Time travel with DB2 for i - Temporal tables on IBM i 7.3

Scott Forstie
DB2 for i Business Architect
forstie@us.ibm.com
@Forstie_IBMi

For…

DB2 for i

- Standard compliant
- Secure
- Scalable
- Functionally Advanced
- Excellent Performance
- Easier to use
- Easier to maintain

Value Proposition

Continual Investment and Innovation

V5R2
- SQE Stage 1
- IASPs
- Identity columns
- Savepoints
- UNION in views
- Scalar subselect
- UDTFs
- DECLARE GLOBAL TEMPORARY TABLE
- Catalog views

V5R3
- Partitioned tables
- UFT-8 & UTF-16
- ICU sort sequence
- MQTs
- Sequences
- Implicit char/numeric
- BINARY / VARBINARY
- GET DIAGNOSTICS
- DRDA Alias
- DECIMAL(63)
- SQE Stage 3
- Ragged SWA
- QDBRPLAY
- Online Reorganize

V5R4
- WebQuery
- SSD Memory Preference
- On Demand Performance Center
- Health Center
- Completion of SQL Core
- Scalar fullselect
- Recursive CTE
- INSTEAD OF triggers
- Descriptor area
- XA over DRDA
- DDM 2-phase
- Scrollable cursor
- 2M SQL statement
- 1000 tables in a query

6.1
- Omnifind
- MySQL storage engine
- DECFLOAT
- Grouping sets / super groups
- INSERT in FROM
- VALUES in FROM
- Extended Indicator Variables
- Expression in Indexes
- ROW CHANGE TIMESTAMP
- Statistics catalog views
- CLIENT special registers
- SQE Stage 6
- DDM and DRDA IPv6
- Deferred Restore of MQT and Logicals
- Environmental limits

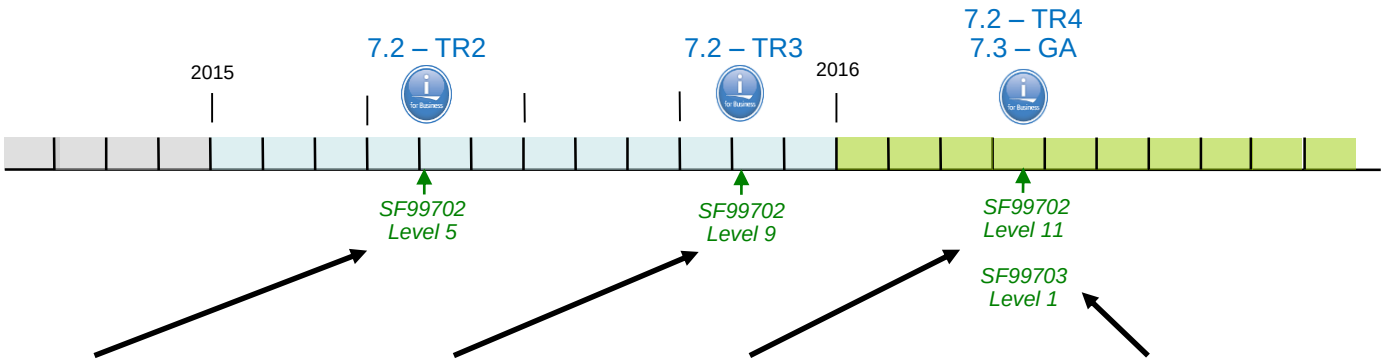
7.1
- XML Support
- Encryption enhancements (FIELDPROCs)
- Result set support in embedded SQL
- CURRENTLY COMMITTED
- MERGE
- Global variables
- Array support in procedures
- Three-part names and aliases
- SQE Logical file support
- SQE Adaptive Query Processing
- EVI enhancements
- Inline functions
- CREATE OR REPLACE
- TR-timed enhancements
- DECLARE GLOBAL TEMPORARY TABLE
- Catalog views

7.2
- Row and Column Access Control
- XMLTABLE
- CONNECT BY
- OLAP Extensions
- TRANSFER OWNERSHIP
- Named arguments and defaults for parameters
- Obfuscation of SQL routines
- Array support in UDFs
- Timestamp precision
- Multiple-action Triggers
- Built-in Global Variables
- 1.7 Terabyte Indexes
- Navigator Graphing and Charting

7.3
- Temporal Tables
- Generated columns for auditing
- OLAP Extensions
- Regression Functions / Covariance
- EVI Only Access
- More Built-in Global Variables
- More SQL Scalar functions
- More IBM i Services
- CREATE OR REPLACE TABLE
- ATTACH & DETACH Partition
- System Limits for IFS

TR	SQL Enhancement
TR4	**Fair Lock vs No Lock**
TR3	**LIMIT & OFFSET**
TR2	**CREATE OR REPLACE TABLE & DB2 JSON Store**
TR1	**Regular Expressions**

DB2 for i – Enhancements delivered via DB2 PTF Groups



TR2-timed Enhancements

- Create OR REPLACE table
- JSON – DB2 Store Technology Preview
- SQE Performance improvements
- And more…

TR3-timed Enhancements

- LIMIT and OFFSET
- Guardium V10 and other database security monitoring enhancements
- SQE Performance improvements
- More IBM i Services
- New SQL built-in functions
- Enhancements for SAP on i clients

TR4-timed Enhancements

- Inlined UDTFs
- Trigger (re)deployment
- More IBM i Services
- New DB2 built-in Global Variables
- Enhanced SQL Scalar functions
- Guardium on i enhancement
- Evaluation option for DB2 SMP & DB2 Multisystem

Enhancements in 7.3:

- Temporal Tables
- Generated columns for auditing
- New OLAP built-ins
- Raised architecture limits
- New support for partitioned tables
- More IBM i Services
- All TR-timed enhancements

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

Reasons to Upgrade – Database

Why move to IBM i 7.2?

- Database performance
  - ✓ SQE handles Native DB access
  - ✓ New I/O Costing Model
  - ✓ EVI Only Access
- Data-centric security
  - ✓ Row & Column Access Control for SQL and DDS files
- Developer productivity
  - ✓ Default parameters on functions
  - ✓ Built-in Global Variables
  - ✓ Many other improvements
- Workload insight
  - ✓ Improved SQL Plan Cache
  - ✓ Performance Data Perspectives

Why move to IBM i 7.3?

- Data-centric history
  - ✓ System-period Temporal table support for SQL tables and DDS created physical files
- Data-centric accountability
  - ✓ Generated columns for SQL and DDS files
  - ✓ Authority Collection to avoid excess authority
- On-Line Analytical Processing (OLAP)
  - ✓ New OLAP built-in functions
  - ✓ Improved capabilities for DB2 Web Query, Cognos Analytics and other BI tools
- Improved value from priced options
  - ✓ DB2 SMP – Parallel execution of OLAP
  - ✓ DB2 Multisystem – Attach/Detach partitions
- Plus 7.3 TR-timed enhancements

Knowledge Center and IBM i 7.3

Read about it… (live links in the pdf)

- SQL Reference - What's New
- SQE Optimizer - What's New
- Temporal Tables - Administration
- Temporal Tables - Programming
- Generated Columns for Auditing
- On-Line Analytical Processing (OLAP) specifications
- OLAP specifications - Examples
- IBM i Navigator - database enhancements

DB2 for i – Tech Tip Series

Follow my adventures in a new Tech Tip Series where I explain DB2 for i on IBM i 7.3.

“TechTip: i Illuminate 7.3 – Series”

Accompany an apprentice wizard on this tour of IBM i 7.3 and avoid being whomped by a willow or suffer from petrification.

i illuminate 7.3

http://www.mcpressonline.com/ibm-i-os/400-i5/os/techtip-i-illuminate-73-%E2%80%93-series-premier.html
http://www.mcpressonline.com/database/techtip-i-illuminate-73-%E2%80%93-time-turner.html
http://www.mcpressonline.com/database/techtip-i-illuminate-73%E2%80%94get-a-grip.html

Temporal Tables
&
Generated Columns

http://www.ibm.com/developerworks/ibmi/techupdates/i73

DB2 for i – Business questions

With Temporal Table & Generated columns, you can:

- **Show me the client reps from two years ago?**
- **Produce an inventory report using a different point in time**
- **Who deleted that row?**
- **Who last updated this row?**

DB2 for i – SQL answers

With Temporal Table & Generated columns, you can:

- **Show me the client reps from two years ago?**
  SELECT CLIENT_REP FROM ACCOUNTS
  FOR SYSTEM_TIME AS OF CURRENT TIMESTAMP – 2 YEARS

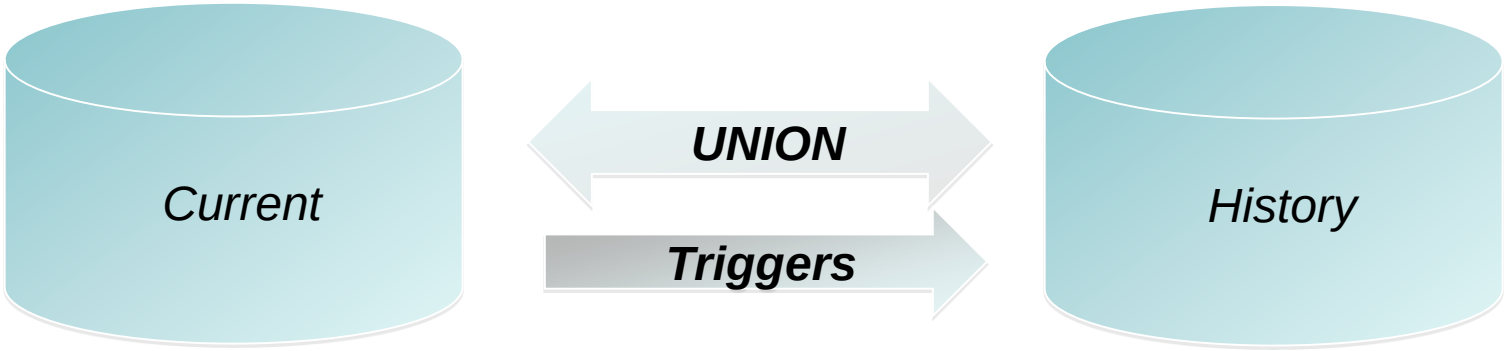
- **Produce an inventory report using a different point in time**
  SET CURRENT TEMPORAL SYSTEM_TIME '2016-03-22 17:00:00';
  CALL GENERATE_INVENTORY_REPORT();

