DB2 Temporal Tables

Its about time!

Andy Youens

FormaServe Systems Ltd

IBM Champion 2021, 2022 & 2023

Presenter - Andy Youens

FormaServe

- IBM i Consultant/Instructor at FormaServe for over 33 years
- Over 40 years IT experience working with S/34, S/36, AS400, iSeries & IBM i
- Specialties IBM i RPG PHP DB2 & Open Source on IBM i
- Instructor & an article writer
- IBM Champion 2021, 2022 & 2023 - Certified IBM Developer
- Prior to IT, Andy is proud to have served 10 years in the Royal Navy

Temporal Tables - Its about time!

- Introduce in DB2 for i in 2016
- As part of Version 7.3 release
- I gave the first workshop on Temporal as part of the iUG summer event 2016
- Also in other databases
- Not widely used or known about

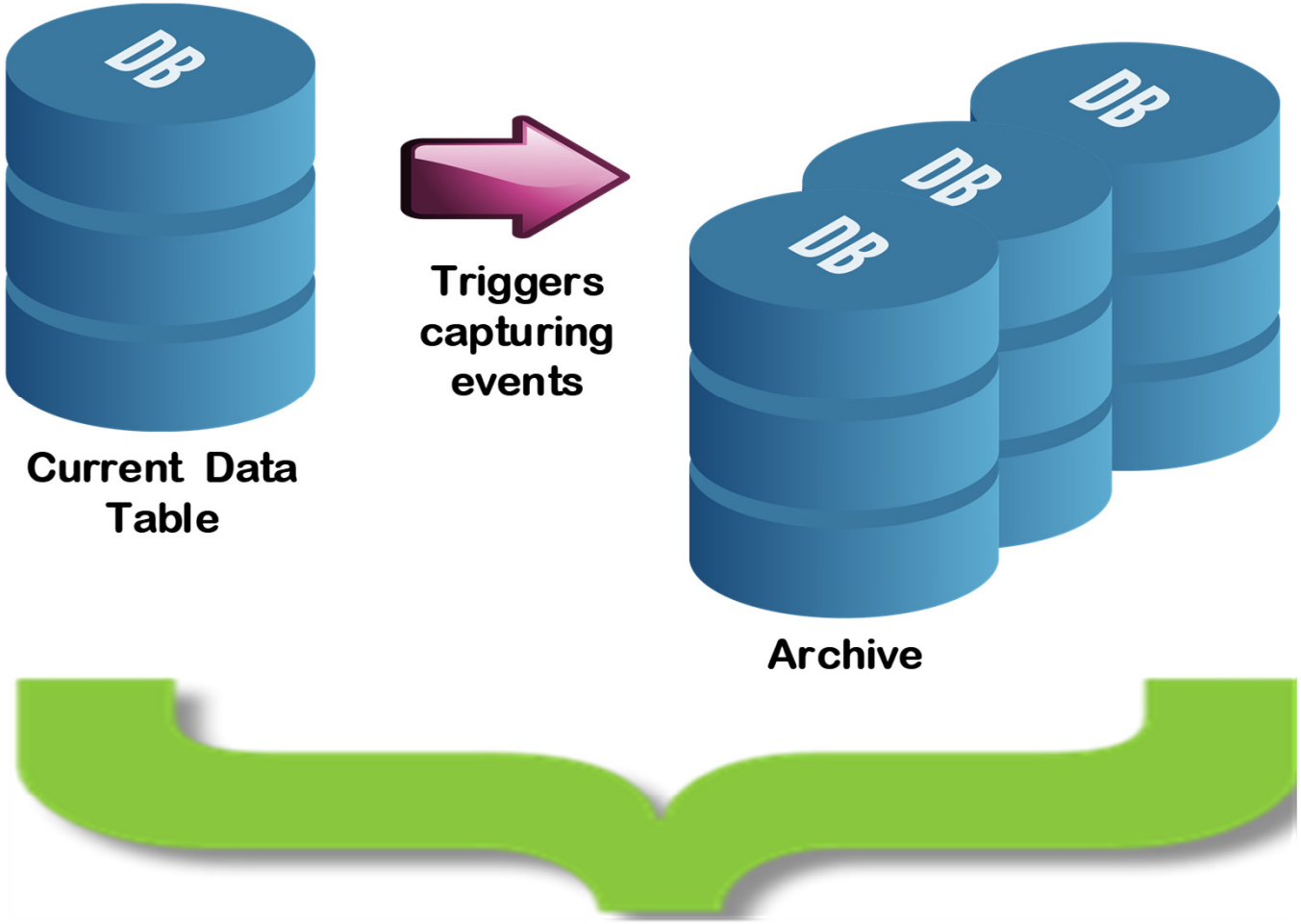
Temporal Tables - Its about time!

- They are not temporary tables
- Allows you to query historical data
- Captures the lifespan of a record based on the dates
- Use temporal tables & save yourself some time!

Can Answer These Questions?

- Do we have a complete audit of all data changes?
- Can you report the price of an item at any specific time?
- What were the prices of our stock this time last year?
- What were the status of the accounts, before we merged?
- How long was a product sold at a particular rate?
- Show me all the stages a support ticket went through before closing?
- One of our clients use them to store currency exchange rates

