The Data That Time Forgot…

Andy Ward – Senior Principal Consultant

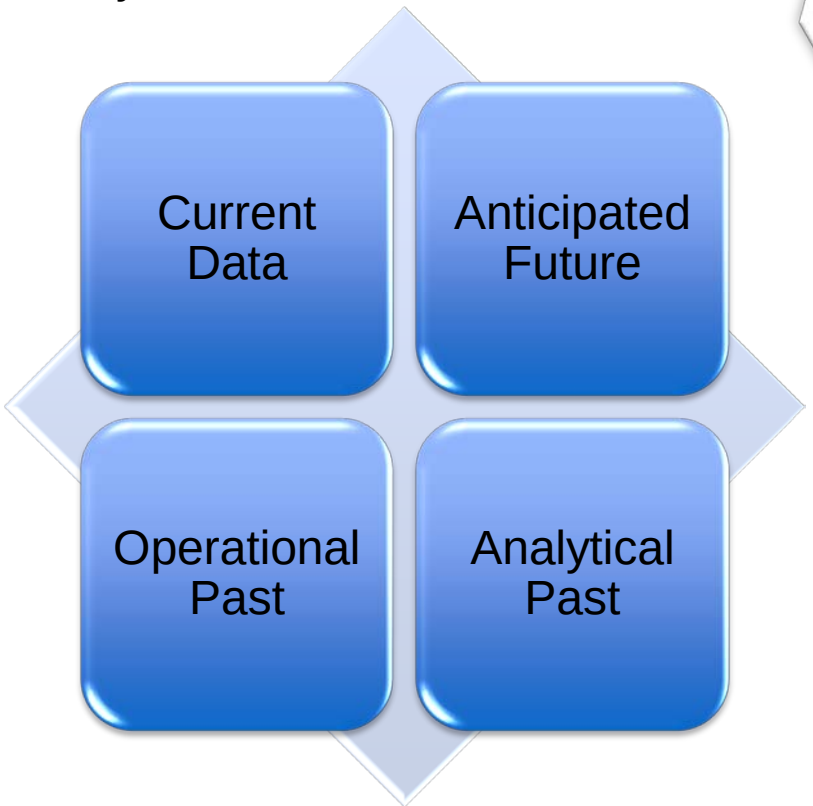
December 2012

Agenda

- Temporal a definition
  - Real world and RDBMS related
- Time Periods
  - System Time
    - Definition and Walkthrough
  - Business Time
    - Definition and Walkthrough
  - Why do we need Temporal Tables?
- Further Temporal components
  - Bi-Temporal Tables
  - Temporal Uniqueness
  - Standards
- A Few Final Thoughts

A definition

- Temporal
  - Of or relating to, or limited by time
  - Lasting for a relatively short time
    - Not really the case for DB2



Time Periods

Types of Time Period…1

- System Time or Period (real time)
  - Tracking when data changes are made to table, when a row is modified or inserted for example (DATE or TIMESTAMP)
  - Managed by DB2
- Business Time or Application Period (valid time)
  - Tracking the effective dates of certain business conditions, interest rate for a period or sum insured for an insurance policy for example (DATE or TIMESTAMP)
  - Application and DB2 controlled

Types of Time Period…2

- Bi-Temporal
  - An object that contains both System and Business time tracking
- Periods are inclusive/exclusive
  - Also known as closed/open or half open
  - The start time is included in the period, the end time is not
  - Except for one SQL construct explained later

A Few Words On TIMESTAMPS

- Prior to DB2 10 the TIMESTAMP precision was fixed at one microsecond (10^{-6}) – still the default
  - i.e. yyyy-mm-dd-hh.mm.ss.nnnnnn
- DB2 10 now supports TIMESTAMP precision up to a picosecond (10^{-12})
  - i.e. yyyy-mm-dd-hh.mm.ss.nnnnnnnnnnnn
  - And everything in between!
    - TIMESTAMP(0) – TIMESTAMP(12)
- Time zone support is also now available
  - An extra two bytes holds the timezone offset
  - Not available for temporal tables though

Why Do We Need Temporal Tables?

