



Activating New Year's online sale catalog at the stroke of midnight.

DB2 Temporal

Status: GA DB2z V10 Oct. 2010
As alpha in DB2 luw

Customer's equity position on 2/20/2009 reported on customer's 2/28/2009 statement.

ACADEMY AWARDS
NOMINATIONS
Original Screenplay
Sound
Sound Editing (WIN)
Original Song:



1985 Production \$19M: Receipts \$381M

Price of returned merchandise on day of purchase

Customer's equity position on 2/20/2009 after all known corrections.



DB2 Temporal Team



Customer complaint about a seven month old bill



Need to enable compliance for packaged app.

Names of out-of-network specialists who billed for service to the insured client between 5/1/2006 and 10/1/2007.



What DB2 brings to your Temporal Apps?

- Cost savings via up to 10x¹ Application Simplification
 - Complexity
 - Lines of code
- Up to 10x¹ Reduction in Time to Deployment
- Better ROI from consultant engagement

¹: YMMV

What do you get for Compliance?

- Ability to move it from application to database layer

What is the magic sauce?

- Powerful, novel, yet intuitive SQL (in ISO SQL 2011)
- End users deploy without IT Staff intervention
- Query past/future data with current queries
- Same schema name, simply add novel temporal clause
- Response time for current DML/queries preserved
- Response time for past/future queries comparable

bankdata

"We are really thrilled about "Temporal Data" feature – this feature has the potential to **significantly reduce overheads**.

We have estimated that **80% of our existing temporal applications** could have used "the DB2 10 temporal features" instead of application code - this feature will **drastically save developer time, testing time** – and even more importantly make applications easier to understand so **improve business efficiency and effectiveness**"

Frank Petersen -System Programmer

The new temporal functionality in DB2 10 for z/OS will allow us to **drastically simplify** our date-related queries.

In addition, **we'll be able to reduce our storage costs** by using cheaper storage for inactive rows and **reduce our processing cost** by having DB2 handle data movement more efficiently than the custom code we've written to do the same work in the past

Large Insurance Company

Summary of Proposal

- Business Time (Effective Dates, Valid Time, From/To-dates)
 - Every row has a pair of timestamps (dates) set by App
 - Begin time: when the business deems the row valid
 - End Time: when the business deems row validity ends
 - Query at current, any prior, or future point/period in business time
- System Time (Assertion Dates, Knowledge Dates, Transaction Time, Audit Time, In/Out-dates)
 - Every row has another pair of timestamps set by DBMS
 - Begin time: when the row was inserted in the DBMS
 - End Time: when the row was modified/deleted
 - Modified rows begin time is the modification time
 - Query at current or any prior point/period in system time
- Bi-temporal (Two dimensional history, Two dimensional milestoneing)
 - Inclusion of both System Time and Business Time in row

What we heard from customers and how that was translated that into a product feature?

- From CIO of leading property lines insurance company: “We work in an environment of constantly changing state and local laws, we need to support the evolution of a policy”
 - Translated into business time support
- Lead Application developer, Insurance: “A lot of time is spent validating that the date predicates have been correctly coded by the application”
 - Translated into FOR PORTION OF clause in BUSINESS TIME
- VP, Investment Banking: “We have complicated procedures to ensure that our customer has exactly one position effective at any point in time, and our code still lets some bad data through”
 - Translated to PRIMARY KEY BUSINESS TIME WITHOUT OVERLAPS

Business Time Use Case: Health Insurance

| Date | Kind of Event | Event |
|-------------|---------------|---|
| 1 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 6/11/2004 | Future | Company schedules co-pay increase to \$15 starting 1/1/2005 |
| 3 6/1/2005 | Present | Employee C054 cancels Policy P667 as of today |
| 4 9/1/2005 | Past | Correct error by retroactively updating policy to POS from 9/1/2004 to 3/1/2005 |

DDL: *CREATE TABLE Policy*
(Empl VARCHAR(4) NOT NULL,
Type VARCHAR(4),
Plcy VARCHAR(4) NOT NULL,
Copay VARCHAR(4),
Eff-Beg DATE,
Eff-End DATE,
PERIOD BUSINESS_TIME (Eff-Beg, Eff-End),
PRIMARY KEY (Empl,Plcy BUSINESS_TIME WITHOUT OVERLAPS)
);

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|---------|---------|
|------|------|------|-------|---------|---------|

Business Time Use Case: Health Insurance - DML

| Date | Kind of Event | Event |
|-------------|---------------|---|
| 1 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|---------|---------|
|------|------|------|-------|---------|---------|



INSERT INTO POLICY

VALUES ('C054','HMO', 'P667', '\$10', '1/1/2004', '12/31/9999')

1/1/2004



| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 |

Business Time Use Case: Health Insurance - DML

| | Date | Kind of Event | Event |
|---|-----------|---------------|---|
| 1 | 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 | 6/11/2004 | Future | Company schedules co-pay increase to \$15 starting 1/1/2005 |

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 |



UPDATE POLICY FOR PORTION OF BUSINESS_TIME
FROM DATE '1/1/2005' TO '12/31/9999'
SET Copay='\$15'
WHERE Plcy='P667';



| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 12/31/9999 |

Business Time Use Case: Health Insurance - DML

| Date | Kind of Event | Event |
|-------------|---------------|---|
| 1 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 6/11/2004 | Future | Company schedules co-pay increase to \$15 starting 1/1/2005 |
| 3 6/1/2005 | Present | Employee C054 cancels Policy P667 as of today |