Original Method



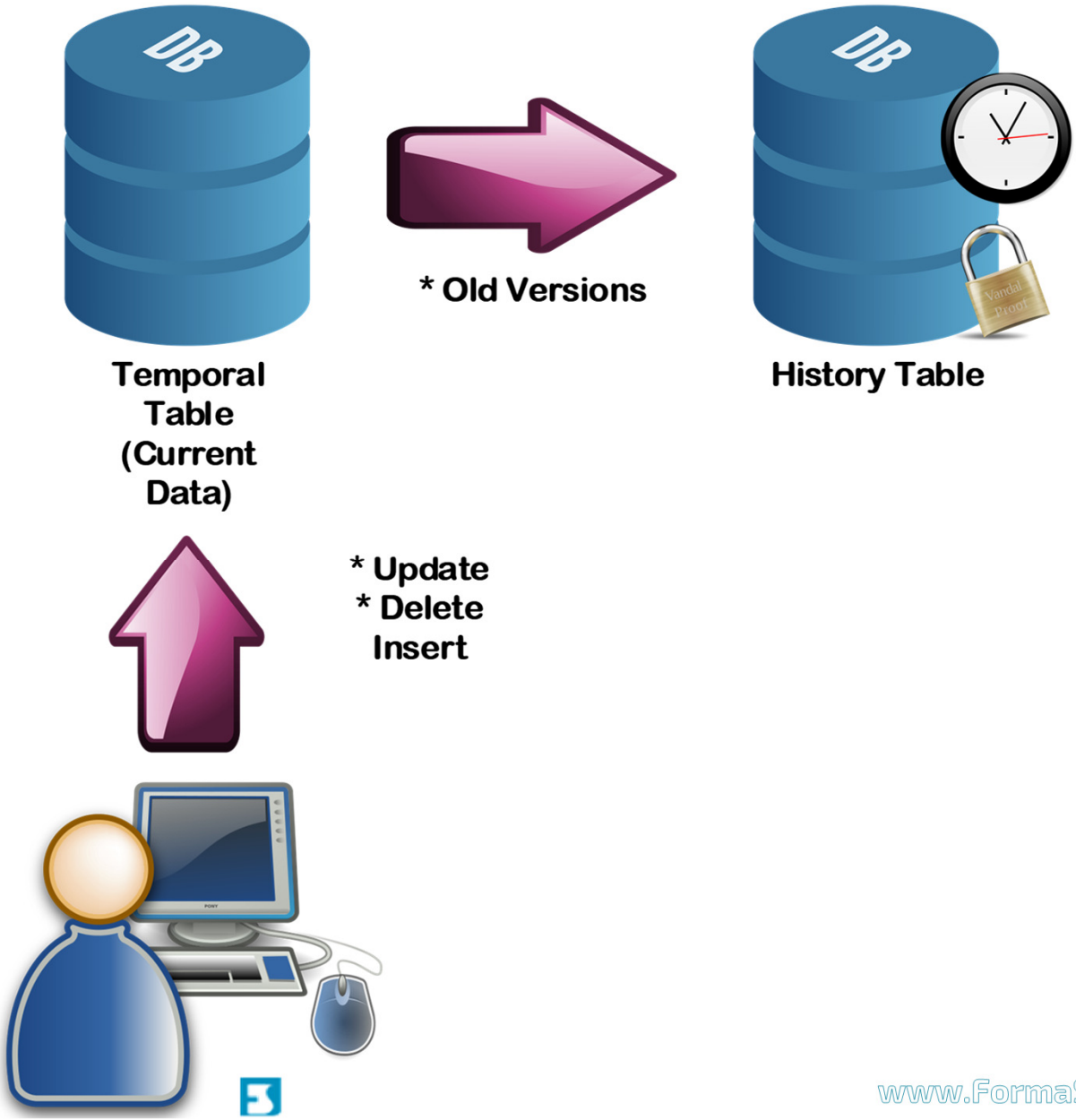
What are Temporal Tables?

- Temporal data refers to records that are versioned
- For any given record, there is a current version & possibly older versions
- We have live rows & dead rows!
- All managed by the database – Not by journaling!
- Also know as System-Period Temporal Tables

What are Temporal Tables?

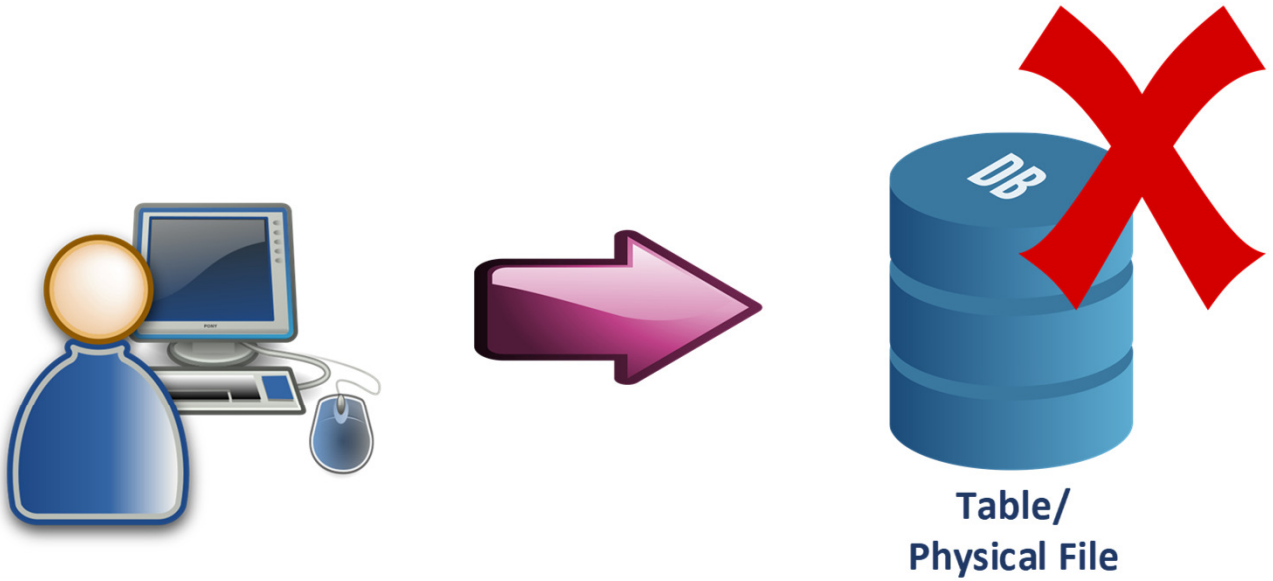
- A temporal table looks just like a normal table or PF
- Supports column types, normal indexes, foreign keys, etc.
- Insert, read, update, delete style operations work as expected
- Temporal tables are really 2 tables:
  - One contains the current values
  - The other handles old versions

What are Temporal Tables?



