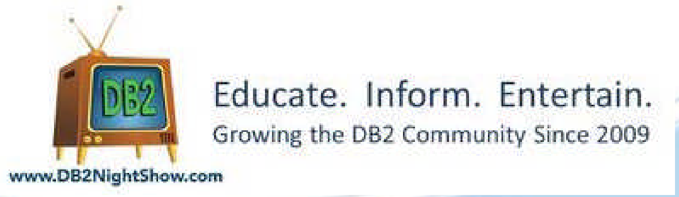
Information Management

IBM


DB2
Educate. Inform. Entertain.
Growing the DB2 Community Since 2009
www.DB2NightShow.com

The DB2Night Show Episode #77

Temporal Queries and Analytics in an IBM InfoSphere Warehouse V10 Environment

Pat Bates, WW Technical Sales for Big Data and Warehousing, jpbates@us.ibm.com

June 25, 2012

Temporal Tables - Business Benefits

- **Provides an increased business insight to clients**
  - Can access not just currently committed data, but data at any period of time since data collection inception
  - Can incorporate business logic and policies that are a function of time, like effective dates and validity enforcement

- **Provides a mechanism for data change tracking to meet data compliance rules**
  - Can easily determine **ALL** data values for a particular business entity over time (even deleted values)

- **Provides recovery for business data that was erroneously updated or deleted without having to perform complex database recovery scenarios**
  - Can retrieve data that was inadvertently deleted or updated and use this information to restore the affected data

Temporal Tables - Business Benefits

- **Provides lower application development and maintenance costs**
  - Reduce application logic and shorten application development time by eliminating custom solutions involving triggers, additional application logic, and increased data complexity
  -
- **Provides flexibility and application transparency for use even with packaged applications**
  - Does not require application changes in order to utilize Time Travel Query

Two Notions of Time and Events

- **Database (or System) Event**
  - Tracks events at the moment they occur in the database system
  - Useful to track data changes in the data warehouse
  - Examples include
    - Reporting on transaction-based processes
    - Auditability and history of changes to records in the warehouse
    - Recovery of updated / deleted information
- **Business Event**
  - Tracks events according to their applicability according to business rules
  - Useful for reporting and analytics based on when certain information is in force
  - Examples include

Simple Example

- **March 1**
  - John switches to a new car insurance
- **March 28**
  - The coverage amount of John's policy is increased, to be effective **April 1**
- **March 29**
  - John is involved in a minor accident
- **April 5**
  - John submits a claim for the damage to his car
- **April 10**
  - John requests a policy change to reduce his deductible from $500 to $250. This change takes effect **April 15**
- **April 17**
  - An agent at the insurance company reviews John's claim from April 5 to authorize payment
- **Which coverage amount and deductible should be applied?**

How to Define a System-Period Temporal Table

1. **CREATE a table with a SYSTEM_TIME attribute**

```
CREATE TABLE travel(
  trip_name CHAR(30) NOT NULL PRIMARY KEY,
  destination CHAR(12) NOT NULL,
  departure_date DATE NOT NULL,
  price DECIMAL (8,2) NOT NULL,
  sys_start TIMESTAMP(12) NOT NULL generated always as row begin implicitly hidden,
  sys_end TIMESTAMP(12) NOT NULL generated always as row end implicitly hidden,
  tx_start TIMESTAMP(12) generated always as transaction start id implicitly hidden,
  PERIOD SYSTEM_TIME (sys_start, sys_end)) in travel_space;
```

Captures the begin and end times when the data in a row is current

2. **CREATE the history table**

```
CREATE TABLE travel_history like travel in hist_space;
[ALTER TABLE travel_history APPEND ON;] OPTIONAL
```

3. **ADD VERSIONING to the system-period temporal table to establish a link to the history table**

```
ALTER TABLE travel
ADD VERSIONING USE HISTORY TABLE travel_history;
```