- **Who deleted that row?**
  SELECT AUDIT_USER, AUDIT_JOB FROM SALES
  FOR SYSTEM_TIME FROM CURRENT DATE – 1 MONTH TO CURRENT DATE WHERE AUDIT_OP = 'D'

- **Who last updated this row?**
  SELECT AUDIT_USER, AUDIT_CLIENT_IP FROM ITEM_FACT
  WHERE ITEM_KEY = '125A16'

History – Do It Yourself

Accessing Data
- SELECT



Accessing Data
- SELECT

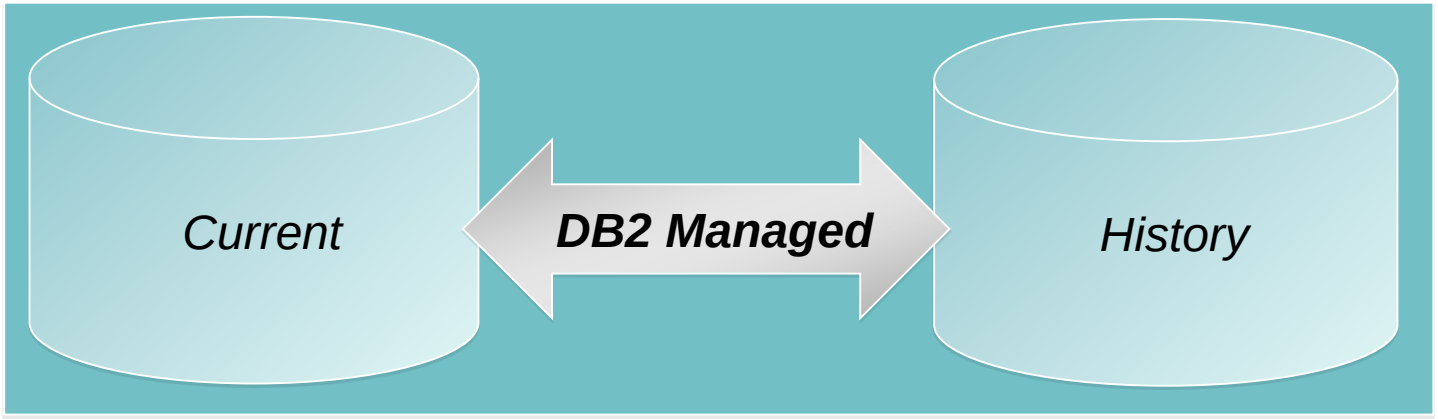
Modifying Data
- INSERT
- UPDATE
- DELETE

Modifying Data
- INSERT
- UPDATE
- DELETE

History – DB2 for i Managed

Accessing Data
- SELECT

~~Accessing Data~~
- ~~SELECT~~



Modifying Data
- INSERT
- UPDATE
- DELETE

Modifying Data
- ~~INSERT~~
- ~~UPDATE~~
- DELETE (DBE Only)

Temporal construction for data-centric history

```
ALTER TABLE account
 ADD COLUMN row_birth
   TIMESTAMP(12) NOT NULL
   GENERATED ALWAYS AS ROW BEGIN
 ADD COLUMN row_death
   TIMESTAMP(12) NOT NULL
   GENERATED ALWAYS AS ROW END
 ADD COLUMN transaction_time
   TIMESTAMP(12)
   GENERATED ALWAYS AS TRANSACTION START ID
 ADD PERIOD SYSTEM_TIME (row_birth, row_death)
```

Establish birth/death of a row

```
CREATE TABLE account_hist LIKE account
```

Create history table

```
ALTER TABLE account
   ADD VERSIONING USE HISTORY TABLE account_hist
```

Enable Temporal tracking

Temporal construction for data-centric history

```
ALTER TABLE account
 ADD COLUMN row_birth
   TIMESTAMP(12) NOT NULL IMPLICITLY HIDDEN
   GENERATED ALWAYS AS ROW BEGIN
 ADD COLUMN row_death
   TIMESTAMP(12) NOT NULL IMPLICITLY HIDDEN
   GENERATED ALWAYS AS ROW END
 ADD COLUMN transaction_time
   TIMESTAMP(12)           IMPLICITLY HIDDEN
   GENERATED ALWAYS AS TRANSACTION START ID
 ADD PERIOD SYSTEM_TIME (row_birth, row_death)
```

Establish birth/death of a row

```
CREATE TABLE account_hist LIKE account
```

Create history table

```
ALTER TABLE account
   ADD VERSIONING USE HISTORY TABLE account_hist
```

Enable Temporal tracking

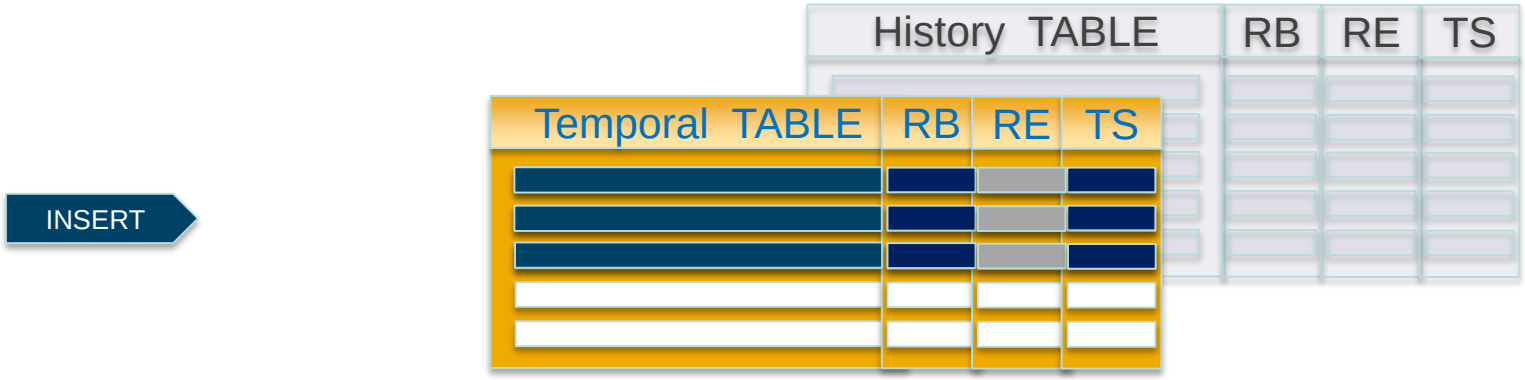
Accessing a Temporal Table

- **SQL statements reference the current table, DB2 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

Temporal in motion

Inserting rows does not impact the history table

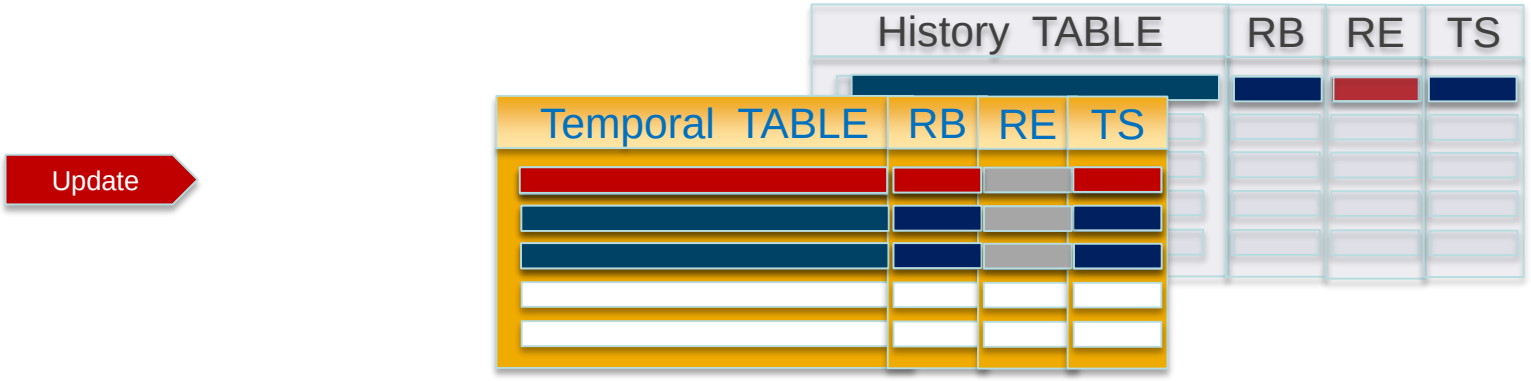
- ROW BEGIN (RB) Column – timestamp when the row was born
- ROW END (RE) Column – set to “end of time”