- Scenarios
  - Financial Institution (SYSTEM)
    - Auditors may require a report on changes made to a clients records over the last 5 years
  - Travel Agency (BUSINESS)
    - Ensuring elements of a booking tie up (i.e. Flights to London, but car hire in LA for the same period should flag an error)
  - Customer Service (SYSTEM + BUSINESS)
    - A client calls asking why he was recently denied a hire car after an accident. What did the data look like when he called?
- Application reasons
  - Simplify application logic and the need for complex triggers and stored procedures
  - Reduce query complexity

Why Do We Need Temporal Tables…cont'd?

- Standards based technology
  - Consistency and data quality

System Time
aka System Period

System Time

- Adding System Time to a table
  - Requires 2 columns
    - SYSTIME_FROM TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN
    - SYSTIME_TO TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END
      - Must be WITHOUT TIMEZONE and are not updatable
  - And a PERIOD definition
    - PERIOD SYSTEM_TIME(SYSTIME_FROM, SYSTIME_TO)
    - This PERIOD cannot be DROPped
- This in itself is not particularly useful, unless you are only interested in the very last action

Versioning

- Only applicable to System Time
- Requires an additional TIMESTAMP column
  - TRANS_TIME TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID
    - Must be WITHOUT TIMEZONE and is not updatable
    - Interestingly this can be NULLable
- Requires an additional history table
- Requires a connection between the base table and the history table
  - Create the base table (or alter an existing table)
  - Create the history table
  - Alter the base table to include versioning

History Table

- A copy of the base table
  - Excluding temporal period definitions and GENERATED… syntax for relevant temporal columns
  - Simplest way to create this object is via the CREATE TABLE x LIKE y syntax
- Only used for System Time
  - Automatically maintained by DB2 when data is altered, keeps a temporal SYSTEM orientated audit trail

Versioning...2

Create the Base Table

```
CREATE TABLE insurance
(policy_id CHAR(4) NOT NULL,
 registration_number CHAR(6) NOT NULL,
 sum_insured INT NOT NULL,
 hire_car CHAR(1) NOT NULL,
 sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 create_id TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID,
PERIOD SYSTEM_TIME(sys_start,sys_end));
```

Working with System Time...1

- Lets start simply inserting into the insurance base table at 11:30 on 01/01/11
- INSERT INTO INSURANCE (policy_id, registration_number, sum_insured, hire_car) VALUES (1111, ABC123, 10000, Y);

Current Table (INSERT)

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	10000	Y	01/01/11 11:30	12/31/9999 24:00	01/01/11 11:30

History Table (NO ACTIVITY)

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id

Working with System Time...2

- Now lets assume the row is updated on 03/02/11 at 12:48
- UPDATE INSURANCE SET SUM_INSURED = 20000 WHERE POLICY NUMBER = ‘1111’

Current Table (UPDATE)

U

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	20000	Y	03/02/11 12:48	12/31/9999 24:00	03/02/11 12:48

History Table (INSERT)

I

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	10000	Y	01/01/11 11:30	03/02/11 12:48	01/01/11 11:30

Working with System Time...3

- And updated again on 05/03/11 at 15:45
- UPDATE INSURANCE SET CAR_HIRE = ‘N’ WHERE POLICY NUMBER = ‘1111’

Current Table (UPDATE)

	Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
U	1111	ABC123	20000	N	05/03/11 15:45	12/31/9999 24:00	05/03/11 15:45

History Table (INSERT)

	Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
I	1111	ABC123	10000	Y	01/01/11 11:30	03/02/11 12:48	01/01/11 11:30
	1111	ABC123	20000	Y	03/02/11 12:48	05/03/11 15:45	03/02/11 12:48