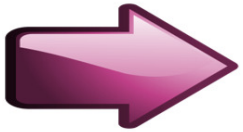
Normal Table

- After 4 transactions
  - 1 Insert
  - 2 Updates
  - 1 Delete



No record of transactions exist!

With Temporal Support



Temporal Table

No record of transactions exist!

All records can be seen

History Table

History table captures 3 records

1. After the 1st change
2. After the 2nd update
3. The deleted row

Its Not Journaling!

- 'I can do that already with journaling!'
- Journaling & temporal are two totally different methods
- Journaling does not produce relational data that you can run a SQL statement over
- The main purpose of journaling is transaction logging & recovery, often used for replication
- Journalling is the wrong tool for the job!

Temporal Works On Timestamps!

- 3 timestamp columns indicate the lifespan of row
- **Row Begin -** time when the row became current
- **Row End -** time when the row ceased to be current
- **Transaction Start ID -** A unique timestamp value
- Created using the **Generated Always As** clause
- Must defined as a **Timestamp(12)** column
- **All populated by the system clock**

System Time Period

- The System_Time Period column indicates the time span when the version of a row was current
- Defined by a row begin & a row end column
- Maintained by the database manager

Creating Temporal Table Steps

- To create & use temporal tables, you must follow the 3 stages below
  - Create table with additional system timestamp columns
  - Create a history table
  - Link the two together
- The temporal & its history file must exist in the same schema/library
- Both files must be journalled

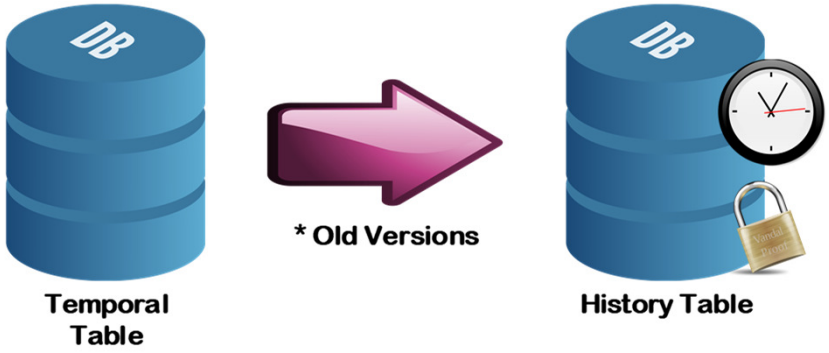
Creating A New Temporal Table

- Add 3 additional timestamp columns to the table
- Add a Period System_Time register

```
CREATE TABLE Contact_Info
  ( Company_no Int NOT NULL,
        Name    Char(50),
        Email   Char(50),
        Tel_No Char(25),
        Sys_Start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
        Sys_End     TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
        Ts_ID       TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID,
        PERIOD SYSTEM_TIME (Sys_Start, Sys_End) );
```