Temporal in motion

Updating rows causes rows to be added to the history table

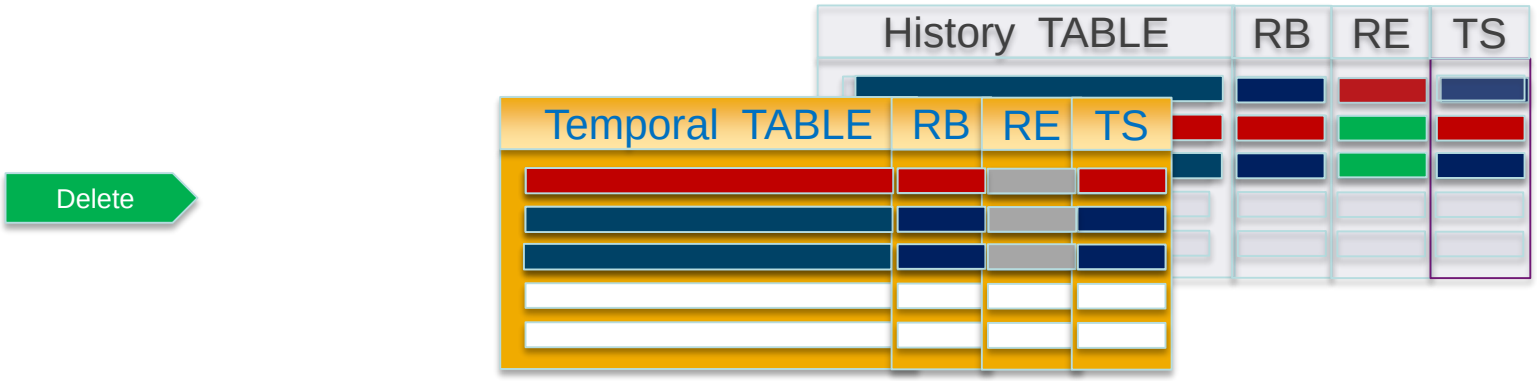
- ROW BEGIN (RB) Column – timestamp when the row was born
- ROW END (RE) Column – the death of the row results in the RE of the historical row matching the RB of the active row



Temporal in motion

Deleting rows removes them from the temporal table and adds them to history table

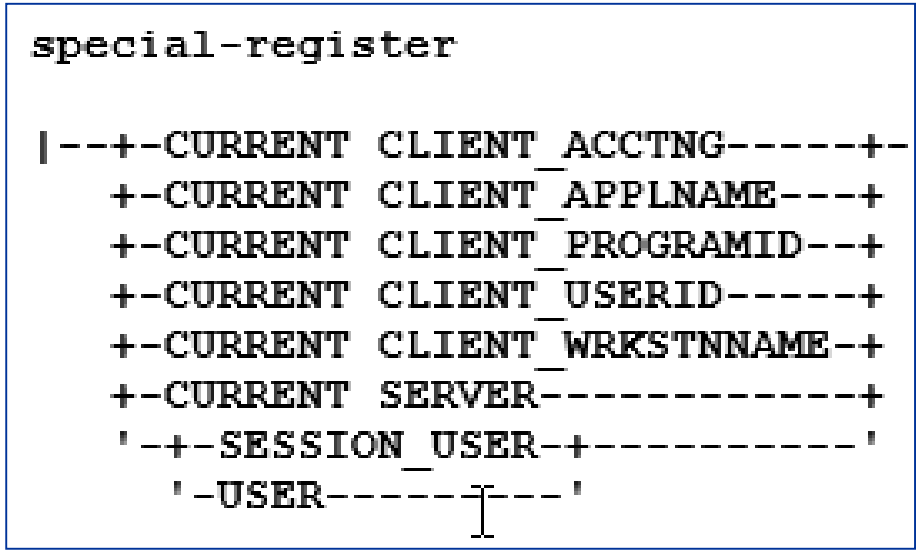
- ROW END (RE) Column – set to the death time of the row

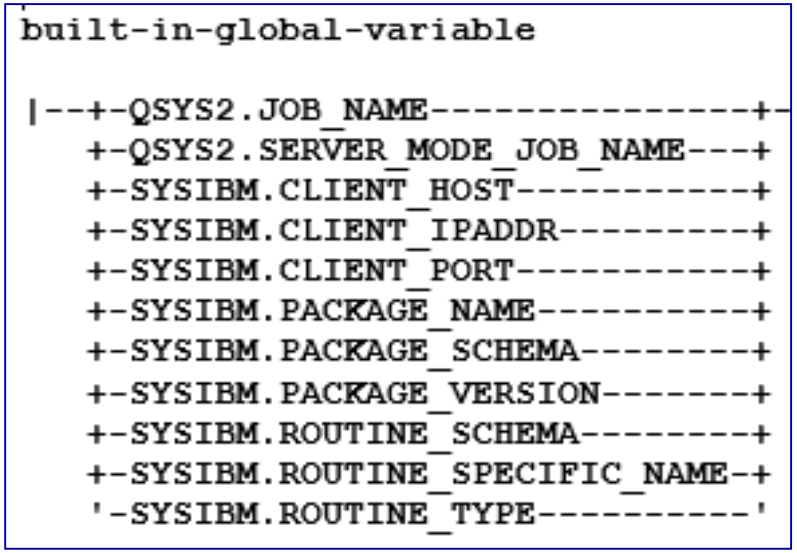


DB2 for i
&
Row Level Auditing

Row level auditing with Generated Columns

- **What you have on previous releases:**
  - When was this row last updated? (*row-change-timestamp-clause*)
- **New Generated expressions in IBM i 7.3:**
  - DATA CHANGE OPERATION (I/U/D)
  - Special register
  - Built-in Global Variable





Row level auditing with Generated Columns

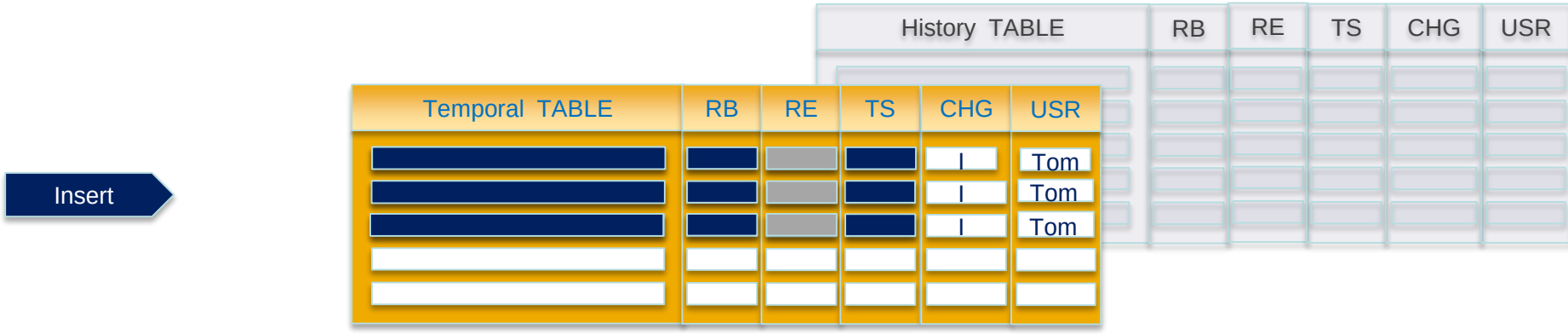
- **Establish generated columns into existing files**
- **Works for SQL Tables & DDS Created Physicals**
- **No need to change applications**

```
ALTER TABLE  account
ADD COLUMN audit_type_change CHAR (1)
  GENERATED ALWAYS AS (DATA CHANGE OPERATION)
ADD COLUMN audit_user VARCHAR(128)
  GENERATED ALWAYS AS (SESSION_USER)
ADD COLUMN audit_client_IP VARCHAR(128)
  GENERATED ALWAYS AS (SYSIBM.CLIENT_IPADDR)
ADD COLUMN audit_job_name VARCHAR(28)
  GENERATED ALWAYS AS (QSYS2.JOB_NAME)
```

Data Change Operation and Row-level Auditing detail

History table stores previous versions of a system-period temporal table's rows

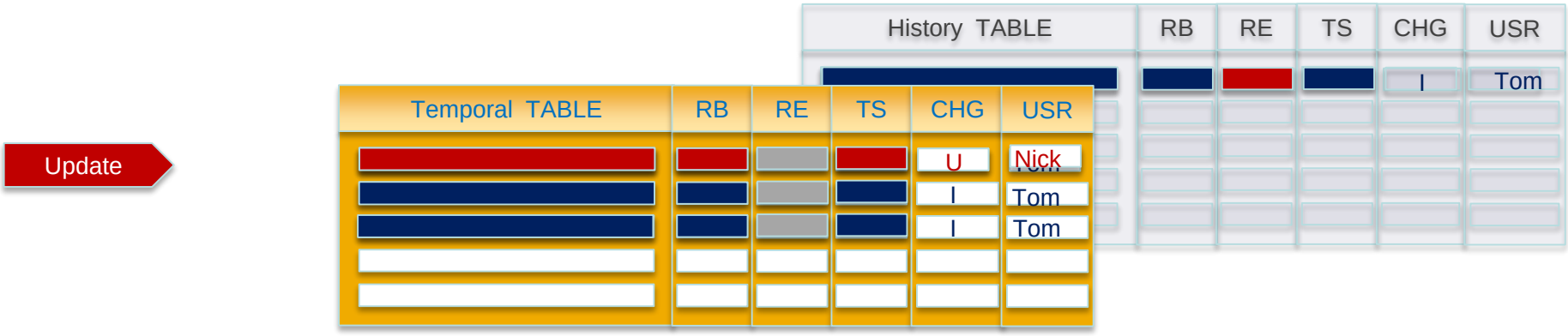
- ROW BEGIN (RB) Column – timestamp when the rows were born
- ROW END (RE) Column – set to "end of time"
- Data Change Operation (CHG) – 'I' for INSERT
- Session User (USR) – identity of inserter