Working With System Time...4

- SELECTing data
  - A normal SELECT takes data from the current table as per usual
  - New FROM clauses
    - Transparently query both the CURRENT and HISTORY tables as required
    - FOR SYSTEM TIME AS OF ...
      - Query data at a certain point in time
    - FOR SYSTEM TIME FROM ... TO ...
      - Query from a certain time to a certain time
      - Inclusive/Exclusive
    - FOR SYSTEM TIME BETWEEN ... AND ...
      - Query data between a range
      - Inclusive/Inclusive

Working With System Time...5

— SELECT FROM INSURANCE FOR SYSTEM TIME AS OF 14/02/11

Current Table

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	20000	N	05/03/11 15:45	12/31/9999 24:00	05/03/11 15:45

History Table

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	10000	Y	01/01/11 11:30	03/02/11 12:48	01/01/11 11:30
1111	ABC12	2000	Y	03/02/11 12:48	05/03/11 15:45	03/02/11 12:48

Result Set:

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	20000	Y	03/02/11 12:48	05/03/11 15:45	03/02/11 12:48

Working With System Time...6

—SELECT...SYSTEM_TIME BETWEEN 01/01/11 AND 05/03/11

Current Table

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	20000	N	05/03/11 15:45	12/31/9999 24:00	05/03/11 15:45

History Table

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	10000	Y	01/01/11 11:30	03/02/11 12:48	01/01/11 11:30
1111	ABC12	2000	Y	03/02/11 12:48	05/03/11 15:45	03/02/11 12:48

Result Set

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	10000	Y	01/01/11 11:30	03/02/11 12:48	01/01/11 11:30
1111	ABC123	20000	Y	03/02/11 12:48	05/03/11 15:45	03/02/11 12:48

H
H
C

Working With System Time...7

- DELETEing the row at 16:42 on 12/03/11
- DELETE FROM INSURANCE WHERE POLICY_ID = 1111

Current Table (DELETE)

D

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id

History Table (INSERT)

I

Pol_id	Reg_#	Sum	Hire	Sys_Start	Sys_End	Create_id
1111	ABC123	10000	Y	01/01/11 11:30	03/02/11 12:48	01/01/11 11:30
1111	ABC123	20000	Y	03/02/11 12:48	05/03/11 15:45	03/02/11 12:48
1111	ABC12	2000	N	05/03/11 15:45	12/03/11 16:42	05/03/11 15:45

A Few things to Consider...

- No utility operation is allowed against a system period temporal table that will delete data
  - LOAD REPLACE, REORG DISCARD etc.
- No clones are allowed
- Current and history table must be the only tables in their tablespaces
- Current and History tables must be recovered as a set (unless VERIFYSET NO keyword is used)
- Schema attributes cannot be ALTERed for current or history tables
- History tables and their relevant TSs and DBs cannot be explicitly dropped
  - They are implicitly dropped when the base table is dropped

Business Time
aka Application Period or Valid Time

Business Time

- A period managed by an application or a user, and in some cases by DB2
- No history table is maintained or queried
  - All business_time data is contained in the current table (unless bi-temporal tables and queries are used)
- Periods of business time can be defined as UNIQUE
  - Covered later
- Requires 2 columns
  - BUSTIME_FROM TIMESTAMP(6) NOT NULL
  - BUSTIME_TO TIMESTAMP(6) NOT NULL
    - Can also be DATE type columns
    - Must be WITHOUT TIMEZONE and are not updatable (via non-temporal SQL, -545)
- And a PERIOD definition
  - PERIOD BUSINESS_TIME(BUSTIME_FROM, BUSTIME_TO)