Additional Columns

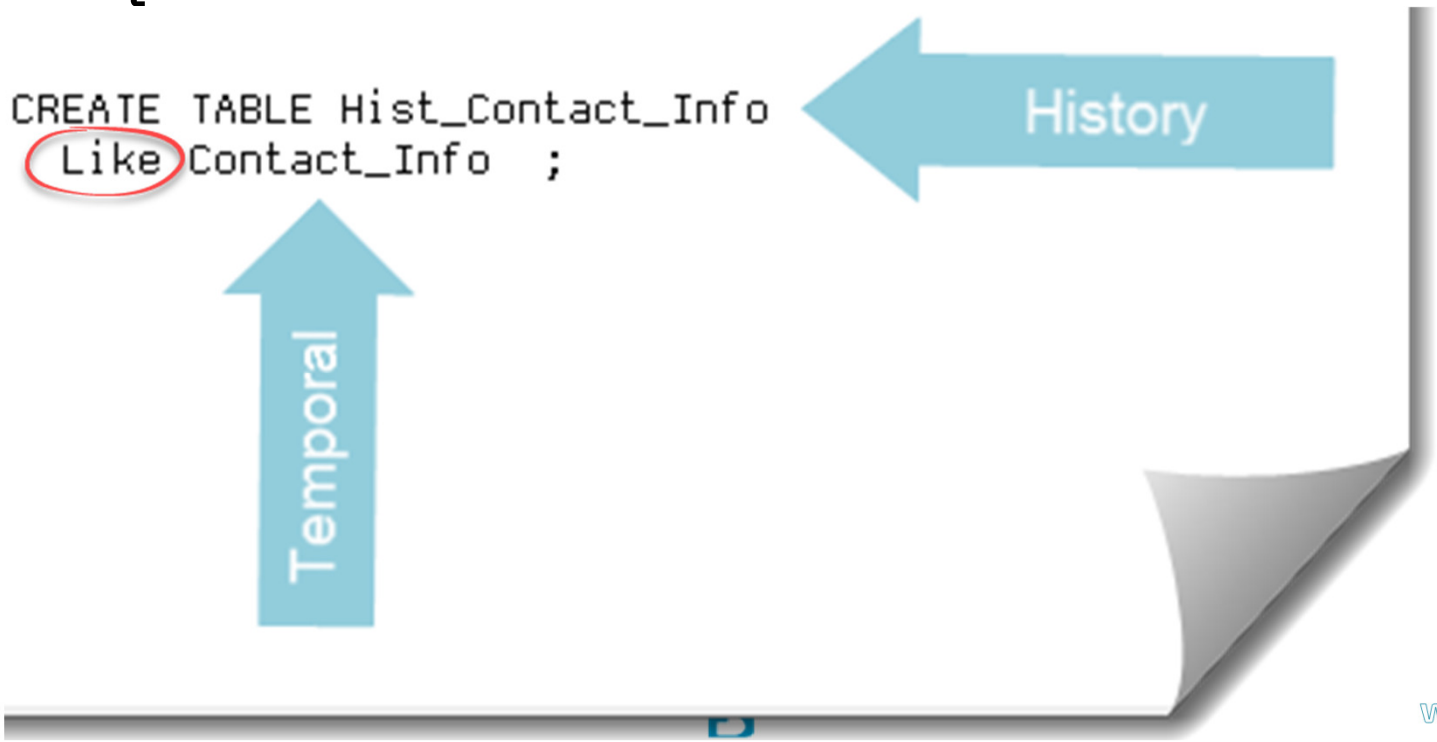
History Tables



- Temporal tables are linked to a history table
- History tables save old/dead rows
- When a row is updated/deleted, DB2 inserts a copy of the old row into its history table & timestamps it
- Gives ability to retrieve data from previous times
- Columns in the history table & temporal table must have the exact same attributes
- A history table cannot be attached to more than one temporal table
- History table is not referenced by queries

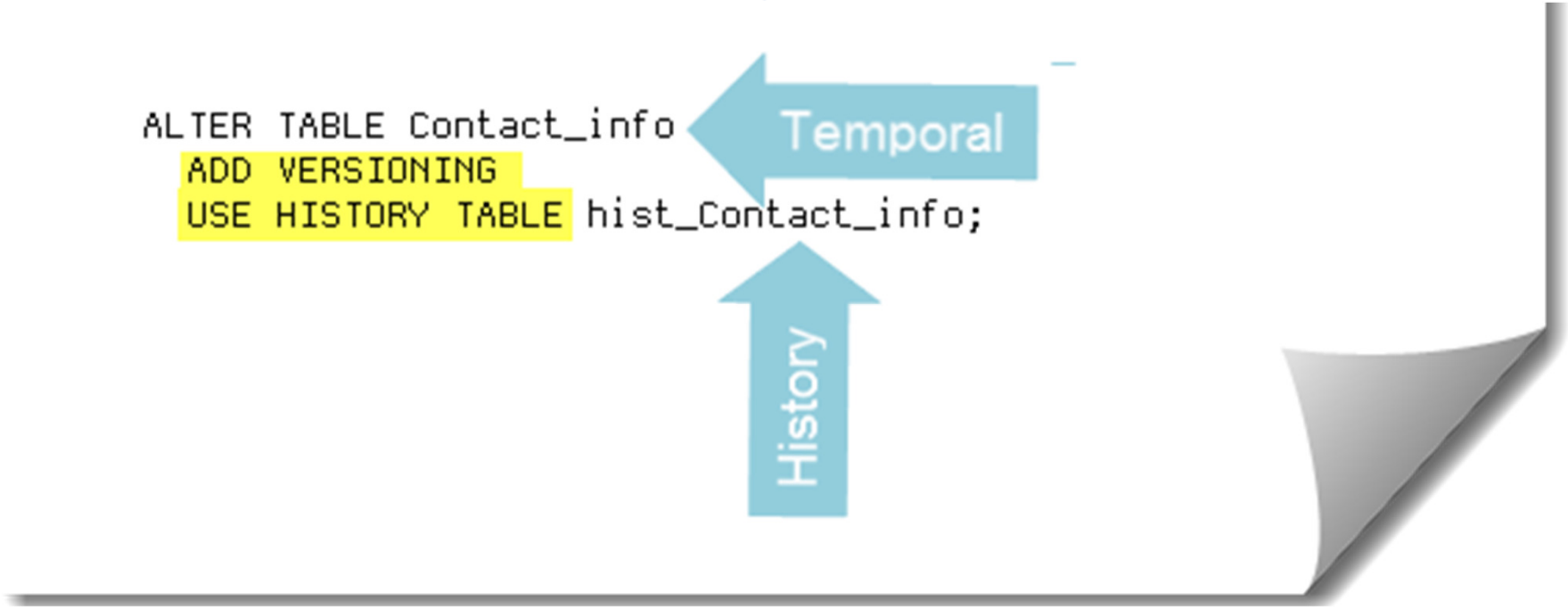
Create History

- Create the history table with all the attributes of the temporal table
- Use the SQL LIKE function to achieve this



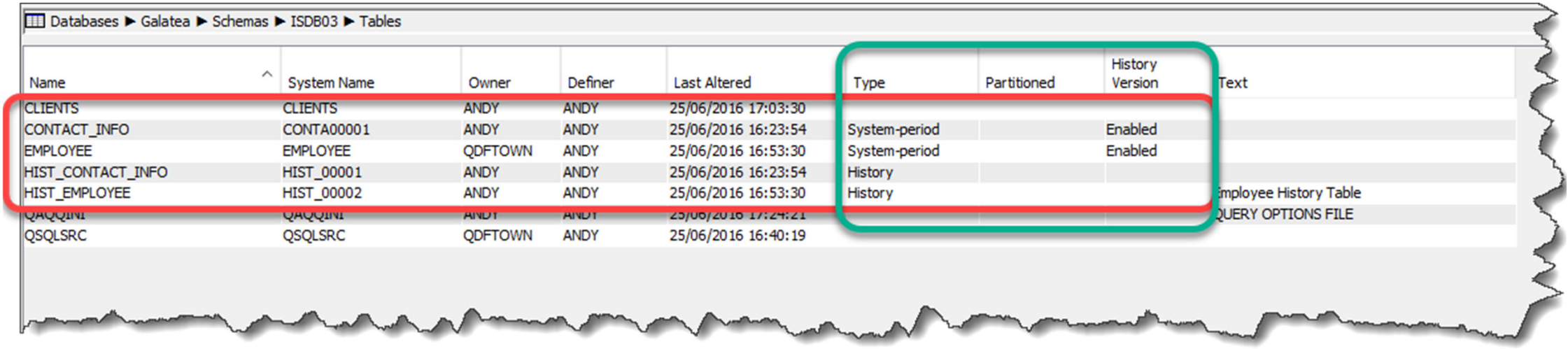
Link Temporal & History Tables

- Link the temporal to its history table using the Alter Table Function & Add Versioning



Creating Temporal - ACS Schemas

- So much easier using ACS!