Data Change Operation and Row-level Auditing detail

History table stores previous versions of a system-period temporal table's rows

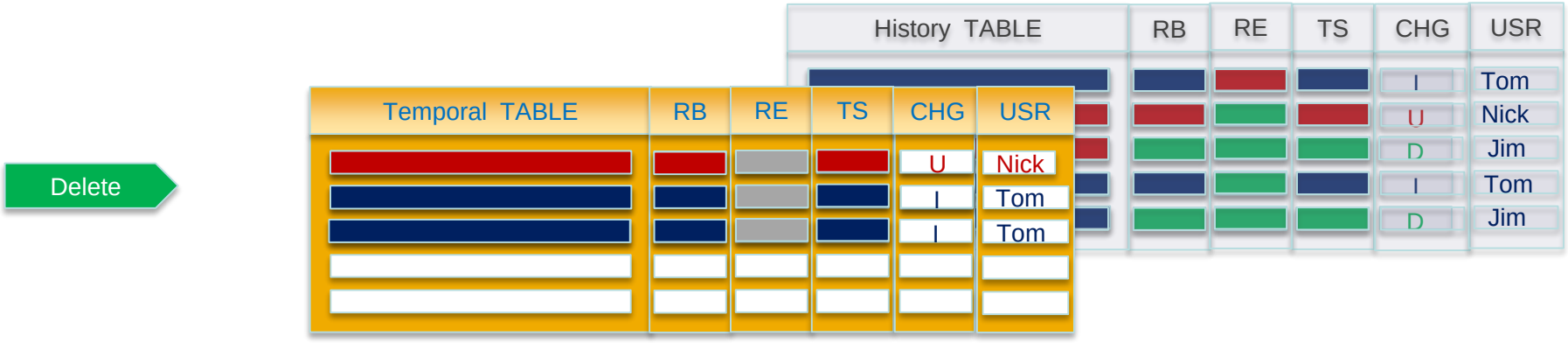
- ROW BEGIN (RB) Column – Birth
- ROW END (RE) Column – Death
- Data Change Operation (CHG) – 'U' for UPDATE
- Session User (USR) – identity of updater



ON DELETE ADD EXTRA ROW – in motion

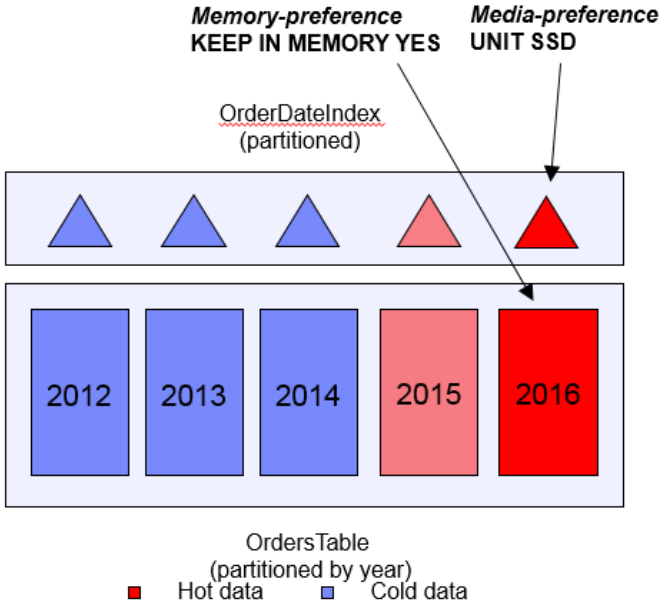
History table stores previous versions of a system-period temporal table's rows

- ROW BEGIN (RB) Column – Birth
- ROW END (RE) Column – Death
- Data Change Operation (CHG) – 'D' for DELETE
- Session User (USR) – identity of deleter



DB2 Multisystem (feature of IBM i)

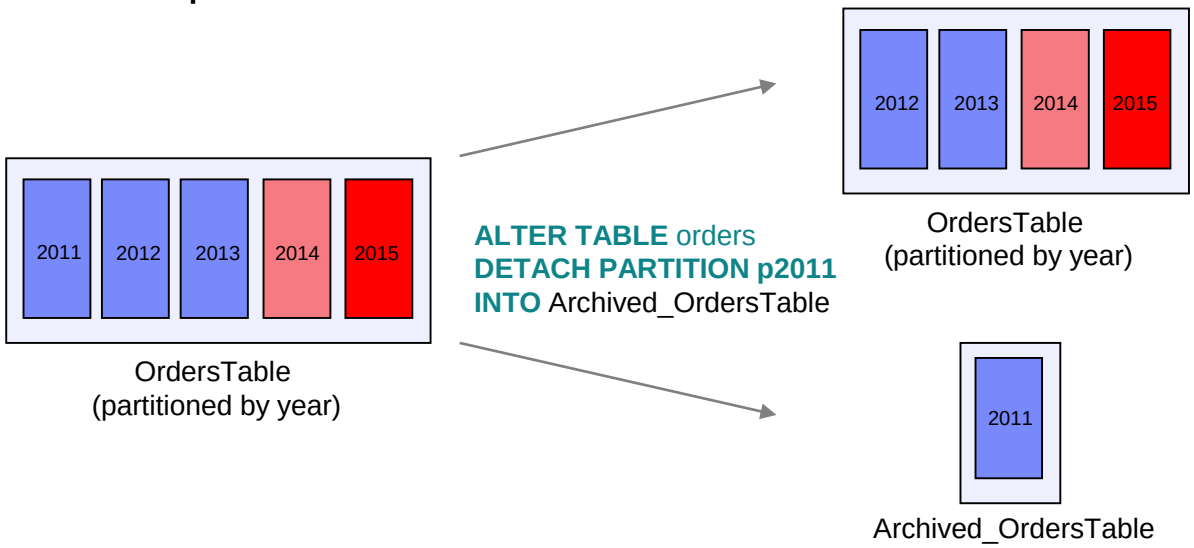
- **Provides ability to partition tables**
  - Non-partitioned tables are limited to 4.2B rows or 1.7TB
  - Partitioning multiplies these limits by up to 256 times
    - Limits of over one trillion rows and 435TB
  - Management benefits
    - Efficient removal of old data
    - Faster save times
    - Ability to detach partitions in IBMi 7.3
    - Improved query performance
- **Planning is critical**
  - White Paper:
    Table Partitioning Strategies for DB2 for i
    https://ibm.biz/PartitionedTablesIBMi
  - DB2 for i VLDB Consulting Workshop
    https://ibm.biz/DB2CoEworkshops



ALTER TABLE ATTACH and DETACH Partitions

ALTER TABLE DETACH PARTITION allows for the efficient roll-out of a partition that is no longer needed to be kept online.

- **ALTER TABLE DROP PARTITION –** Delete the data
- **ALTER TABLE DETACH PARTITION –** Retain the data, creating a new single partition, partitioned table



Temporal history – rows organized by time

- Temporal table history tables contain rows that are natural to organize by time.
- The history table can be partitioned, even if the system-time temporal table is not partitioned
- Why consider using local partitioning for your history table?
  1. Improved query execution
  2. Reduced index maintenance
  3. Faster save times
  4. Ease of use when data is has aged beyond relevance

```
CREATE TABLE account_history LIKE account
PARTITION BY RANGE ( row_death)
(PARTITION  p2016 STARTING ('01/01/2016') INCLUSIVE ENDING ('01/01/2017') EXCLUSIVE,
 PARTITION  p2017 STARTING ('01/01/2017')  INCLUSIVE ENDING ('01/01/2018') EXCLUSIVE,
 PARTITION  p2018 STARTING ('01/01/2018') INCLUSIVE ENDING ('01/01/2019') EXCLUSIVE,
 PARTITION  p2019 STARTING ('01/01/2019') INCLUSIVE ENDING ('01/01/2020') EXCLUSIVE );
```

Partitioned History table

DB2 for i priced OS options – evaluation copy

Try before you buy! On any IBM i 7.x release!

DB2 Symmetric Multiprocessing – Option 26
DB2 Multisystem – Option 27

The IBM Lab Services DB2 for IBM i team has the ability to allow you to evaluate either of these options for up to 70 days, for no charge.

This is a simpler, no strings attached, way to evaluate these valuable database options.

Interested?