Creating a Business Time Temporal Table

Create the Business Time Table

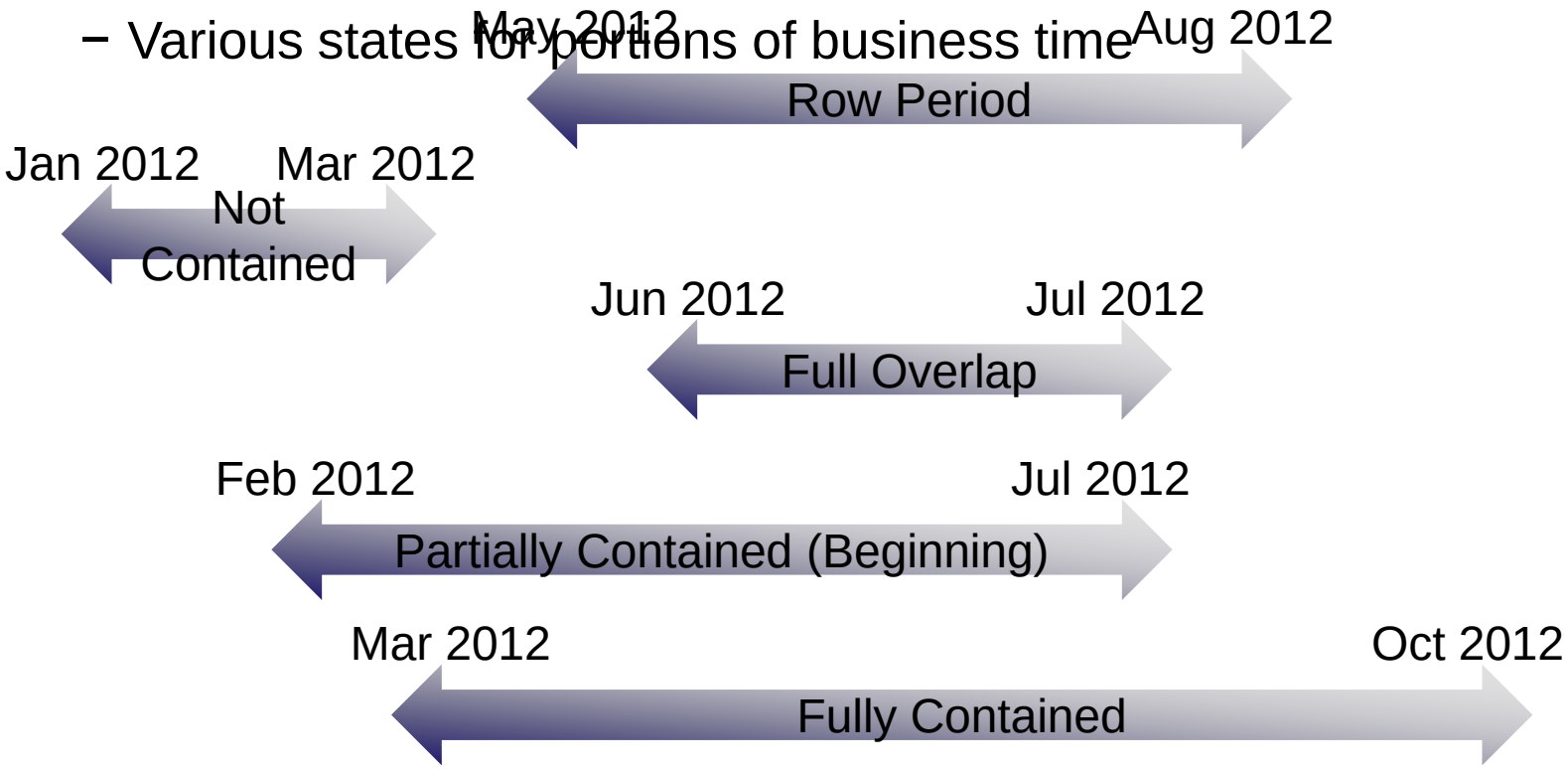
```
CREATE TABLE insurance
(policy_id CHAR(4) NOT NULL,
 registration_number CHAR(6) NOT NULL,
 sum_insured INT NOT NULL,
 hire_car CHAR(1) NOT NULL,
 bus_start TIMESTAMP(6) NOT NULL,
 bus_end TIMESTAMP(6) NOT NULL,
 PERIOD BUSINESS_TIME(bus_start,bus_end),
 PRIMARY KEY(policy_id, BUSINESS TIME WITHOUT OVERLAPS)
);
```

