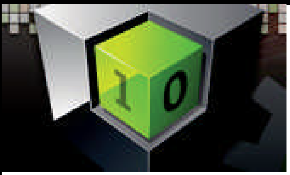
Information Management

IBM



DB2 10 A Technical Overview of New Features Part II

Glen Sheffield and Danny Arnold
IBM Information Management, Worldwide Enablement Team

May 3, 2012

Disclaimer

IBM

Time Travel Query
(Temporal Tables)

Time Travel Query

Lower Costs and Risks with Less Code

- **Optimized for meeting audit and compliance inquires**
- **Point-in-time queries without the burden of changing application**
- **Standardized support for temporal insert, update, and delete operations**
- **Reduce risks, lower costs, and save time**

employees

EmpID	Dept	System_start	System_end
12345	M15	05/31/2000	12/31/9999

employees_history

EmpID	Dept	System_start	System_end
12345	J13	11/15/1995	01/31/1998
12345	M24	01/31/1998	05/31/2000
67890	K25	11/15/1995	03/31/2000

Which department is employee 12345 in?

```
SELECT Dept FROM employees
     WHERE EmpID=12345
```

Which department was employee 12345 in on 12/01/1997?

```
SELECT Dept FROM employees
    FOR SYSTEM_TIME AS OF '12/01/1997'
  WHERE EmpID=12345
```

Create Historical Queries with Less Effort and Reduced Costs

Temporal Tables - What are They? (cont.)

- **Built into DB2**
  – Automatic and transparent

- **Three types of temporal tables**

System-period temporal tables (STTs)

- DB2 stores deleted rows or old versions of updated rows in a history table
- You can query the past state of your data
- Example: employees who have left the company

Application-period temporal tables (ATTs)

- You assign a date range to a data row, indicating the period when the data is valid
- Example: insurance policy effective dates

Bi-temporal Tables

- Combination of STT and ATT
- Keep user-based period information as well as system-based historical information

System-Period Temporal Tables (STTs)

- **Allow you to maintain historical versions of the rows in the table**
- **You create base table and identical history table, including three specific columns**
  - `Row-begin` column: time at which the row data became current
  - `Row-end` column: time at which the row data was no longer current
  - `Transaction start-ID` column: time when execution started for the transaction affecting the row
- **DB2 migrates rows from base table to history table as changes occur, updating the three columns as required**
  - `Row-begin` and `row-end` times will be different for base and history tables
  - `Transaction start-ID` will usually be same as row-begin for base table
- **After table creation, all management of base and history is automatic and transparent**
- **Queries transparently access history table as needed**

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

Application-Period Temporal Tables (ATTs)

- **Allow you to store time-sensitive data**
  - e.g. insurance policy terms on different dates
- **Each row has a pair of `TIMESTAMP` or `DATE` columns, stored by the application**
  - `Begin column`: represents the time at which row data begins to be valid
  - `End column`: represents the time at which row data ceases to be valid
  - e.g. time range during which a price is in effect for an item
- **Data values in these columns are controlled by the user or application**
- **DB2 adds, splits, or deletes rows as needed, automatically and transparently**
- **Can be used to model data in the past, present, and future**
- **Constraints can be automatically enforced to disallow overlapping validity periods**
- **Unlike System-period temporal tables (STTs), no separate history table is required**
  - Current and past data are together in the 'base' table

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 `PERIOD`s 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

Application-Period Temporal Table – Unique Enforcement

- **Manu Wilderness trip has sold out, so we'll add another section departing on 11/2/2011, which is available starting on 10/01/2011 through the end of 2011**
  - Current date = Sept. 1, 2011

```
INSERT INTO travel VALUES (
'Manu Wilderness','Peru','11/02/2011',1500.00,'10/01/2011','01/01/2012')
```

INSERT fails: bus_start and bus_end PERIOD of inserted row cannot overlap bus_start and bus_end times of existing rows (they form a unique primary key with trip_name)

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	06/01/2011
Manu Wilderness	Peru	08/02/2011	1000.00	06/01/2011	07/01/2011
Manu Wilderness	Peru	08/02/2011	1500.00	07/01/2011	01/01/2012

Application-Period Temporal Table – Valid Insert

- **Solution to `INSERT` error on previous page. Change name of new section to 'Manu Wilderness 2' and re-`INSERT`**
  - Current date = Sept. 1, 2011

```
INSERT INTO travel VALUES (
'Manu Wilderness 2','Peru','11/02/2011',1500.00,'10/01/2011','01/01/2012')
```

SUCCESS!