Name	System Name	Owner	Definer	Last Altered	Type	Partitioned	History Version	Text
CLIENTS	CLIENTS	ANDY	ANDY	25/06/2016 17:03:30				
CONTACT_INFO	CONTA00001	ANDY	ANDY	25/06/2016 16:23:54	System-period		Enabled	
EMPLOYEE	EMPLOYEE	QDFTOWN	ANDY	25/06/2016 16:53:30	System-period		Enabled	
HIST_CONTACT_INFO	HIST_00001	ANDY	ANDY	25/06/2016 16:23:54	History			
HIST_EMPLOYEE	HIST_00002	ANDY	ANDY	25/06/2016 16:53:30	History			mployee History Table
QAQQINI	QAQQINI	ANDY	ANDY	25/06/2016 17:24:21				QUERY OPTIONS FILE
QSQLSRC	QSQLSRC	QDFTOWN	ANDY	25/06/2016 16:40:19				

Changing an Existing Table

- Add timestamp columns to your existing table
- Define Period System_Time to track when a row is current
- Does not show past history!
- Beware of level checks!

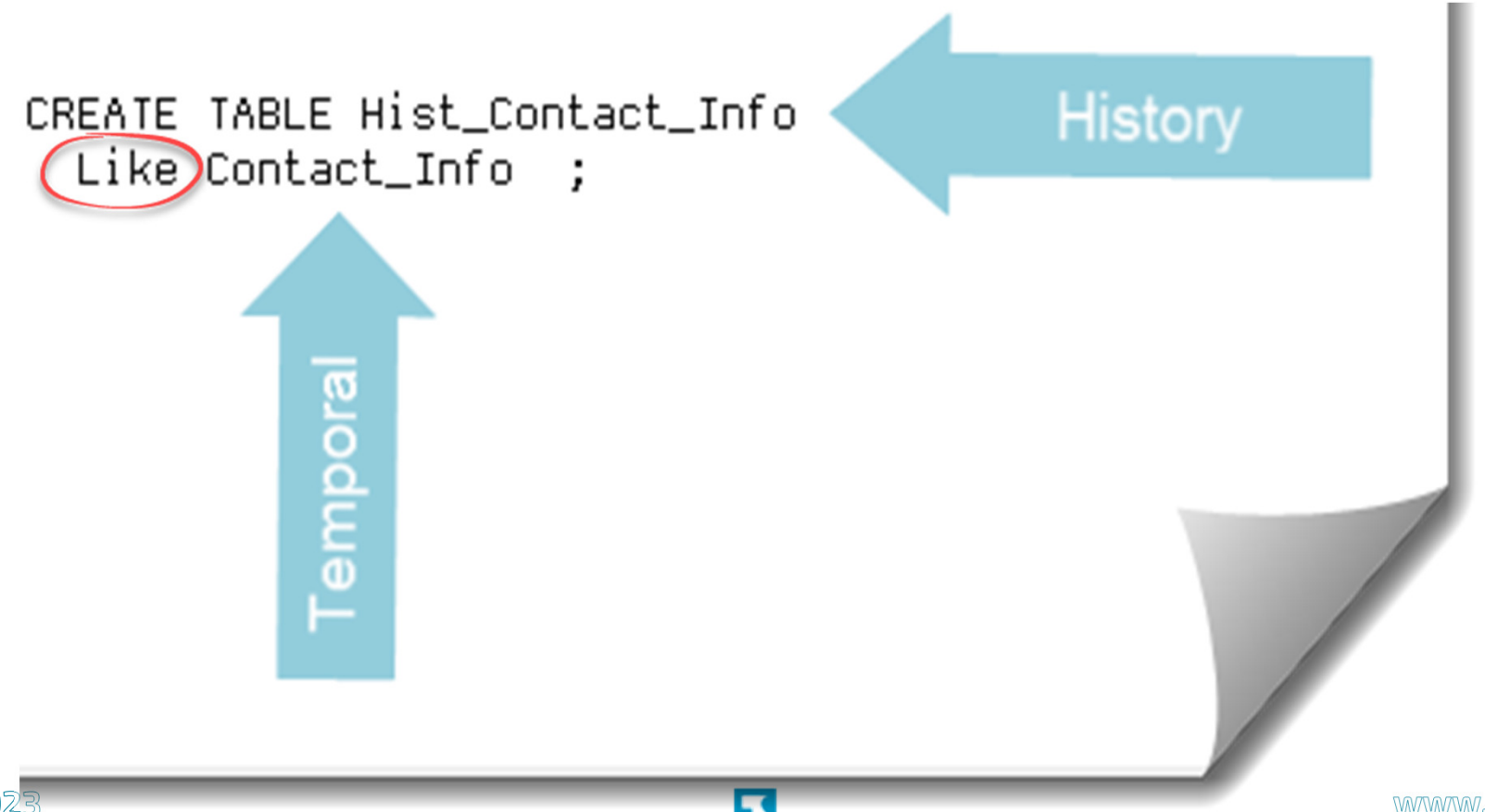
```
ALTER TABLE EMPLOYEE

        ADD COLUMN SYS_START    TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN
        ADD COLUMN SYS_END      TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END
        ADD COLUMN TS_ID        TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID

        ADD PERIOD SYSTEM_TIME(SYS_START, SYS_END)   ;
```

Create History Table

- As before, use the LIKE clause to create history table



Linking the Tables

- As before, link the temporal to the history
- Use the Alter Table syntax for this