Working with Business Time...1

- SELECTing data (as per system time)
  - A normal SELECT takes data from the current table as per normal
  - New FROM clauses
    - FOR BUSINESS_TIME AS OF ...
      - Query data at a certain point in time
    - FOR BUSINESS_TIME FROM ... TO ...
      - Query from a certain time to a certain time
      - Inclusive/Exclusive
    - FOR BUSINESS_TIME BETWEEN ... AND ...
      - Query data between a range
      - Inclusive/Inclusive
- INSERTing data
  - Provide the values for the START and END periods

Working with Business Time...2

- UPDATEing and DELETEing data
  - A new clause has been added FOR PORTION OF BUSINESS_TIME FROM value1 TO value2 (SELECT FROM UPDATE/DELETE not supported)
    - Inclusive/Exclusive
  - Various states for portions of business time

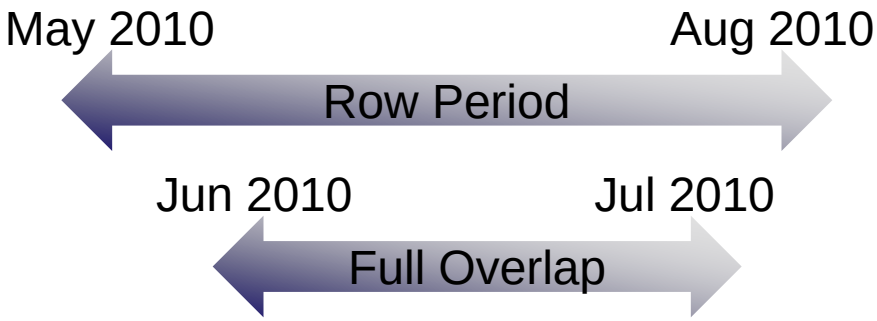


Working with Business Time...3

- Not contained
  - No action is taken
- Fully contained
  - The row is updated or deleted
- Fully overlapping
  - The row is updated or deleted
    - If updated, BUS_START is set to value1, BUS_END is set to value2
    - If deleted the portion of business time is removed
  - Two new rows are inserted one contains the original start date and value1 as the end date, one contains the original end date and value2 as the start date
    - Consider any INSERT triggers you have defined

Working with Business Time...4

— Fully overlapping data period




```
UPDATE BUS_INSURANCE FOR PORTION OF BUSINESS_TIME
FROM 'JUNE 2010'
TO 'JULY 2010'
SET HIRE_CAR = 'N'
WHERE POLICY_ID = '1111';
```

Working with Business Time...5

```
UPDATE BUS_INSURANCE FOR PORTION OF BUSINESS_TIME
FROM 'JUNE 2010'
TO 'JULY 2010'
SET HIRE_CAR = 'N'
WHERE POLICY_ID = '1111';
```

Original Data

Pol_id	Reg_#	Sum	Hire	Bus_Start	Bus_End
1111	ABC123	20000	Y	MAY 2010	AUG 2010

New Data (CURRENT TABLE)

U
I
I

Pol_id	Reg_#	Sum	Hire	Bus_Start	Bus_End
1111	ABC123	20000	N	JUN 2010	JUL 2010
1111	ABC123	20000	Y	MAY 2010	JUN2010
1111	ABC123	20000	Y	JUL 2010	AUG 2010

