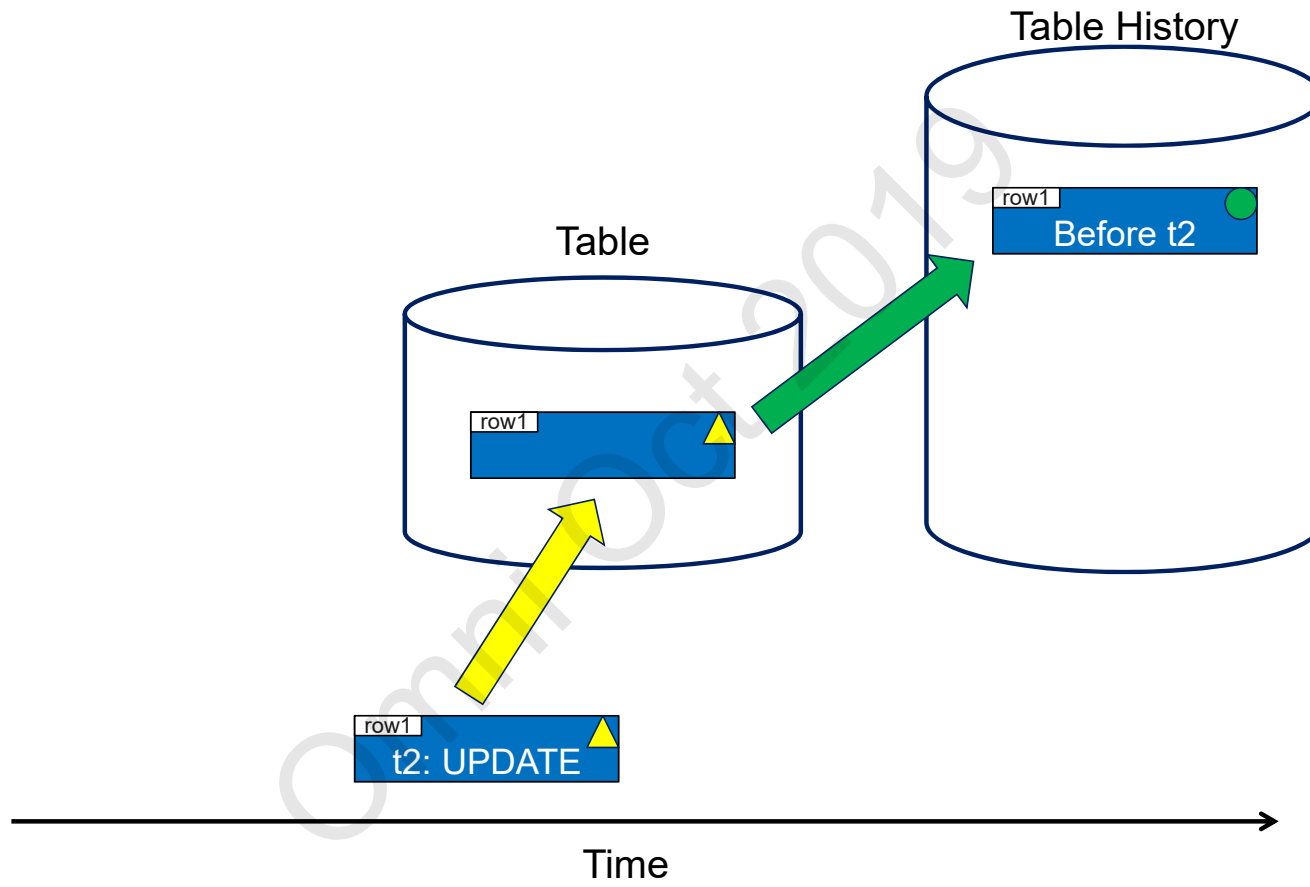
Option 2: history table

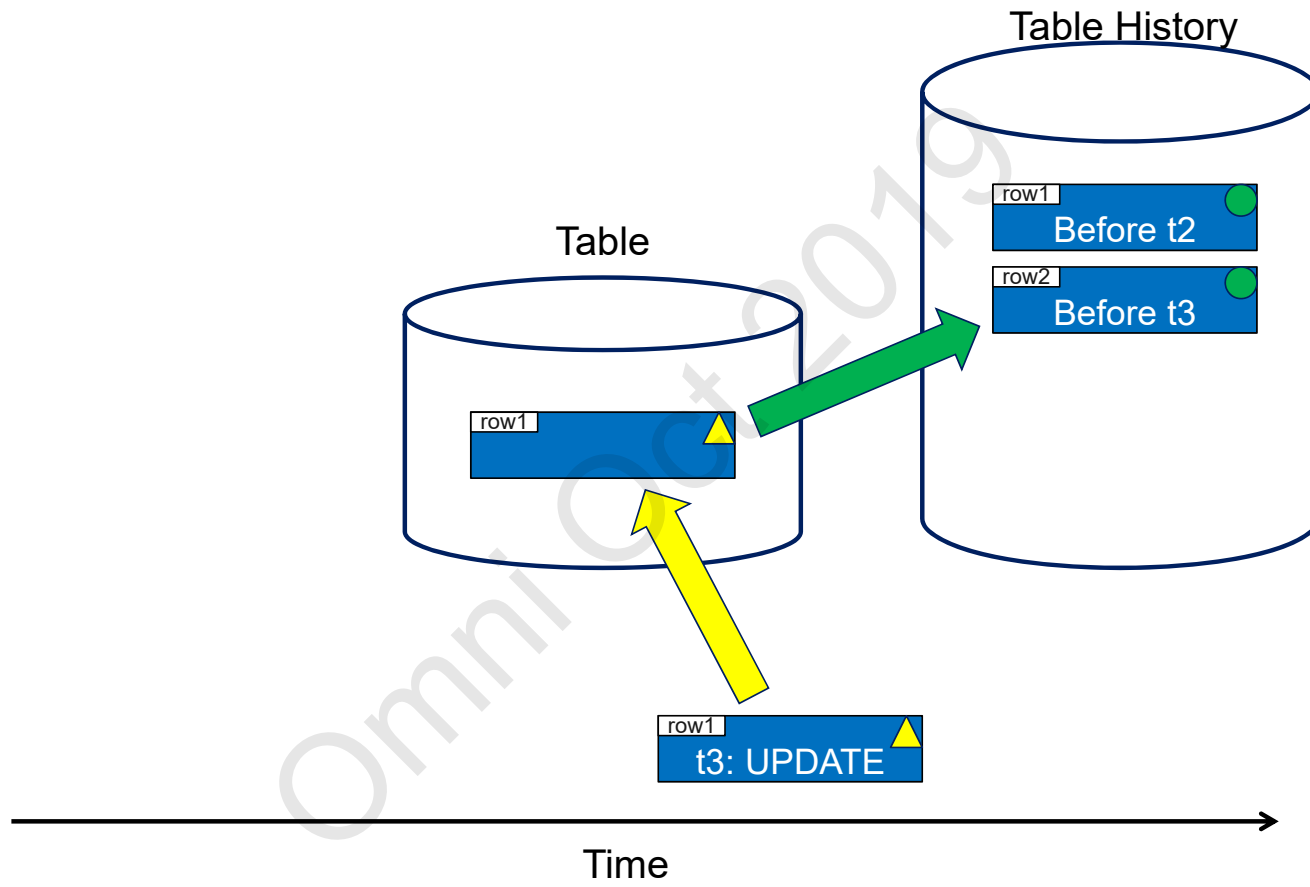


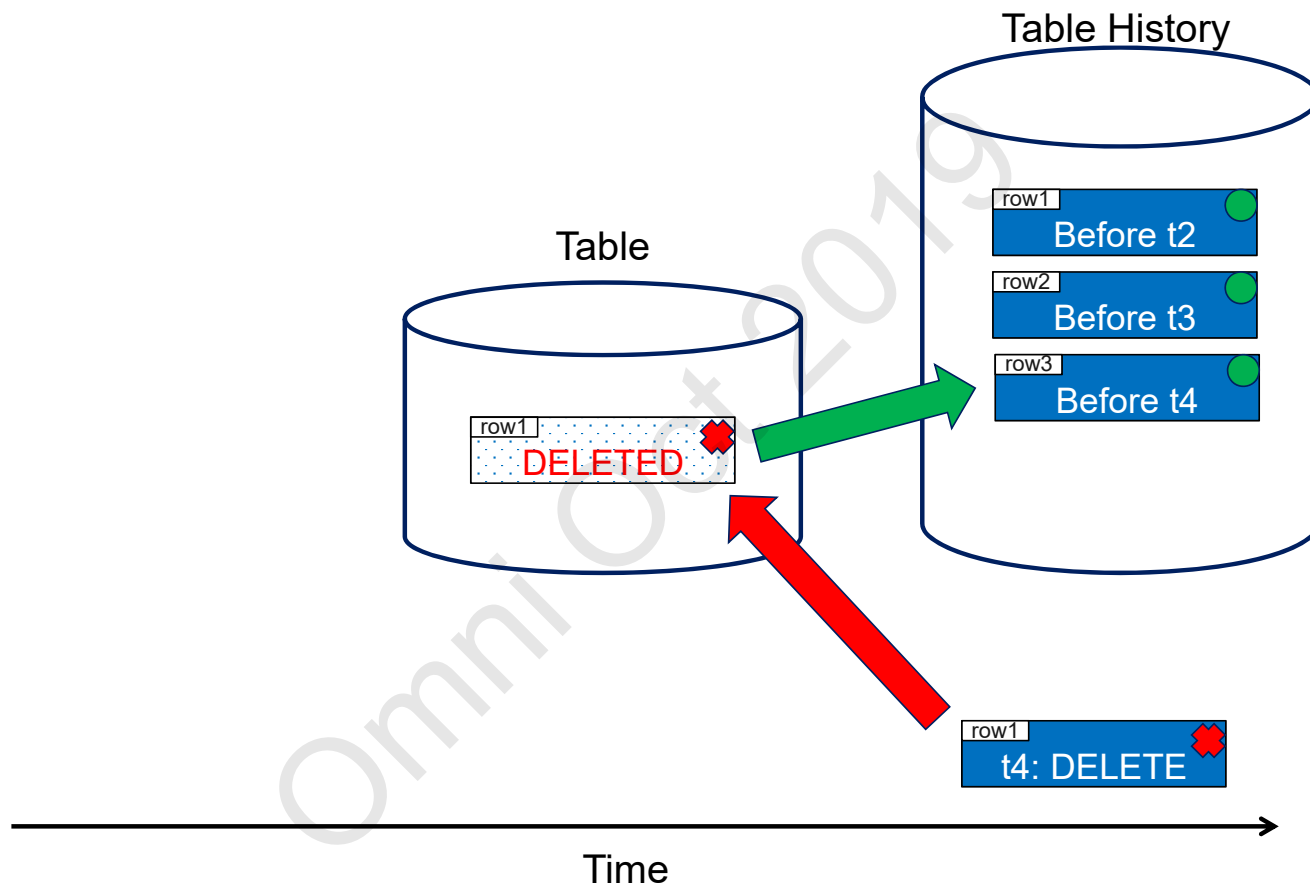








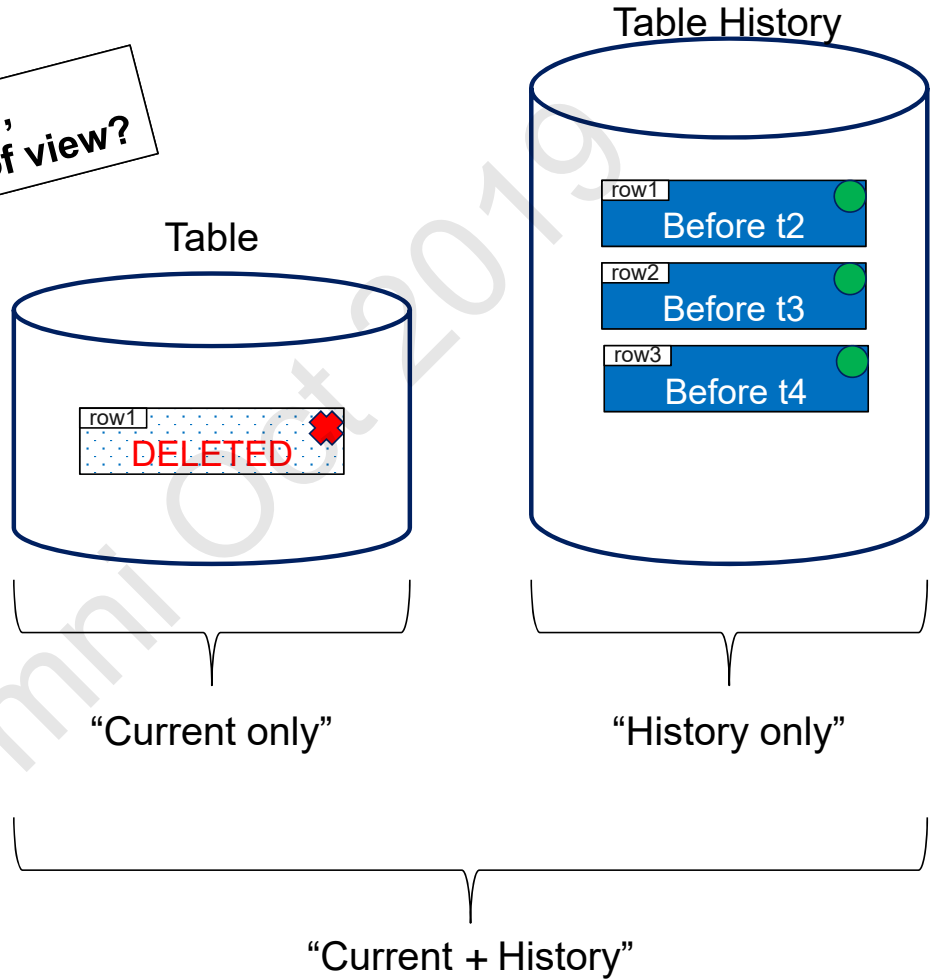




After 4 transactions,
what is the user's point of view?



All managed by Db2!



With Db2 Temporal Tables, you can ask:

- **Who was the client rep two years ago?**
SELECT CLIENT_REP FROM ACCOUNTS
FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP – 2 YEARS
- **Who were the client reps over the last five years?**
SELECT CLIENT_REP FROM ACCOUNTS
FOR SYSTEM_TIME FROM CURRENT_TIMESTAMP – 5 YEARS
TO CURRENT_TIMESTAMP
- **Run the inventory report using a different point in time**
SET **CURRENT TEMPORAL SYSTEM_TIME** '2018-12-26 17:00:00';
CALL GENERATE_INVENTORY_REPORT();



Db2 keeps the history of each row in a temporal table over time

What pieces are needed for tracking data (row) history?

- A repository for the rows' history – **history table**
- A unique identifier of rows modified under the same transaction – **transaction id**
- When was a row's historic value valid? – **row begin and row end** timestamps



```
ALTER TABLE employee  
  ADD COLUMN instance_begin  
    TIMESTAMP(12) NOT NULL  
    GENERATED ALWAYS AS ROW BEGIN  
  ADD COLUMN instance_end  
    TIMESTAMP(12) NOT NULL  
    GENERATED ALWAYS AS ROW END  
  ADD COLUMN transaction_id  
    TIMESTAMP(12)  
    GENERATED ALWAYS AS TRANSACTION START ID ADD PERIOD  
    SYSTEM_TIME (instance_begin, instance_end)
```

Establish birth/death of a row

```
CREATE TABLE employee_history LIKE employee
```

Create history table

```
ALTER TABLE employee ADD VERSIONING USE HISTORY TABLE  
employee_history
```