Contact…
Rob Bestgen (bestgen@us.ibm.com) or
Scott Forstie (forstie@us.ibm.com)

Temporal – history behind the scenes

SELECT * FROM account WHERE ACCT_ID = '88880001';

ACCT_ID	BALANCE	TRANSACTION_TIME	INSTANCE_BEGIN	INSTANCE_END	TRANSACTION_ID
88880001	60000.00	2014-12-20 10:05:18.617454000000	2014-12-20 10:05:18.617454000000	9999-12-30 00:00:00.000000000000	2014-12-20 10:05:18.617454000000

SELECT * FROM account_hist WHERE ACCT_ID = '88880001';

ACCT_ID	BALANCE	TRANSACTION_TIME	INSTANCE_BEGIN	INSTANCE_END	TRANSACTION_ID
88880001	3000.00	2013-01-02 10:02:16.987139000000	2013-01-02 10:02:16.987139000000	2013-05-05 14:36:16.637149000000	2013-01-02 10:02:16.987139000000
88880001	10.00	2013-05-05 14:36:16.637149000000	2013-05-05 14:36:16.637149000000	2013-12-30 10:50:59.637124000000	2013-05-05 14:36:16.637149000000
88880001	50000.00	2013-12-30 10:50:59.637124000000	2013-12-30 10:50:59.637124000000	2014-01-05 10:50:59.611224000000	2013-12-30 10:50:59.637124000000
88880001	9000.00	2014-01-05 10:50:59.611224000000	2014-01-05 10:50:59.611224000000	2014-03-05 21:12:23.321216000000	2014-01-05 10:50:59.611224000000
88880001	1000.00	2014-03-05 21:12:23.321216000000	2014-03-05 21:12:23.321216000000	2014-09-01 14:01:11.111231000000	2014-03-05 21:12:23.321216000000
88880001	100.00	2014-09-01 14:01:11.111231000000	2014-09-01 14:01:11.111231000000	2014-12-20 10:05:18.617454000000	2014-09-01 14:01:11.111231000000

28

Temporal – more example queries

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

```
SELECT T1.BALANCE AS BALANCE_2013,
       T2.BALANCE AS BALANCE_2014
FROM account FOR SYSTEM_TIME AS OF '2013-12-31' T1,
     account FOR SYSTEM_TIME AS OF '2014-12-31' T2
WHERE T1.ACCT_ID = '88880001' AND T2.ACCT_ID = '88880001';
```

BALANCE_2013	BALANCE_2014
50000.00	60000.00

Temporal – more example queries

- Query all versions of rows for account 88880001

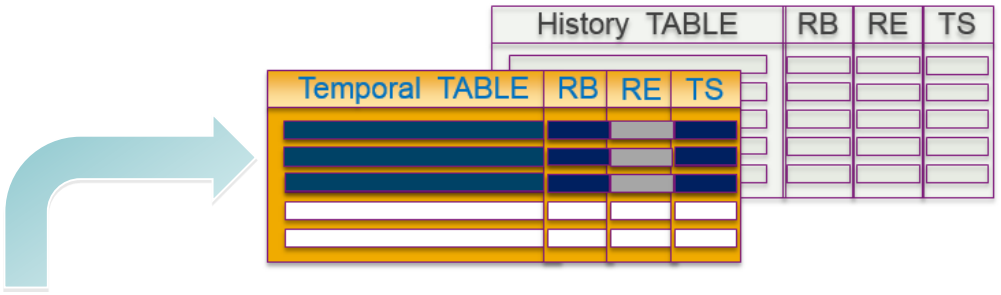
LAG is one of many new OLAP specifications added in IBM i 7.3

```
SELECT ACCT_ID,
        BALANCE,
        BALANCE  - LAG(BALANCE,1,0)
        OVER(ORDER BY TRANSACTION_TIME) AS CHANGES,
       TRANSACTION_TIME,
       ROW_DEATH
FROM account FOR SYSTEM_TIME
BETWEEN '0001-01-01' AND '9999-12-30'
WHERE ACCT_ID= '88880001'
ORDER BY transaction_time ASC;
```

ACCT_ID	BALANCE	CHANGES	TRANSACTION_TIME	INSTANCE_END
88880001	3000.00	-2990.00	2013-01-02 10:02:16.987139000000	2013-05-05 14:36:16.637149000000
88880001	10.00	49990.00	2013-05-05 14:36:16.637149000000	2013-12-30 10:50:59.637124000000
88880001	50000.00	-41000.00	2013-12-30 10:50:59.637124000000	2014-01-05 10:50:59.611224000000
88880001	9000.00	-8000.00	2014-01-05 10:50:59.611224000000	2014-03-05 21:12:23.321216000000
88880001	1000.00	-900.00	2014-03-05 21:12:23.321216000000	2014-09-01 14:01:11.111231000000
88880001	100.00	59900.00	2014-09-01 14:01:11.111231000000	2014-12-20 10:05:18.617454000000
88880001	60000.00	-60000.00	2014-12-20 10:05:18.617454000000	9999-12-30 00:00:00.000000000000

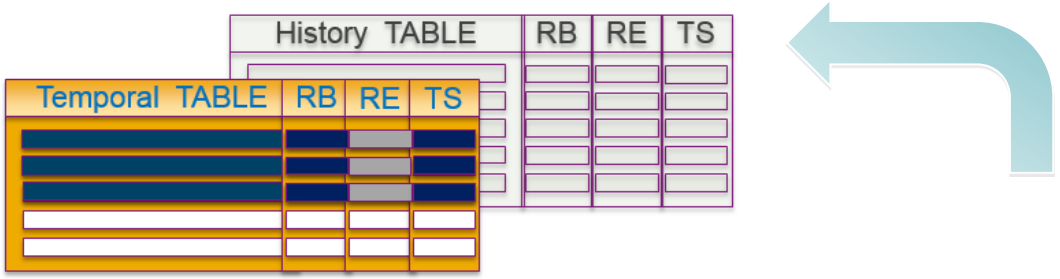
Temporal – System-period temporal table details

- Can be either a DDS-created physical file or an SQL table
- Associated with a single history table
- Must be journaled
- Generated columns can be IMPLICITLY HIDDEN
- Things you **can do** while versioning is enabled:
  - ❖ Add columns or expand their width
  - ❖ Attach Partitions
- Things you **can't do** while versioning is enabled:
  - ❖ Add Generated columns
  - ❖ Drop Columns or reduce their width
  - ❖ Drop or Detach Partitions
  - ❖ Use DSPDBR or DSPFD to view temporal existence or details



Temporal – History table details

- Must be an SQL table and reside within the same library
- Must match the production table format
- Must be journaled
- Can be partitioned or non-partitioned
- Things you **can do** with history
  - ❖ Remove old history
    - DELETE
    - TRUNCATE
    - ALTER TABLE DROP PARTITION
    - ALTER TABLE DETACH PARTITION
- Things you **can't do** with history:
  - ❖ Drop, alter or change the history table
  - ❖ Use DSPDBR or DSPFD to view temporal existence or details



SYSTIME - Bind Option

Programs have a build time control for System Time Sensitivity:

- SYSTEM_TIME_SENSITIVE column within 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**
  - This means that the special register has no effect
- Programs re(built) on IBM i 7.3 are by default, **time sensitive**
  - This means that the special register has effect

Build time controls:

- Routines (SQL/External) → SET OPTION SYSTIME = *YES or *NO
- CRTSQLxxx → OPTION(*SYSTIME or *NOSYSTIME)
  - Specifies that references to system-period temporal tables in both static and dynamic SQL statements are affected by the value of the CURRENT TEMPORAL SYSTEM_TIME special register.
- RUNSQLSTM → SYSTIME(*YES or *NO)

CURRENT TEMPORAL SYSTEM_TIME – special register

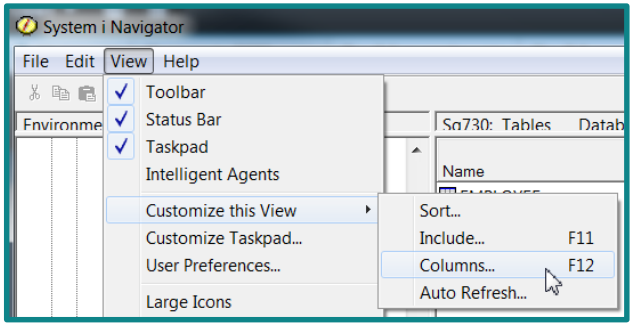
- The register affects any system-period temporal table in the query
  - Allows reuse of previous functions/procedures with new periods of time
  - Effects queries executed after setting the register
  - Works for external functions/procedures (C/C++/RPG)
  - When this register set to a non-null value:
    - Explicit time specification cannot be used within the SQL query
    - Cursors not updatable

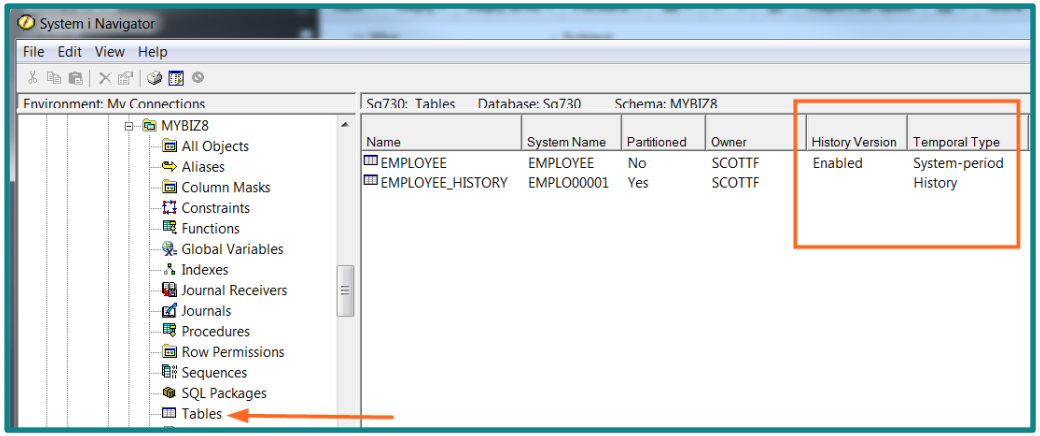
```
SET CURRENT TEMPORAL SYSTEM_TIME = '2014-09-02';

SELECT * FROM account WHERE ACCT_ID = '88880001';
```