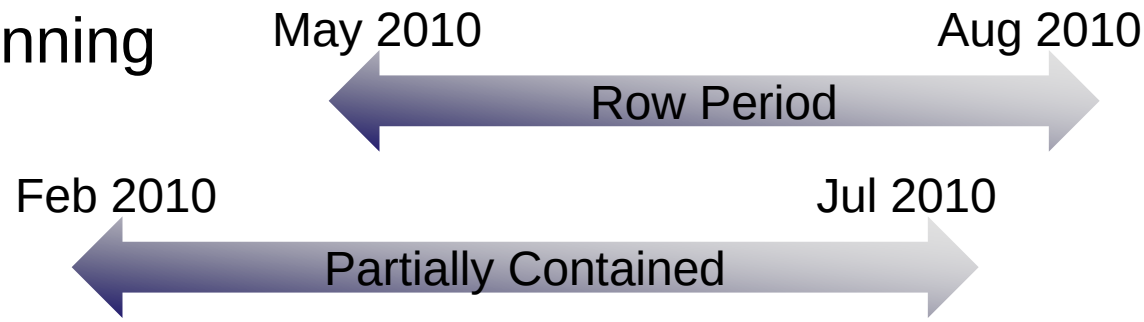
ca technologies

Working with Business Time...6

- Partially contained, overlapping beginning of the period
  - The row is updated or deleted
    - BUS_START is set to original value, BUS_END is set to value2
  - A new row is inserted
    - BUS_START is set to value2, BUS_END holds original value
- Partially contained, overlapping the end of the period
  - A new row is inserted
    - BUS_START holds original value, BUS_END is set to value1
  - The row is updated or deleted
    - BUS_START is set to value1, BUS_END holds original value

Working With Business Time...7

— Partially contained data period, overlapping at the beginning




```
UPDATE BUS_INSURANCE FOR PORTION OF BUSINESS_TIME
FROM 'FEBRUARY 2010'
TO 'JULY 2010'
SET HIRE_CAR = 'N'
WHERE POLICY_ID = '1111';
```

Working with Business Time...8

```
UPDATE BUS_INSURANCE FOR PORTION OF BUSINESS_TIME
FROM 'FEBRUARY 2010'
TO 'JULY 2010'
SET HIRE_CAR = 'N'
WHERE POLICY_ID = '1111';
```

Original Data

Pol_id	Reg_#	Sum	Hire	Bus_Start	Bus_End
1111	ABC123	20000	Y	MAY 2010	AUG 2010

New Data (CURRENT TABLE)

U

I

Pol_id	Reg_#	Sum	Hire	Bus_Start	Bus_End
1111	ABC123	20000	N	MAY 2010	JUL 2010
1111	ABC123	20000	Y	JUL 2010	AUG 2010

ca technologies

Business Time Temporal Uniqueness

- The PRIMARY KEY clause now provides the ability to ensure business periods are unique
  - BUSINESS_TIME WITHOUT OVERLAPS
    - Adds the business period start and end columns to the index in ASCending order to enforce there are no overlaps in time
  - The Business Time defining columns are not explicitly included in the definition
- Currently no support for ALTER INDEX to set temporal uniqueness
  - Object will have to be dropped and recreated

Bi-Temporal Tables

Bi-Temporal Tables

- It is possible to define tables that contain both PERIOD SYSTEM_TIME and BUSINESS_TIME clauses
  - Providing the best of both worlds
- And use queries like this...
  - SELECT * FROM INSURANCE FOR BUSINESS_TIME AS OF ‘1/11/2010’ FOR SYSTEM_TIME AS OF ‘15/11/2010’ WHERE POLICY_ID=‘1111’;
  - Showing what the business data looked like at a specified system time
    - A BUSINESS_TIME only view may show different information as it only accesses the base table

A Simple Example Using the….

Disclaimer: This is based purely on winners of the Tour de France according to the UCI. It is not commenting one way or the other as to whether Lance Armstrong was or was not involved in doping.

Creating a Bi-Temporal Table

```
CREATE TABLE BI_CYCLING
 (YEAR CHAR(4) NOT NULL,
  WINNER CHAR(30) NOT NULL,
  SYS_START TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL
```

Create the History Table

```
CREATE TABLE BI_CYCLING_HISTORY LIKE BI_CYCLING
```

```
  BUS_END DATE NOT NULL,
  PERIOD BUSINESS_TIME(BUS_START,BUS_END),
  PERIOD SYSTEM_TIME(SYS_START,SYS_END));
```

Working With Bi-Temporal Tables...2

—A subset of the data from the base table as of NOV 2012

Year	Winner	Sys_Start	Sys_End	Bus_Start	Bus_End
1997	Jan Ullrich	JUL 1997	DEC		998
1998	Marco Pantani				1999
1999	Vac		9999	JUL 1999	JUL 2000

So what happened….
Somebody must have won it?