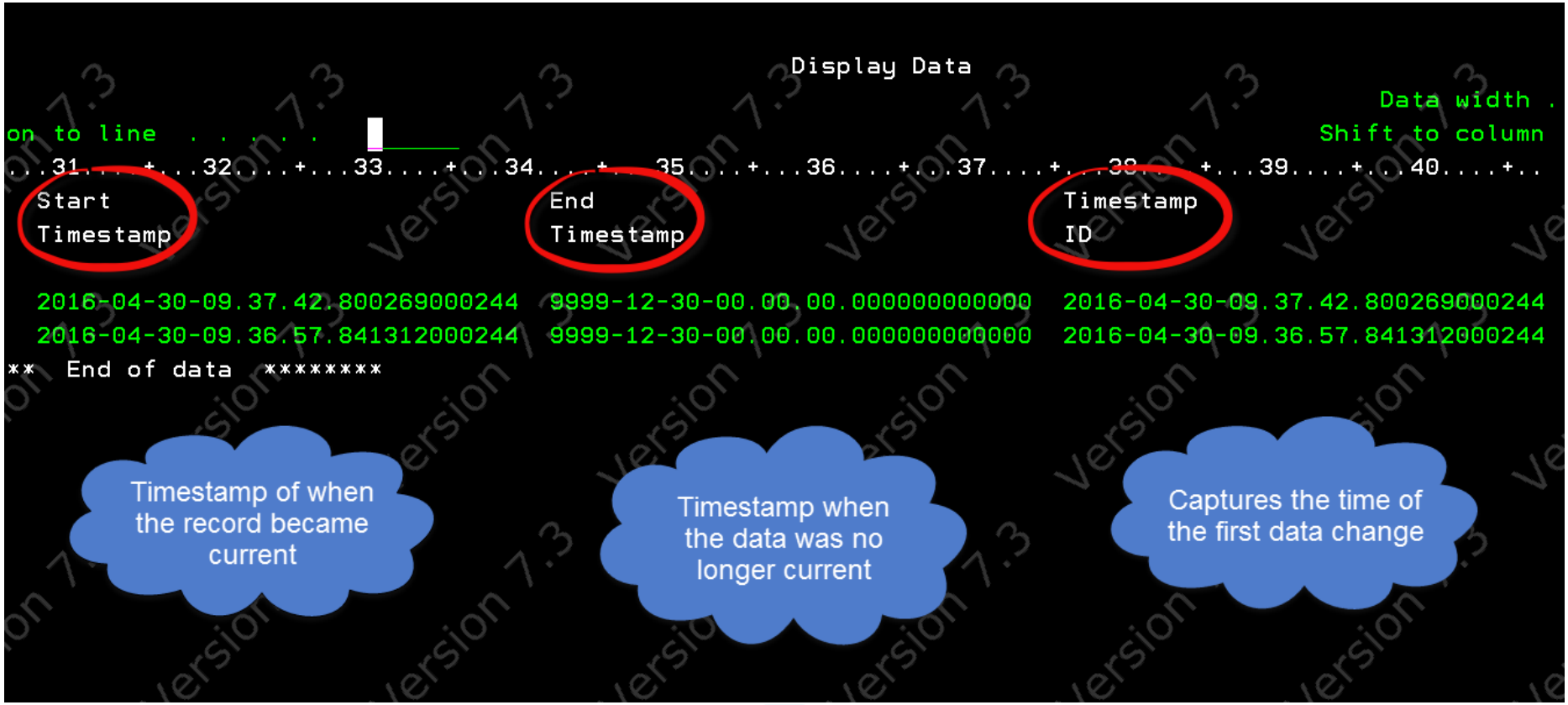
```
ALTER TABLE EMPLOYEE
        ADD VERSIONING
        USE HISTORY TABLE HIST_EMPLOYEE  ;
```

Inserting Data

- Inserting data into a temporal table is no different to inserting data into a regular table
- The database manager automatically generates the values for the Row Begin & Row End timestamp columns when data is inserted
- The database manager also generates the Transaction Start ID value that uniquely identifies the transaction that is inserting the row