Insert Data Into a System-Period Temporal Table

- Add new trips: Amazonia, departing on 10/28/2011, and Ski Heavenly Valley, departing on 3/1/2011

Current Date = January 1, 2011

```
INSERT INTO travel VALUES ('Amazonia','Brazil','10/28/2011',1000.00)
INSERT INTO travel VALUES ('Ski Heavenly Valley', 'California','03/01/2011',400.00)
```

System validity period
(inclusive, exclusive)

trip_name	destination	departure_ date	price	sys_start	sys_end
Amazonia	Brazil	10/28/2011	1000.00	01/01/2011	12/30/9999
Ski Heavenly Valley	California	03/01/2011	400.00	01/01/2011	12/30/9999

Both `sys_start` and sys_end columns are inserted by DB2, not the application. For simplicity, they are represented here as `DATE`s, rather than `TIMESTAMP`s

Alter and Update a System-Period Temporal Table

- **Destination name is not explicit enough. Alter the `DESTINATION` column to make it longer**
  - Current Date = February 15, 2011

```
ALTER TABLE travel ALTER COLUMN destination SET DATA TYPE VARCHAR(50)
```

- **Now `UPDATE` the destination column for Ski Heavenly Valley to make it clearer**
  - Note: history table modification is automatically done by DB2

```
UPDATE travel SET destination = ‘Lake Tahoe, CA’
WHERE trip_name = ‘Ski Heavenly Valley’
```

New `sys_start` date

Base table

trip_name	destination	departure_date	price	sys_start	sys_end
Amazonia	Brazil	10/28/2011	1000.00	01/01/2011	12/30/9999
Ski Heavenly Valley	Lake Tahoe, CA	03/01/2011	400.00	02/15/2011	12/30/9999

System validity period inclusive, exclusive)

History table

trip_name	destination	departure_date	price	sys_start	sys_end
Ski Heavenly Valley	California	03/01/2011	400.00	01/01/2011	02/15/2011

DB2 inserted row into history table automatically and supplied `sys_start` and sys_end dates

Delete from a System-Period Temporal Table

- We are no longer offering the Ski Heavenly Valley trip – `DELETE` it
  - Current Date = April 1, 2011

```
DELETE FROM travel WHERE trip_name = ‘Ski Heavenly Valley’
```

Base table

trip_name	destination	departure_date	price	sys_start	sys_end
Amazonia	Brazil	10/28/2011	1000.00	01/01/2011	12/30/9999

Ski Heavenly Valley has been removed from base table

System validity period (inclusive, exclusive)

History table

trip_name	destination	departure_date	price	sys_start	sys_end
Ski Heavenly Valley	California	03/01/2011	400.00	01/01/2011	02/15/2011
Ski Heavenly Valley	Lake Tahoe, CA	03/01/2011	400.00	02/15/2011	04/01/2011

DB2 inserted row into history table automatically and supplied `sys_start` and sys_end dates

Query a System-Period Temporal Table
(These queries access the table on the previous page)

- **Query the past: what trips were available on 03/01/2011 for less than $500?**
  - Current date = May 1, 2011

```
SELECT trip_name FROM travel FOR SYSTEM_TIME AS OF '03/01/2011'
WHERE price < 500.00
```

  - Result: Ski Heavenly Valley

- **Query the present: what trips are currently available to Brazil?**

```
   SELECT trip_name FROM travel WHERE destination = 'Brazil'
```

  - Result: Amazonia

Defaults to the current table only - functions as if we added
FOR SYSTEM TIME AS OF CURRENT DATE

- **Query the past and the present: In 2011, how many different tours were offered?**

```
SELECT COUNT (DISTINCT trip_name) FROM travel
FOR SYSTEM_TIME BETWEEN '01/01/2011' AND '01/01/2012'
```