…ber Inclusive/Exclusive)

Year		Sys_Start	Sys_End	Bus_Start	Bus_End
1999	Vacated	OCT 2012	DEC 2099	JUL 1999	JUL 2000

Working With Bi-Temporal Tables...2

— The records were updated in OCT 2012 after Lance Armstrong was stripped of his titles

```
UPDATE BI_CYCLING FOR PORTION OF BUSINESS_TIME
FROM 'JUL 1999' TO 'JUL 2000'
SET WINNER = 'Vacated'
WHERE YEAR = '1999';
```

Current Table

Year	Winner	Sys_Start	Sys_End	Bus_Start	Bus_End
1997	Jan Ullrich	JUL 1997	DEC 9999	JUL 1997	JUL 1998
1998	Marco Pantani	JUL 1998	DEC 9999	JUL 1998	JUL 1999

U

Year	Winner	Sys_Start	Sys_End	Bus_Start	Bus_End
1999	Lance Armstrong	JUL 1999	OCT 2012	JUL 1999	JUL 2000

I

Working With Bi-Temporal Tables...3

— The following bi-temporal query shows us that Lance Armstrong was indeed the winner…until OCT 2012

```
SELECT * FROM BI_CYCLING
FOR BUSINESS_TIME AS OF 'JUL 1999'
FOR SYSTEM_TIME AS OF 'SEP 2012';
```

Current Table

Year	Winner	Sys_Start	Sys_End	Bus_Start	Bus_End
1997	Jan Ullrich	JUL 1997	DEC 9999	JUL 1997	JUL 1998
1998	Marco Pantani	JUL 1998	DEC 9999	JUL 1998	JUL 1999

Year	Winner	Sys_Start	Sys_End	Bus_Start	Bus_End
1999	Lance Armstrong	JUL 1999	OCT 2012	JUL 1999	JUL 2000

In Closing...

A Couple of Gotchas

- No COMMIT between base table update and history select
  - History is not updated
  - Consider this when operating in the same UOW
- Views can reference temporal tables
  - But…Temporal SQL extensions cannot be used against a view (-20524)

A Quick Word on Performance

- The DB2 10 Performance Topics Redbook documents the performance tests run for temporal tables
  - System Period tables were found to outperform user defined triggers by around 35% for SELECT processing and 15% for DELETE processing (CPU)
  - Business Period tables were found to outperform user defined stored procedure by around 62% when row-splitting was required (CPU)
- Ease of development
  - No triggers, stored procedures, or code over and above for standard SQL execution are required

ANSI and SQL Standards

- First proposed as a standard back in 1993
  - An ISO project responsible for temporal support was cancelled back in 2001 due to disagreements
- Temporal extensions to ANSI/ISO delivered fully in SQL:2011
  - DB210 formally adheres to SQL:2003
- Despite this late inclusion many vendors have included temporal abilities in their DBMS for a number of years
  - Oracle 10g (2006), Teradata (2010), DB2 zOS (2010)

Final Thoughts

- How likely are people to undo their existing application logic?
  - May well be high on the list for new applications though
- Does an inclusive/exclusive approach suit your needs?
- Consider the impact of non-temporal updates and deletes for business time columns – these are valid
- Consider recoveries carefully
  - VERIFYSET NO (DB2 10) gives you the option to shoot yourself in the foot
- Look out for conflicting (and incorrect) documentation, PDFs, Redbooks.
  - Not all of what I found when researching this topic worked correctly
- What does the future hold?
  - Multi-Temporal tables – (i.e. System Time, Business Time and Decision Time)?

Summary

- Why do we need temporal tables?
- Time periods
  - System time
  - Business time
- Bi-Temporal tables
- A few things to think about