Inserting - Timestamp Columns

- The database updates the timestamp columns



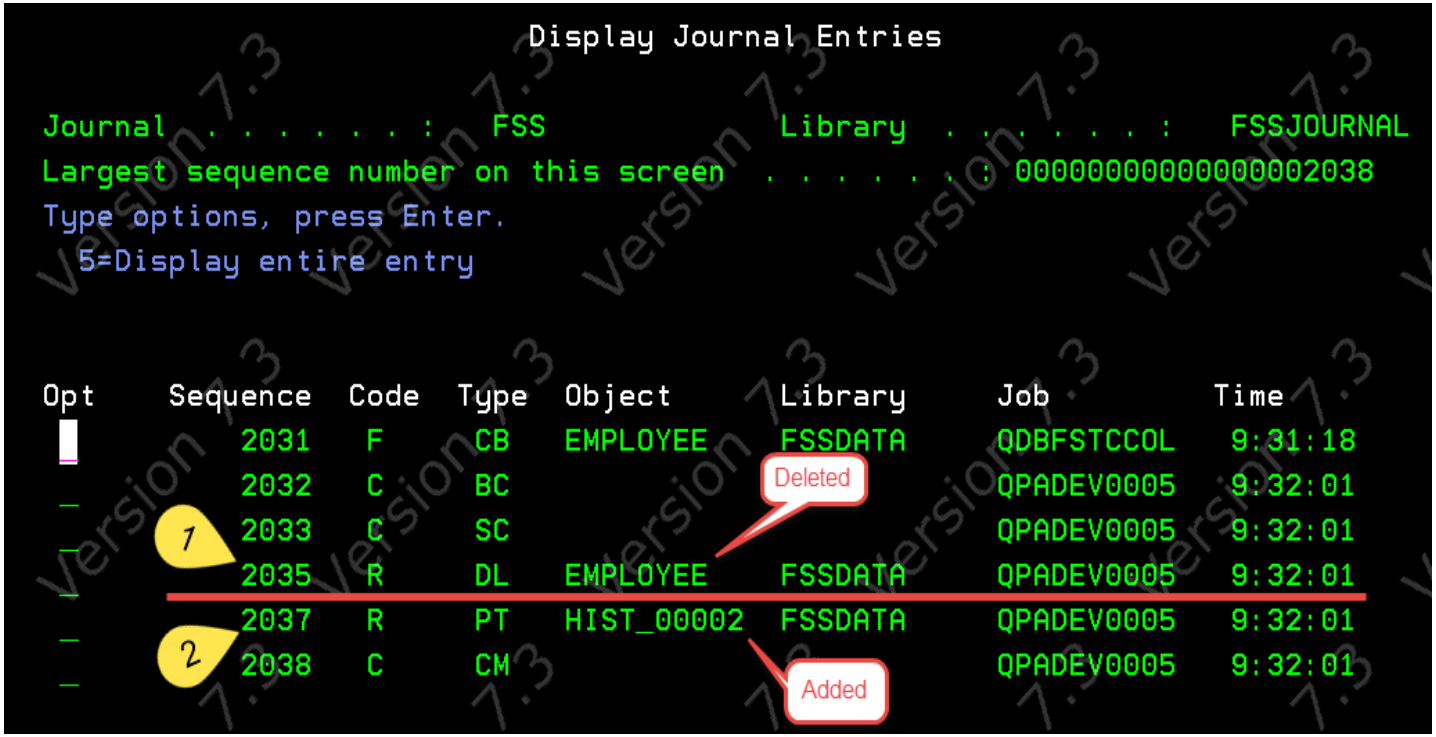
Updating Data

- In addition to updating the temporal table, the UPDATE statement causes the database manager to insert a copy of the existing row into its history table
- The history row is generated as part of the same transaction that updates the row

```
UPDATE FSSDATA/EMPLOYEE
  SET EMSAL = 4500.00, EMBON = 1000
  WHERE emid = 604  ;
```

Deleting Data

- Deleting a row from a temporal table removes the row from the table & adds a row to its history table



Querying Temporal Tables

Results can include current values & previous historic values

To query a temporal table, specify the time using **System_Time**

If you do not specify timestamp periods on the query, the database will act as a normal SQL statement

Querying Using For System_Time

- To query the current & past states
- Time periods can be specified in 3 ways:
  - **As Of** date/time
  - **From** date/time **To** date/time
  - **Between** date/time **And** date/time

Querying Using For System_Time

- Using As Of
- The result set will be a snapshot in time

```
/* Query Using As Of  */
Select * from Employee
    FOR SYSTEM_TIME AS OF '2016-04-30-12.00.00.000000000000'
    Where EmID = 606 ;
```

```
Employee   Surname              First     Start                          End                            Timestamp
ID                              Name      Timestamp                      Timestamp                      ID

     606   Youens               Chloe     2016-04-30-12.19.36.208596000244  2016-04-30-13.09.04.872120000244  2016-04-30-12.19
****  End of data  ********
```

The record as of 12:00 on 30/04

Querying Using For System_Time

- Using From & To
- The Begin Column & End Column of the period are exclusive

```
Select * from  EMPLOYEE
  FOR SYSTEM_TIME FROM '2016-04-30-00.00.00.000000000000'
       TO '2016-04-30-14.00.00.000000000000'
  Where emid = 606    ;
```

Display Data

```
ion to line  . . . . .
....1....+....2....+....3....+....4....+....31....+....32....+....33....+....34....+....35....+....36....+....37....+....38....+....39
Employee   Surname                First     Start                          End                            Timestamp
ID                                Name      Timestamp                      Timestamp                      ID

     606   Youens                 Chloe     2016-04-30-13.12.04.393234000244  9999-12-30-00.00.00.000000000000  2016-04-30-13.1
     606   Youens                 Chloe     2016-04-30-09.37.42.800269000244  2016-04-30-12.19.36.208596000244  2016-04-30-09.3
     606   Youens                 Chloe     2016-04-30-12.19.36.208596000244  2016-04-30-13.09.04.872120000244  2016-04-30-12.1
     606   Youens                 Chloe     2016-04-30-13.09.04.872120000244  2016-04-30-13.12.04.393234000244  2016-04-30-13.0
****  End of data  ********
```

Includes ALL current & history records

All records between 00:01 & 13:59 on the 30/04

Querying Using For System_Time

- Using **Between**
- The **Begin Column** of the period is **inclusive,** while the **End Column is exclusive**

```
Select * from  EMPLOYEE
  FOR SYSTEM_TIME BETWEEN '2016-04-30-12.00.00.000000000000'
    AND '2016-04-30-13.00.00.000000000000'
  Where emid = 606
```

```
tion to line  . . . . .
+....1....+....2....+....3....+....4...  .31.  +...32....+...33....+...34....+   35....+...36....+...37....+...38....+...39..
 Employee   Surname                First   Start                          End                            Timestamp
 ID                                Name    Timestamp                      Timestamp                      ID

      606   Youens                Chloe   2016-04-30-09.37.42.800269000244  2016-04-30-12.19.36.208596000244  2016-04-30-09.37.
      606   Youens                Chloe   2016-04-30-12.19.36.208596000244  2016-04-30-13.09.04.872120000244  2016-04-30-12.19.
***** End of data ********
```

All records between 12:00 & 12:59 on 30/04

Do I Have Any Temporal Tables?

```
select * from qsys2.systables where temporal_type <> 'N' ;
```

```
94
95 select * from Qsys2.Systables where temporal_type <> 'N' ;
96
```