Enable Temporal tracking

```
ALTER TABLE emp_oyee
ADD COLUMN instance_begin
TIMESTAMP(12) NOT NULL IMPLICITLY HIDDEN
GENERATED ALWAYS AS ROW BEGIN
ADD COLUMN instance_end
TIMESTAMP(12) NOT NULL IMPLICITLY HIDDEN
GENERATED ALWAYS AS ROW END
ADD COLUMN transaction_id
TIMESTAMP(12) IMPLICITLY HIDDEN
GENERATED ALWAYS AS TRANSACTION START ID ADD PERIOD
SYSTEM_TIME (instance_begin, instance_end)
```

Establish birth/death of a row

```
CREATE TABLE emp_oyee_history LIKE emp_oyee
```

Create history table

```
ALTER TABLE emp_oyee ADD VERSIONING USE HISTORY TABLE
emp_oyee_history
```

Enable Temporal tracking

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>

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';
```

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

TABLE_NAME	PERIOD_NAME	BEGIN_COLUMN_NAME	END_COLUMN_NAME	HISTORY_TABLE_NAME
EMPLOYEES	SYSTEM_TIME	SYSTEM_START	SYSTEM_END	EMPLOYEES_HISTORY

QSYS2.SYSHISTORYTABLES

- The names of the associated history tables

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

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 t ... - Db2icoe2.rchland

TABLE_NAME	PERIOD_NAME	HISTORY_TABLE_NAME
EMPLOYEES	SYSTEM_TIME	EMPLOYEES_HISTORY

-
- Can there be more than one temporal table on my system?
 - Absolutely. Temporal is per table, so many tables can be made temporal (except history tables of course)
 - Can a DDS file (not just DDL/SQL) be made temporal?
 - Yes, though the three timestamp fields must exist in the file, which requires ALTER TABLE to add
 - Is journaling required? -- Yes, both for main and history table
 - Can I mix different FOR SYSTEM_TIME settings in different parts of a select e.g. two sides of a UNION can be at different SYSTEM_TIMEs? -- Yes
 - Are constraints supported?
 - Yes on the main table, but constraints are not allowed on the history table

-
- Can I attach an existing table as the history table?
 - Yes, but columns and attributes must exactly match. Simplest is to use CREATE TABLE LIKE
 - Can the history table have a superset of columns of the main table? -- No
 - Can column attributes be different (but compatible) between main table and history table?
 - No, names, attributes, and column order must exactly match
 - Can I ALTER TABLE (or CREATE OR REPLACE TABLE) a temporal (main) table to add a column?
 - Yes. The database automatically adds the same column to the history table
 - Can I ALTER TABLE of a temporal (main) table to drop a column?
 - No. History could be lost so database prevents it
 - You would need to manually handle this yourself (detach, drop column(s), reattach)

-
- Will RPG record level access (RLA – ‘native’) work against a temporal file? Yes and No
 - Yes – writes, updates and deletes will be managed into the history table by database automatically
 - Yes – reads/chains of the main table will work as normal (assuming you hide/manage the timestamp fields)
 - No – if you want to time travel, you have to manage accessing the history table yourself

 - Can I write directly into the history table?
 - No – you must detach it (so it is no longer a history table), then write, then attach again

 - Can I delete rows from the history table?
 - Yes, though you should restrict access to doing this (DBE level function)

 - Is an entire row copy done even if I only update a single column?
 - Yes

SQL access is preferred for **time-based access** since ease of access is built into the language

- When SQL statements reference the current table, Db2 for i automatically accesses the history table as needed
- New clauses on the SELECT statement
 - FOR SYSTEM_TIME AS OF <value>
 - FOR SYSTEM_TIME FROM <value> TO <value>
 - FOR SYSTEM_TIME BETWEEN <value> AND <value>
- New special register
 - CURRENT TEMPORAL SYSTEM_TIME

Note:

- Native time-based access requires manual work. Current and history tables are considered different accesses
- Database **does** still manage (and enforce) history tracking, even for native driven data changes

- Compare balances **between** different points in time for account 88880001