  - Result: 2

How to Define an Application-Period Temporal Table

- CREATE a table with a BUSINESS_TIME attribute

```
CREATE TABLE travel
         (trip_name CHAR(25) NOT NULL,
         destination CHAR(8) NOT NULL,
         departure_date DATE NOT NULL,
         price DECIMAL(8,2) NOT NULL,
         bus_start DATE NOT NULL,
         bus_end DATE NOT NULL,
         PERIOD BUSINESS_TIME (bus_start, bus_end),
         PRIMARY KEY (trip_name, BUSINESS_TIME WITHOUT OVERLAPS));
```

PERIOD (bus_start, bus_end) is (inclusive, exclusive)
The bus_start column in the PERIOD clause must be less than the bus_end column

trip_name plus the bus_start and bus_end PERIOD form a unique primary key.

DB2 enforces that there are no overlapping PERIODs for trip_name.

Insert Data into a Application-Period Temporal Table

- **Add new trip: Manu Wilderness, departing on 08/02/2011**
  - Current date = May 01, 2011

```
INSERT INTO travel VALUES (
‘Manu Wilderness’,’Peru’,’08/02/2011’,1500.00,’05/01/2011’,’01/01/2012’)
```

bus-start and bus_end columns are inserted by the application, not DB2

BUSINESS_TIME period (inclusive, exclusive)

trip_name	destination	departure_ date	price	bus_start	bus_end
Manu Wilderness	Peru	08/02/2011	1500.00	05/01/2011	01/01/2012

Bi-temporal Tables

- **Combine application-period (ATT) and system-period (STT) capabilities**
- **Every row has a pair of `TIMESTAMPs` (`SYSTEM_TIME period`) set by DB2 and a pair of `TIMESTAMP` or `DATE` columns (`BUSINESS_TIME period`) set by the application**

trip_name	Destination	departure_date	price	bus_start	bus_end	sys_start	sys_end
Alligator Swamp	Louisiana	02/15/2011	50.00	02/01/2011	02/16/2011	02/01/2011	12/30/9999

- **You can query in both `business_time` and `system_time`**
  - Example: What trips were offered on June 20, 2011, as recorded in the database on May 10, 2011?

```
SELECT trip_name, destination FROM TRAVEL FOR BUSINESS_TIME AS OF
'06/20/2011' FOR SYSTEM_TIME AS OF  '2011-05-10';
```

- **Similar `INSERT/UPDATE/DELETE` behavior to ATTs**
  - Rows `inserted`/split/`deleted` as required
- **`UPDATE` and `DELETE` cause automatic insertion into the corresponding STT history table**
- **SELECT will go to STT history as needed to get rows**

How to Define a Bi-temporal Table

```
CREATE TABLE travel(
trip_name CHAR(25) NOT NULL,
destination CHAR(8) NOT NULL,
departure_date DATE NOT NULL,
price DECIMAL(8,2) NOT NULL,
BUS_START DATE NOT NULL ,
BUS_END DATE NOT NULL,
SYS_START TIMESTAMP(12) NOT NULL
          GENERATED ALWAYS AS ROW BEGIN IMPLICITLY HIDDEN,
SYS_END TIMESTAMP(12) NOT NULL
          GENERATED ALWAYS AS ROW END IMPLICITLY HIDDEN,
TX_ID TIMESTAMP(12)
          GENERATED ALWAYS AS TRANSACTION START ID IMPLICITLY HIDDEN,
PERIOD SYSTEM_TIME (SYS_START, SYS_END),
PERIOD BUSINESS_TIME (BUS_START, BUS_END),
PRIMARY KEY (trip_name, BUSINESS_TIME WITHOUT OVERLAPS));

CREATE TABLE travel_history LIKE travel;

ALTER TABLE travel ADD VERSIONING USE HISTORY TABLE travel_history;
```