TABLE_NAME	TABLE_OWNER	TABLE_TYPE	COLUMN_COUNT	ROW_LENGTH	TABLE_TEXT	LONG_COMMENT	TABLE_SCHEMA	LAST_ALTERED_TIMESTAMP	SYSTEM_TABLE_NAME	SYSTEM_TABL
CONTACT_INFO	ANDY	T	7	225		-	ISDB03	2016-06-25 16:23:54.890000	CONTA00001	ISDB03
HIST_CONTACT_INFO	ANDY	T	7	225		-	ISDB03	2016-06-25 16:23:54.879000	HIST_00001	ISDB03
HIST_EMPLOYEE	ANDY	T	25	304	Employee History Table	-	ISDB03	2016-06-25 16:53:30.187000	HIST_00002	ISDB03
EMPLOYEE	QDFTOWN	P	25	304		-	ISDB03	2016-06-25 16:53:30.021000	EMPLOYEE	ISDB03
HIST_AUDIT_LOG	FORMASERVE	T	4	608		-	TEMPORAL	2021-06-02 14:23:28.310000	HIST_00005	TEMPORAL
AUDIT_LOG	FORMASERVE	T	4	608		-	TEMPORAL	2021-06-02 14:23:28.320000	AUDIT_LOG	TEMPORAL
CONTACT_INFO	FORMASERVE	T	6	252		-	TEMPORAL	2020-01-09 10:01:01.665000	CONTA00001	TEMPORAL
HIST_CONTACT_INFO	FORMASERVE	T	6	252		-	TEMPORAL	2020-01-09 10:01:01.655000	HIST_00001	TEMPORAL
HIST_EMPLOYEE	FORMASERVE	T	24	302		-	TEMPORAL	2020-01-09 12:10:09.382000	HIST_00002	TEMPORAL
EMPLOYEE	FORMASERVE	T	24	302		-	TEMPORAL	2020-01-09 12:10:09.392000	EMPLOYEE	TEMPORAL

Done: 16 rows retrieved. 07/02/2023, 14:16:36

Stop Versioning


```
alter table employee Drop Versioning ;
```

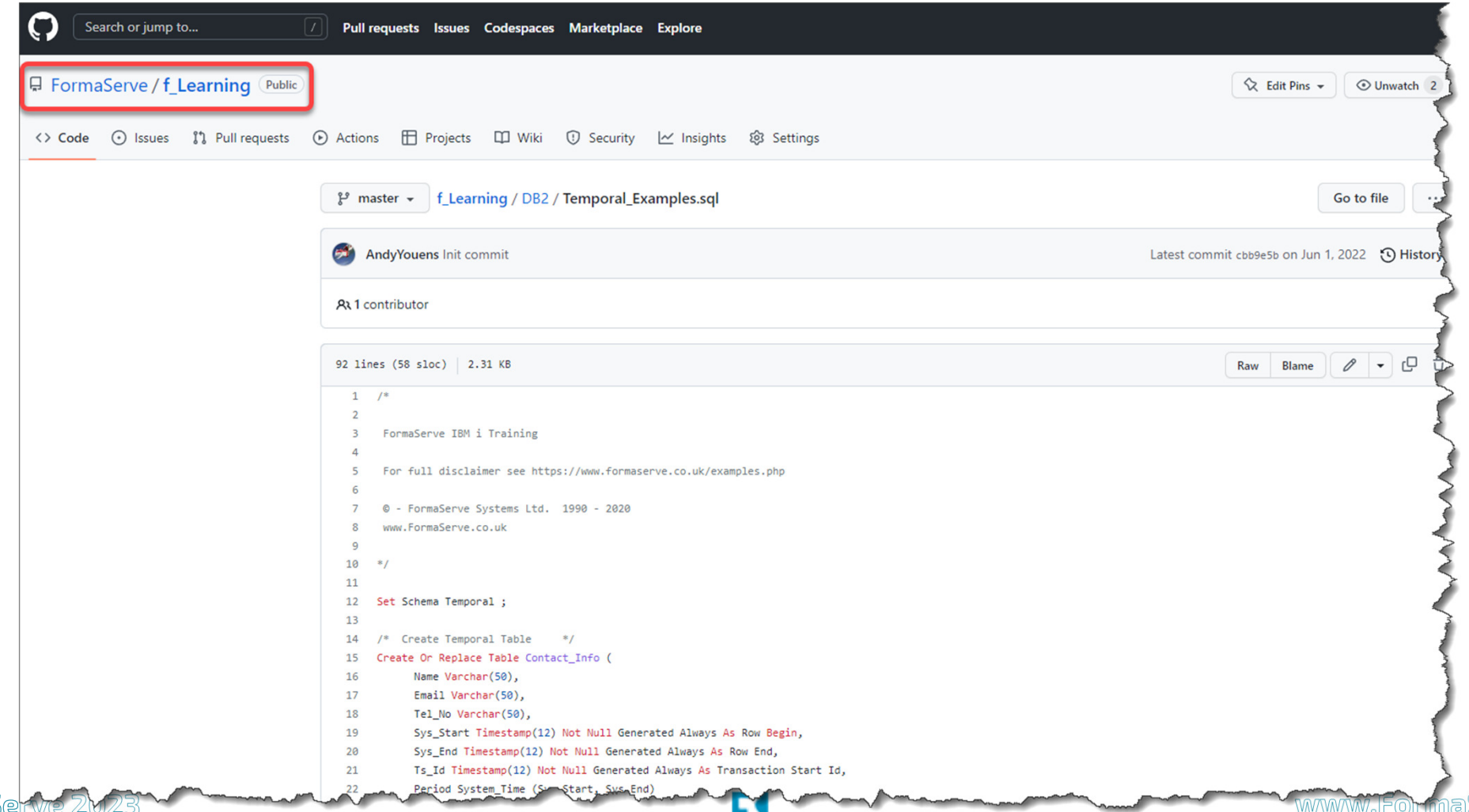
- The SQL **Drop Versioning** feature is used to stop temporal features
- This gives **two independent tables**
- The history table remains in place

Large History File?

- Add partitioning!
- Needs 5770SS1 Option 27 - DB2 Multisystem
- Now part of OS SWMA

```
Create Table Hist_Account Like Account
  Partition By Range (Row_End) (
      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
    );
```

GitHub Examples - FormaServe/f_Learning



Thank You!

- Andy Youens
- Andy@FormaServe.co.uk
- www.formaserve.co.uk
- 01908 109978

Support me on Ko-fi

https://learning.formaserve.co.uk
https://formaserve.github.io/