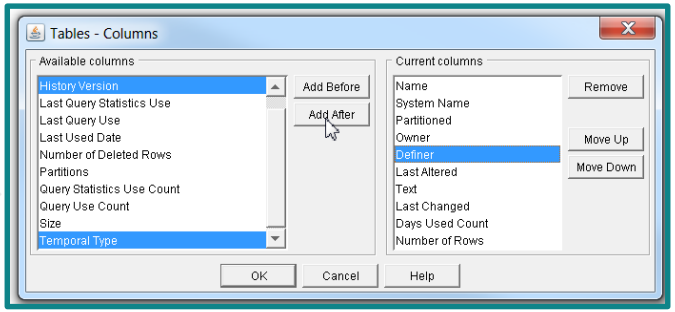
ACCT_ID	BALANCE	TRANSACTION_TIME	INSTANCE_BEGIN	INSTANCE_END	TRANSACTION_ID
88880001	100.00	2014-09-01 14:01:11.111231000000	2014-09-01 14:01:11.111231000000	2014-12-20 10:05:18.617454000000	2014-09-01 14:01:11.111231000000

System i Navigator and Temporal

Schemas → Tables … Add Temporal columns to your Navigator view

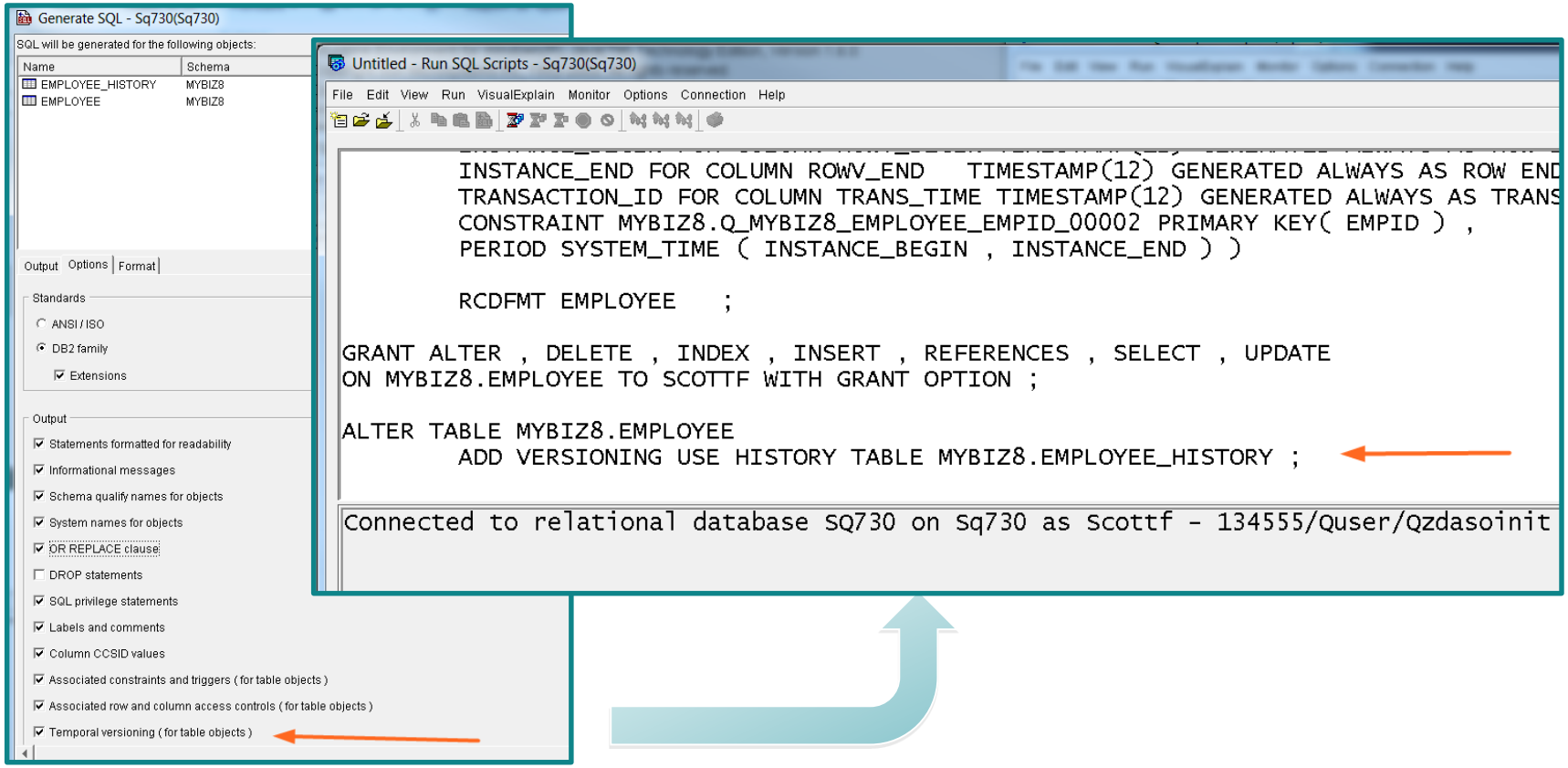






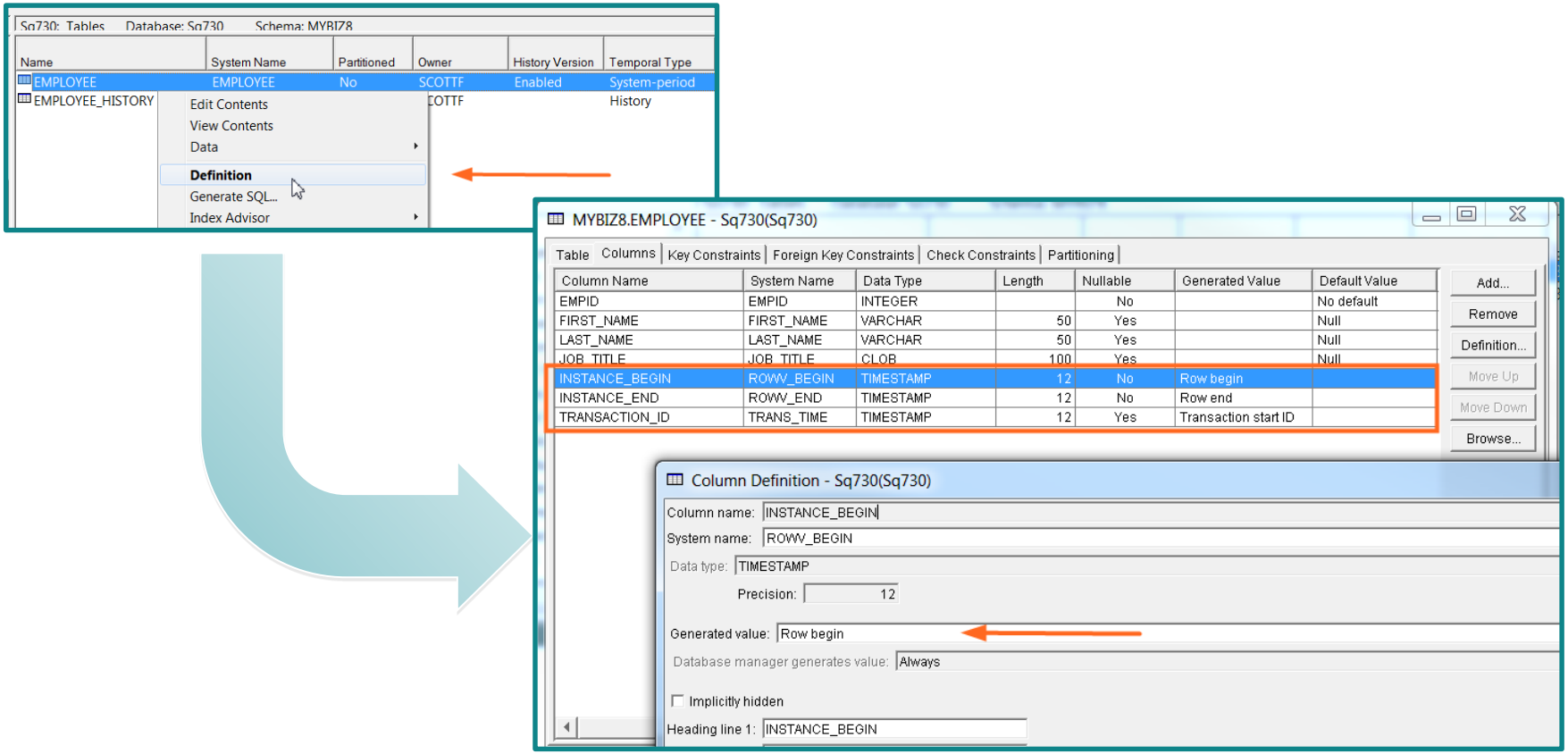
System i Navigator and Temporal

Generate SQL …
Use the Temporal versioning option to generate complete SQL



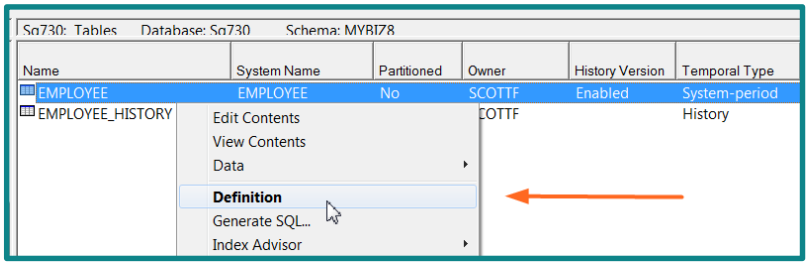
System i Navigator and Temporal

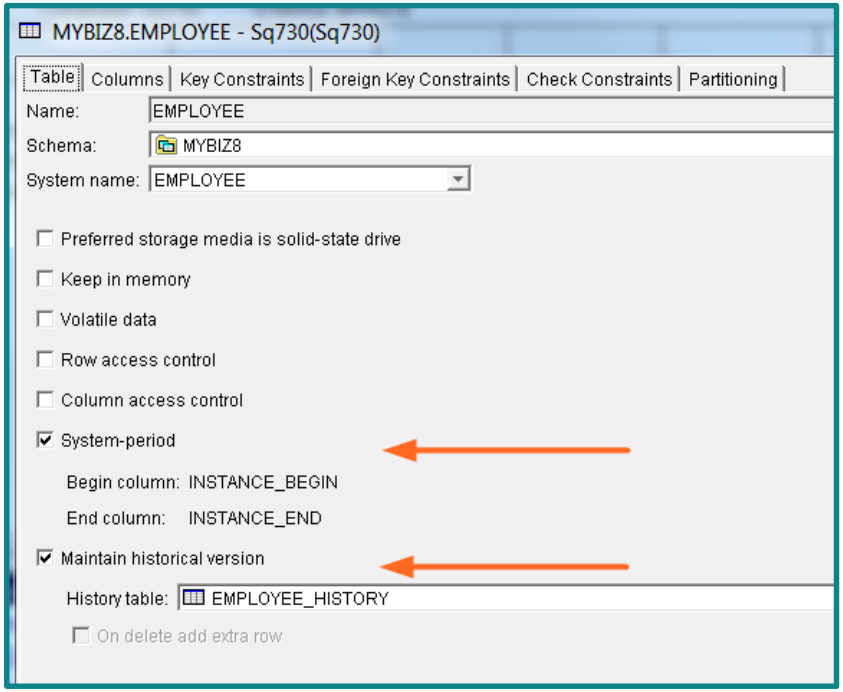
Table Definition… Add the three required system generated columns



System i Navigator and Temporal

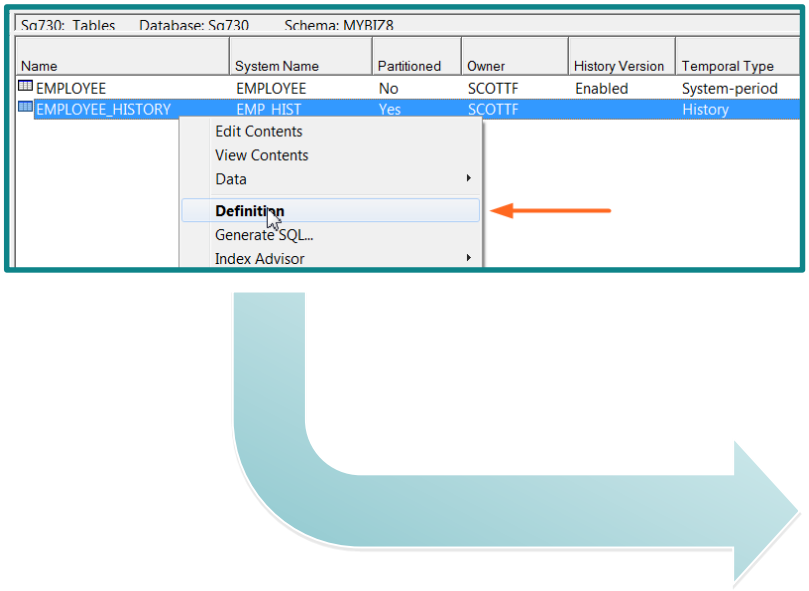
Table Definition… Establish System-period columns and declare the history table

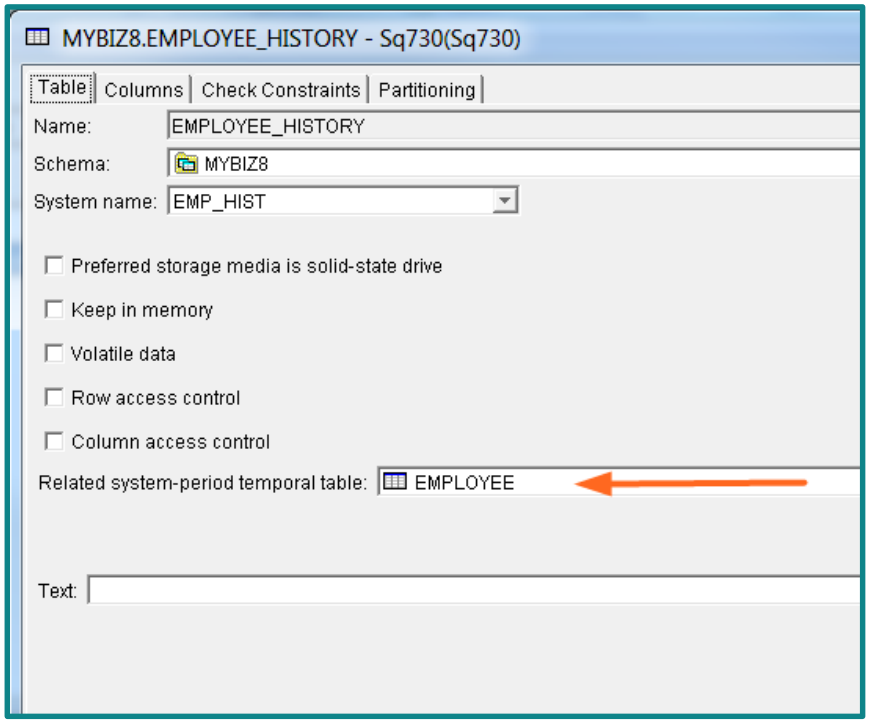




System i Navigator and Temporal