Application-temporal (ATT) keywords

System-temporal (STT) keywords

Bi-temporal: Query in Both System Time and Business Time

- **What departure dates for Alligator Swamp were available for booking on 03/01/2011, as recorded in the database on 02/01/2011?**
  - Current date – June 1, 2011

```
SELECT  departure_date FROM travel FOR BUSINESS_TIME AS OF
'03/01/2011' FOR SYSTEM_TIME AS OF TIMESTAMP
'2011-02-01-00.00.00.000000' WHERE trip_name = 'Alligator Swamp'
```

05/15/2011

Base table

trip_name	destination	departure_date	price	bus_start	bus_end	sys_start	sys_end
Alligator Swamp	Louisiana	02/15/2011	50.00	02/01/2011	02/16/2011	02/01/2011	12/30/9999
Alligator Swamp	Louisiana	05/15/2011	50.00	02/16/2011	05/16/2011	02/01/2011	12/30/9999
Alligator Swamp	Louisiana	09/15/2011	50.00	05/16/2011	06/01/2011	06/01/2011	12/30/2099
Alligator Swamp	Louisiana	09/15/2011	50.00	09/01/2011	09/16/2011	06/01/2011	12/30/9999

History table

trip_name	destination	departure_date	price	bus_start	bus_end	sys_start	sys_end
Alligator Swamp	Louisiana	10/15/2011	50.00	05/16/2011	10/16/2011	02/01/2011	02/02/2011
Alligator Swamp	Louisiana	09/15/2011	50.00	05/16/2011	09/16/2011	02/02/2011	06/01/2011

Views on Temporal Table

- **Views may be defined on system-period temporal tables (base and `history`), application-period temporal tables, or bi-temporal tables**
- **All syntax (e.g. `FOR PORTION OF, AS OF, FROM…TO`, etc.) is supported for views**
- **Two types of views may be defined for temporal tables**
  - View definition containing `FOR BUSINESS_TIME` or `FOR SYSTEM_TIME`
    - Restricts the view to a period in time

```
CREATE VIEW travel_view AS SELECT * FROM travel FOR
SYSTEM_TIME BETWEEN ’06/30/2011’ AND ‘01/01/2012’;
SELECT * FROM travel_view;
```

    - Restriction: queries against the view can’t also contain `FOR BUSINESS TIME` or `FOR SYSTEM TIME`
    - Would lead to ambiguity or conflicts
  - View definition without `FOR BUSINESS_TIME` or `FOR SYSTEM_TIME`
    - Data from all periods is available to the query

```
CREATE VIEW travel_view AS SELECT * FROM travel;
SELECT * FROM travel_view FOR BUSINESS_TIME AS OF ’01/01/2011’;
```

Special Registers

- **You can set the clock back or forward to a specific time for a given session**
  – No changes required for application!

- **Special registers**
  – CURRENT TEMPORAL BUSINESS_TIME
  – CURRENT TEMPORAL SYSTEM_TIME

- **Setting one or both of these registers allows you to query**
  – Past point in SYSTEM_TIME
  – Past or future point in BUSINESS_TIME

```
DB2 SET CURRENT TEMPORAL SYSTEM_TIME   = CURRENT TIMESTAMP – 1 YEAR
DB2 SET CURRENT TEMPORAL BUSINESS_TIME = '2012-12-31'
```

- **Implicit period specification attached to SQL statements**
  – FOR BUSINESS_TIME AS OF CURRENT TEMPORAL BUSINESS_TIME
  – FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME

Time Travel Tables Summarized

- **Temporal tables enable time travel!**
- **Temporal tables may be**
  - System-period temporal tables (STTs)
    - Managed by DB2
    - DB2 maintains a separate history table
  - Application-period temporal tables (ATTs)
    - Managed by the application
    - Current and historical rows are all in the base table
  - Bi-temporal tables
    - Combine System-period and Application-period temporal tables
- **Can create views on STTs or ATTs for** `SELECT` **or** `UPDATE`
- **Can use special registers to query past or future points in time**
- **Can convert current tables to STTs or ATTs**