New line inserted in blue

`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	06/01/2011
Manu Wilderness	Peru	08/02/2011	1000.00	06/01/2011	07/01/2011
Manu Wilderness	Peru	08/02/2011	1500.00	07/01/2011	01/01/2012
Manu Wilderness 2	Peru	11/02/2011	1500.00	10/01/2011	01/01/2012

DELETE from an Application-Period Temporal Table

- **Mudslide has wiped out the Manu Wilderness Lodge. Discontinue the Manu Wilderness trips until they rebuild**
  - Current date = Sept. 15, 2011

```
DELETE FROM travel FOR PORTION OF BUSINESS_TIME  FROM '09/15/2011'
TO '12/30/9999' WHERE trip_name LIKE 'Manu Wilderness%'
```

`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	06/01/2011
Manu Wilderness	Peru	08/02/2011	1000.00	06/01/2011	07/01/2011
Manu Wilderness	Peru	08/02/2011	1500.00	07/01/2011	09/15/2011

DB2 has changed `bus_end` column for Manu Wilderness trip to `09/15/2011`

DB2 has `DELETEd` row for Manu Wilderness 2 trip because the entire `PERIOD` for the row was later than `9/15/2011`

Bi-temporal Tables

- **Combine application-period (ATT) and system-period (STT) capabilities**
- **Every row has a pair of `TIMESTAMP`s (`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

Insert Data into a Bi-temporal Table

- Add new trip: Alligator Swamp, departing 3 times in 2011, on `2/15/2011`, `5/15/2011`, and `10/15/2011`
  - Current date = Feb. 1, 2011

```
INSERT INTO TRAVEL VALUES ('Alligator Swamp', 'Louisiana',
'02/15/2011', 50.00, '02/01/2011', '02/16/2011')
```

- Plus 2 more `INSERT` statements for `departure_dates 05/15/2011` and `10/15/2011`

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	10/15/2011	50.00	05/16/2011	10/16/2011	02/01/2011	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

`_tx_id` is hidden

Update Bi-temporal Business Time

- **Change departure date for 3rd section from October 15 to September 15, and update valid period during which trip can be booked (`bus_end`)**
  - Current date = Feb 2, 2011

```
UPDATE travel SET departure_date = '09/15/2011', bus_end = '09/16/2011''
WHERE trip_name = 'Alligator Swamp' and departure_date = '10/15/2011'
```

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	09/16/2011	02/02/2011	12/30/2099

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

Application determined new `bus_end` time;
DB2 sets new `sys_start` value and inserts old row into history table

Delete Portion of Bi-temporal Business Time

- **Alligator Swamp guide quit; remove trip for 3 months while we find a new guide**
  - Current date – June 1, 2011

```
DELETE FROM travel FOR PORTION OF BUSINESS TIME FROM '06/01/2011'
TO '09/1/2011'WHERE trip_name = 'Alligator Swamp'
```

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`

Information Management

Summary
Part II

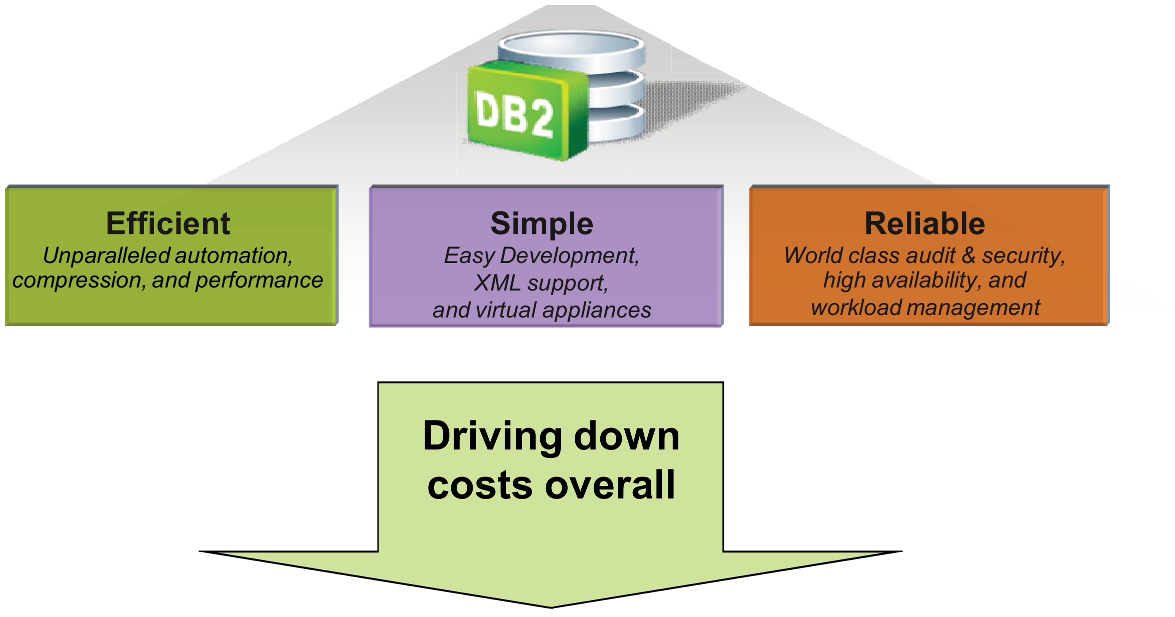
May 3, 2012

Summary

- **Adaptive Compression provides continuous compression advantages as data values change over time without having to rebuild the static table compression dictionary**
  - Combines static table compression with adaptive page level compression
- **DB2 HADR now supports up to 3 standby databases and delayed log apply on an auxiliary standby**
- **Time Travel Query provides Bitemporal table support to enable queries against business time or system time**
  - The capability to query data in the past, present, or future

DB2 10

Building on the Three Tenets of DB2



Efficient
Unparalleled automation, compression, and performance

Simple
Easy Development, XML support, and virtual appliances

Reliable
World class audit & security, high availability, and workload management

Driving down costs overall

DB2 10