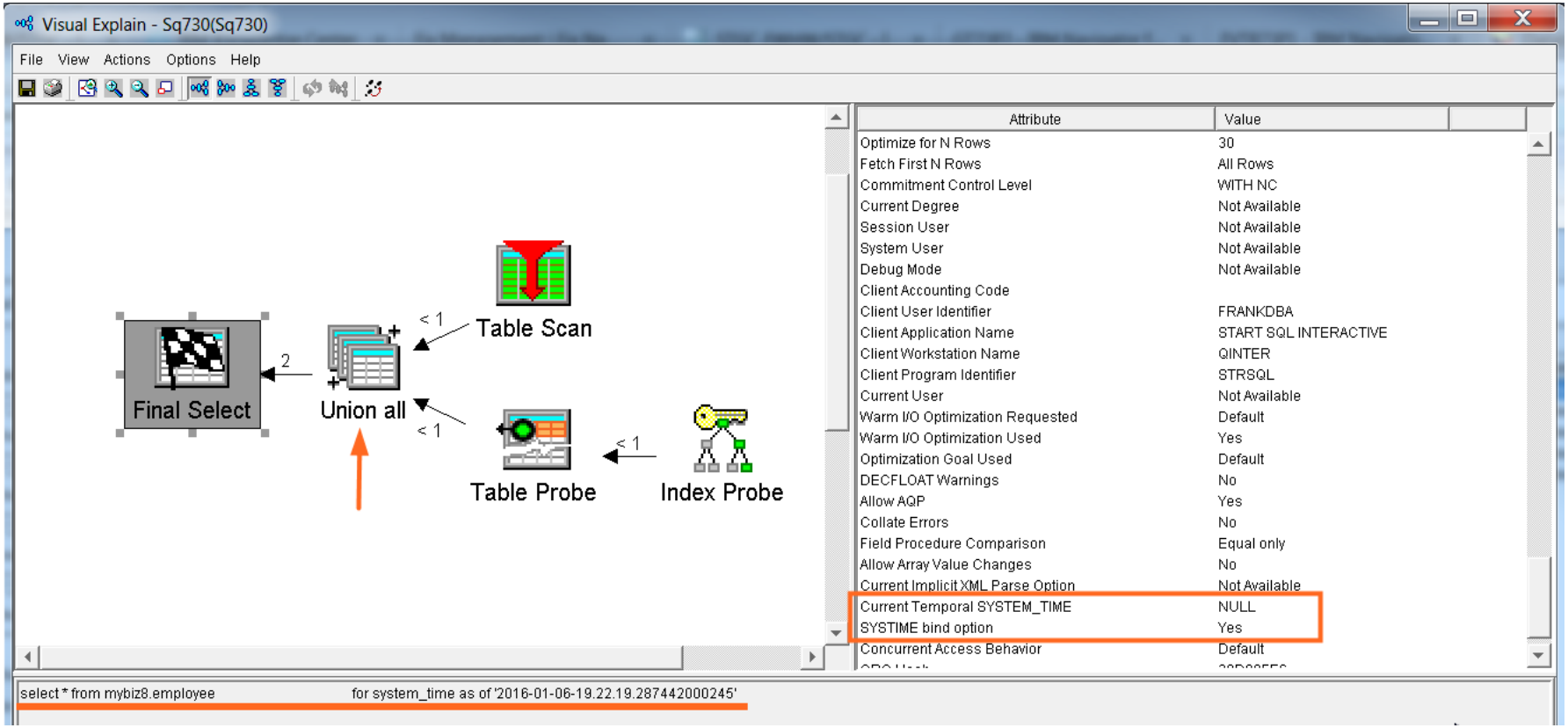
Table Definition… history tables contain a reference to the system-period temporal table





System i Navigator and Temporal

Visual Explain… shows the UNION ALL implementation and Temporal query controls



System i Navigator and Temporal

Users and applications are largely unaware that the history table exists

- SQL Query Engine unions in rows as needed

Consider using Range Partitioning for the History Table

- Organizing Historical rows by “Row End” is easy and has value
- Value: Faster save times, partition avoidance, smart use of IN MEMORY and ON SSD

Performance

- Create radix indexes over “Row Begin” and “Row End” columns

Native I/O

- Native read works against either the temporal or history table
  - Historical queries are unique to SQL
- Generated columns are safe to add
  - DB2 for i ensures the correct values are used

Temporal – Catalogs

- **QSYS2/SYSTABLES**

Contains a column called TEMPORAL_TYPE.