```
SELECT T1.BALANCE AS BALANCE_2017,  
       T2.BALANCE AS BALANCE_2018  
FROM account FOR SYSTEM_TIME AS OF '2017-12-31' T1,  
     account FOR SYSTEM_TIME AS OF '2018-12-31' T2  
WHERE T1.ACCT_ID = '88880001' AND T2.ACCT_ID = '88880001';
```

BALANCE_2017	BALANCE_2018
50000.00	60000.00

System Time Sensitivity is controlled at the program level:

- SYSTEM_TIME_SENSITIVE column within catalog QSYS2.SYSPROGRAMSTAT
 - NULL or 'NO' – Program is not time sensitive
 - 'YES' – Program is time sensitive
- Programs built prior to IBM i 7.3 are by default, **not time sensitive**
 - CURRENT TEMPORAL SYSTEM_TIME is ignored
- Programs (re)built on IBM i 7.3 are by default, **time sensitive**
 - CURRENT TEMPORAL SYSTEM_TIME is applied when queries reference temporal tables

Build time controls:

- Routines (SQL/External) → SET OPTION SYSTIME = *YES or *NO
- CRTSQLxxx → OPTION(*SYSTIME or *NOSYSTIME)
- RUNSQLSTM → SYSTIME(*YES or *NO)

Aggregation (GROUP BY)

- Be very careful aggregating across time ranges: **FOR SYSTEM_TIME BETWEEN** or **FOR SYSTEM TIME FROM**. A 'row' could be counted several times!

Ex:

```
SELECT COUNT(*) FROM ORDER_HEADER  
FOR SYSTEM_TIME BETWEEN '0001-01-01' AND '9999-12-30' ;
```

Does NOT count the total number of orders since the beginning!



Joins (joining tables)

- From a business perspective, do all tables need to be temporal?
 - Is a subset of tables enabled as temporal enough?
- Could the join column(s) change when a row is updated?
 - If so, think through update situations to ensure answers are consistent

Omni Oct 2019

- **Autogenerated columns are a very powerful building block for datacentric programming in that they direct the database to automatically generate column values**
 - Database manages them. Row values cannot be altered, even by a developer
- **Prior to IBM i 7.3, Db2 for i supported:**
 - IDENTITY columns (which are very good for surrogate primary keys)
 - ROW CHANGE TIMESTAMP (which records the time whenever a row is changed)
- **The SQL syntax GENERATED ALWAYS prevents anyone from modifying those column values, including a knowledgeable hacker**
- **IBM i 7.3 includes more autogenerated options:**
 - DATA CHANGE OPERATION (I/U/D)
 - Special register
 - Built-in Global Variable

- **DATA CHANGE OPERATION** is a one character value recording the last data change:
 - I = Insert
 - U = Update
 - D = Delete
- These work well with temporal tables in that history table will provide a timeline of what changes were made and when
 - The Delete ('D') record will be included if the temporal table was configured with the **ON DELETE ADD EXTRA ROW** clause

```
ALTER TABLE fact_table
  ADD COLUMN audit_type_change CHAR (1)
  GENERATED ALWAYS AS (DATA CHANGE OPERATION)
```

Db2 provides different ways to communicate across an application flow. Two in particular are of interest:

1. Special Registers

- Predefined special values that can be referenced in SQL
Examples: CURRENT USER, CURRENT TIMESTAMP, CURRENT DATE...
- Most registers are maintained by the database. However, some registers can be SET by the application
Examples: CLIENT_ACCTNG, CLIENT_USERID...

2. Global Variables

- Variables that can be created and used across SQL statements

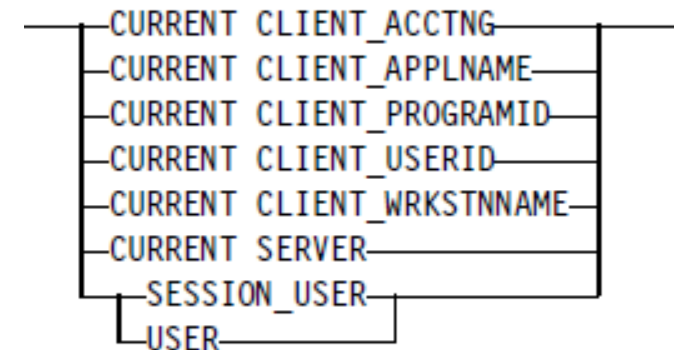
Example:

```
CREATE VARIABLE QGPL.MYVAR INT DEFAULT 123
...
SELECT * FROM MYTAB WHERE MYCOL = QGPL.MYVAR
```

- Database defines and manages some built-in global variables
Examples: QSYS2.JOB_NAME, SYSIBM.CLIENT_IPADDR



- **Special registers can be used to record information about the user making the change and/or the application environment**
- **Client registers can be set by the application to provide additional application information**
- **CURRENT SERVER contains the currently connected server**
- **SESSION_USER and USER contain the user profile currently in use which identifies who is making a change to the database**



```
ALTER TABLE fact_table
  ADD COLUMN audit_app_client_userid VARCHAR(255)
    GENERATED ALWAYS AS (CURRENT CLIENT_USERID)
  ADD COLUMN audit_user VARCHAR(128)
    GENERATED ALWAYS AS (SESSION_USER)
```

- **Built-in global variables are managed by the system and provide additional environmental information**
- **You can use these to monitor things like which job or which IP address is being used to make a change to the database**

```
ALTER TABLE fact_table
  ADD COLUMN audit_job_name VARCHAR(28)
    GENERATED ALWAYS AS (QSYS2.JOB_NAME)
  ADD COLUMN audit_client_IP VARCHAR(128)
    GENERATED ALWAYS AS (SYSIBM.CLIENT_IPADDR)
```

QSYS2.JOB_NAME
QSYS2.SERVER_MODE_JOB_NAME
SYSIBM.CLIENT_HOST
SYSIBM.CLIENT_IPADDR
SYSIBM.CLIENT_PORT
SYSIBM.PACKAGE_NAME
SYSIBM.PACKAGE_SCHEMA
SYSIBM.PACKAGE_VERSION
SYSIBM.ROUTINE_SCHEMA
SYSIBM.ROUTINE_SPECIFIC_NAME
SYSIBM.ROUTINE_TYPE

Omni Oct 2019

- Saving a temporal table is like saving any other file except:
 - **You** must coordinate the save (and restore) of the history table with it
- Temporal table and history table CAN be saved separately
- If a temporal table is restored without the history table or history table is otherwise missing, the table becomes read only until versioning is reestablished or removed
- If the history table is restored first, when the temporal table is restored versioning is automatically reestablished
- Simplest is to do an entire library save (and restore), assuming history table is in the same library

Temporal tables need to be evaluated if you plan to use them in a Replication/HA environment

- Hardware replication is usually fine since it copies at the byte level
- Software (journal) replication does not just 'fall out'
- Replication solutions must take into account the read only aspect of the history table and how database shadows changes into the history table
- Database has supported interfaces that replication vendors can leverage
 - Support varies by vendor

Ask your replication vendor!

Omni Oct 2019

- **Db2 for IBM i homepage:** www.ibm.com/systems/power/software/i/db2

IT infrastructure > Power Systems > Software > IBM i >

IBM DB2 for i


Overview Benefits Getting started Products Resources

DB2 for i (formerly known as DB2 for i5/OS) is an advanced, 64-bit Relational Database Management System (RDBMS) that leverages the high performance, virtualization, and

- **Db2 for IBM i wiki:**
ibm.biz/Bd4fFb

You are in: [DB2 for i Wiki](#) > Welcome to DB2 for i

Welcome to DB2 for i

 | Updated yesterday at 7:18 PM by [drmack2](#) | Tags: *None*

Page Actions ▾

Welcome to the home page for the DB2 for i Wiki. Here you will find a variety of information from the leading experts for DB2 for i within IBM.

Thank You!

www.ibm.com/developerworks/ibmi/techupdates/db2