  - ‘S’ the table is a system-period
  - ’H’ the table is a history table
  - ’N’ the table is neither temporal or history

- **QSYS2/SYSCOLUMNS**

The HAS_DEFAULT column indicates the type of generated column

- **QSYS2/SYSPERIODS**

Contains one row for each table with a system period and identifies temporal and versioning information

- **QSYS2/SYSHISTORYTABLES**

Contains one row for each history table

Temporal – Save and restore

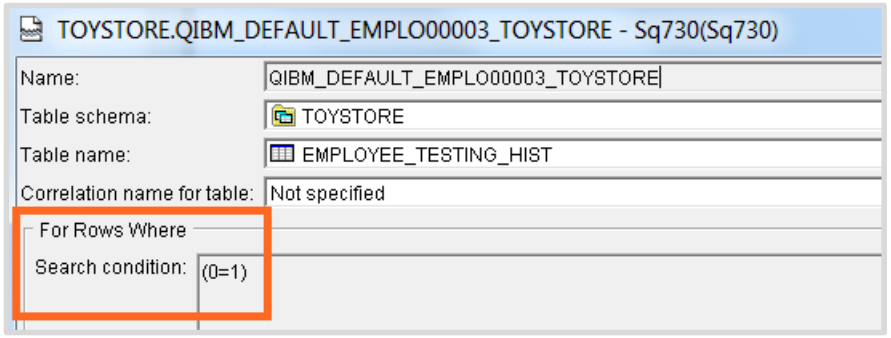
- The system-period temporal table and history table must be explicitly saved
- When a system-period temporal table is restored without its corresponding history table, the restored table's versioning relationship remains **defined** but is not established.

  Defined state will automatically change to versioned after both tables have been restored

- When in a defined state, the only operations that are allowed are:
  - ALTER TABLE ADD VERSIONING
  - ALTER TABLE DROP VERSIONING
  - DROP TABLE

Temporal – Row and Column Access Control

- When Row or Column Access Control (RCAC) is activated for a system-period temporal table, a **default row permission is activated on the history table** when versioning is added
- The default row permission prevents any direct user access to the history table



- Time specification queries use the RCAC rule(s) of the temporal table
- If you need to permit direct access to the history table, deploy additional Row Permissions and/or Column Masks on the history table

Temporal – Performance, Storage and more

How do you assess the impact to storage? What about the performance?

1. Analyze the volume of UPDATEs and DELETEs
2. Consider whether you're going to use ON DELETE ADD EXTRA ROW
3. Consider whether you'll add extra columns for auditing
4. Understand the record length of the file
5. Determine how long historical rows need to remain online
6. Decide whether you'll partition the history table
7. Decide whether to use the media or memory preferences
8. Determine your indexing strategy
9. Review the data model to identify dimension tables that should also be made temporal (repeat steps 1-8 for those tables)
10. Reflect on your HA strategy
    PowerHA → Business as Usual
    Logical Replication → Talk to your HA provider

Or… leverage the DB2 for IBM i Lab Services team of experts by contacting Mike Cain at mcain@us.ibm.com

Temporal – Performance, Storage and more

```
CREATE SCHEMA DBESTUDY;

CREATE OR REPLACE TABLE DBESTUDY.HISTORY_DETAIL
(TABLE_SCHEMA VARCHAR(128),
 TABLE_NAME VARCHAR(128),
 POINT_IN_TIME TIMESTAMP,
 UPDATE_OPERATIONS BIGINT,
 DELETE_OPERATIONS BIGINT) ON REPLACE DELETE ROWS;

--
-- execute this insert once per day
--
INSERT INTO DBESTUDY.HISTORY_DETAIL
  SELECT 'TOYSTORE5', 'SALES', CURRENT TIMESTAMP,
         UPDATE_OPERATIONS, DELETE_OPERATIONS
    FROM QSYS2.SYSTABLESTAT
  WHERE TABLE_SCHEMA = 'TOYSTORE5' AND
        TABLE_NAME   = 'SALES';
```

Finding the previous instance of a row

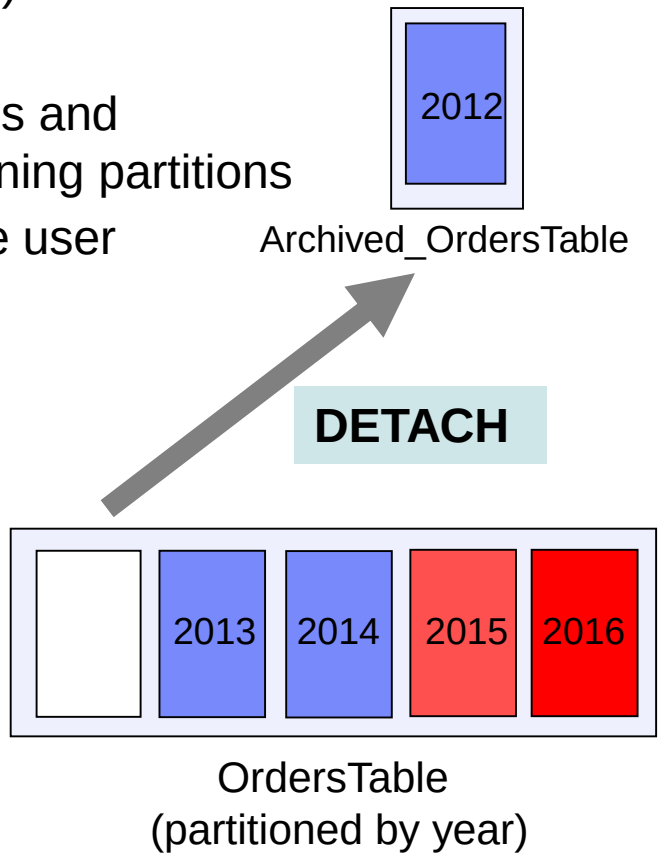
```
--
-- Find the previous instance of the row
--
CREATE OR REPLACE VARIABLE WHAT_TIME_IS_IT
   TIMESTAMP(12);

-- Extract the row birth time for the current row
-- and remove the timestamp uniqueness
SET WHAT_TIME_IS_IT =
   (SELECT TIMESTAMP_FORMAT(VARCHAR(ROW_BIRTH),
      'YYYY-MM-DD HH24:MI:SS:FF12',6)
       FROM EMPLOYEE
       WHERE EMPNO = '000010');

SELECT *
   FROM EMPLOYEE FOR SYSTEM_TIME AS OF
      TEMPTST1.WHAT_TIME_IS_IT
   WHERE EMPNO = '000010';
```

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

DETACH PARTITION – Dependent object rules

Dependent objects on the **source** table (OrdersTable)

- Views are rebuilt to use the remaining partitions
- DDS-created logical files that reference all partitions and Spanning SQL indexes are rebuilt to use the remaining partitions
- MQTs are retained, but need to be refreshed by the user



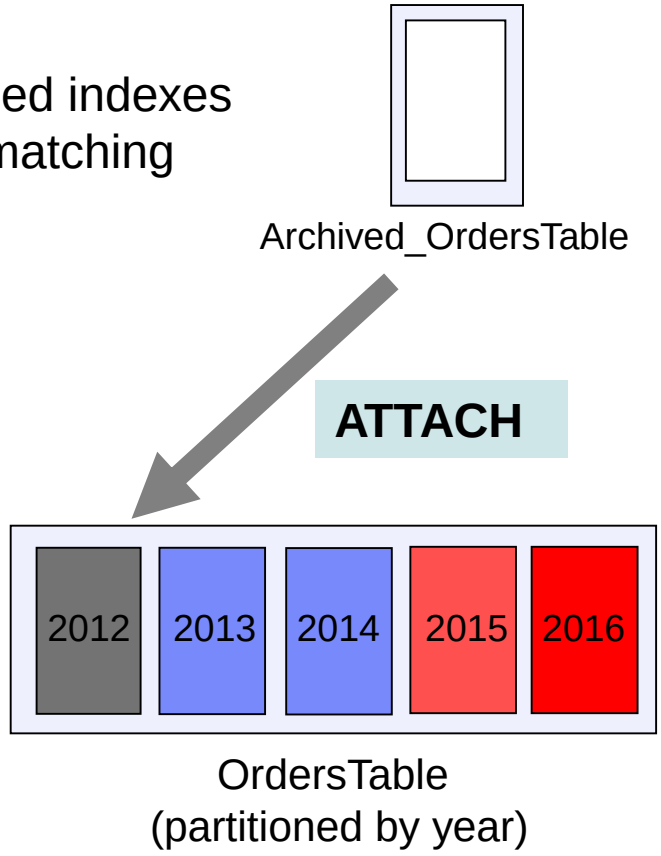
Usage details

- Cannot be a system-period temporal table
- Constraints are not added to the target table
- Privileges are not propagated to the target table
- When RCAC is active, a default row permission is activated on the target table
- An Identity column will not be an identity column in the target table

ATTACH PARTITION – Dependent object rules

Dependent objects on the **source** table (Archived_OrdersTable)

- Views and MQTs are discarded
- Partitioned indexes which correspond with partitioned indexes on the target are retained, as long as they have a matching logical page size
- Active RCAC must match on the source and target



Usage details

Dependent objects on the **target** table (OrdersTable)

- Views are rebuilt to include the new partition
- Spanning indexes are rebuilt
- MQTs are retained, but need to be refreshed
- Partitioned indexes, with no corresponding partitioned index on the source are modified to accommodate for the new partition