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 12/31/9999 |



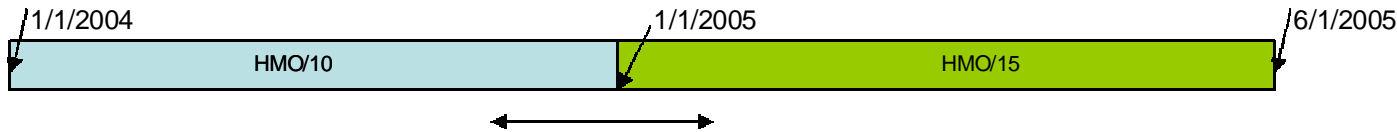
DELETE POLICY FOR PORTION OF BUSINESS_TIME
FROM CURRENT DATE TO '12/31/9999'
WHERE Empl='C054' AND Plcy='P667';



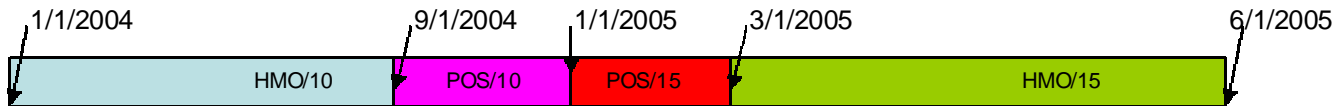
| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|----------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 6/1/2005 |

| Date | Kind of Event | Event |
|-------------|---------------|---|
| 1 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 6/11/2004 | Future | Company schedules co-pay increase to \$15 starting 1/1/2005 |
| 3 6/1/2005 | Present | Employee C054 cancels Policy P667 as of today |
| 4 9/1/2005 | Past | Correct error by retroactively updating policy to POS from 9/1/2004 to 3/1/2005 |

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|----------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 6/1/2005 |

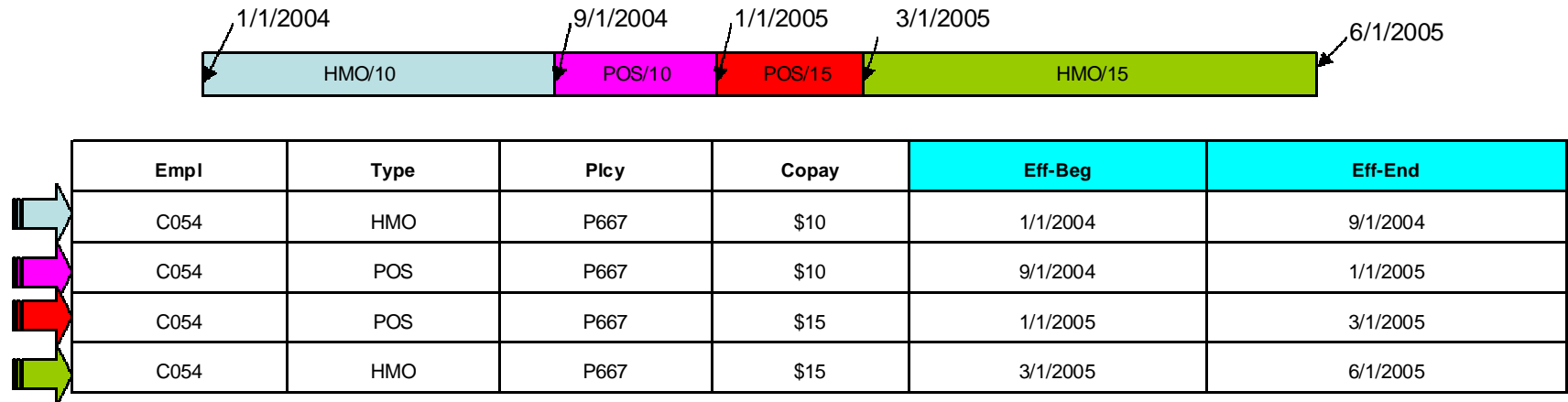


UPDATE POLICY FOR PORTION OF BUSINESS_TIME
 FROM '9/1/2004' TO '3/1/2005'
 SET TYPE='POS'
 WHERE Empl='C054' AND Plcy='P667';



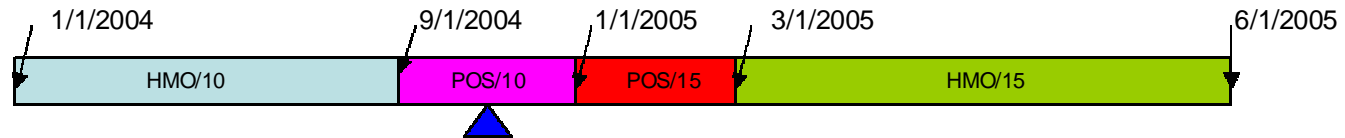
| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|----------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 9/1/2004 |
| C054 | POS | P667 | \$10 | 9/1/2004 | 1/1/2005 |
| C054 | POS | P667 | \$15 | 1/1/2005 | 3/1/2005 |
| C054 | HMO | P667 | \$15 | 3/1/2005 | 6/1/2005 |

KEY WITHOUT OVERLAPS



- Business Constraint
 - Employer covers employee by only one health care policy at any time
 - (Empl, Plcy) unique at any point in business time
 - Conventional notion of DBMS keys insufficient
- DB2 supports novel notion of such “business keys”
 - Specification: on DDL
 - Enforcement: via novel efficient internal structure

| Date | Kind of Event | Event |
|-------------|---------------|---|
| 1 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 6/11/2004 | Future | Company schedules co-pay increase to \$15 starting 1/1/2005 |
| 3 6/1/2005 | Present | Employee C054 cancels Policy P667 as of today |
| 4 9/1/2005 | Past | Correct error by retroactively updating policy to POS from 9/1/2004 to 3/1/2005 |



| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|----------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 9/1/2004 |
| C054 | POS | P667 | \$10 | 9/1/2004 | 1/1/2005 |
| C054 | POS | P667 | \$15 | 1/1/2005 | 3/1/2005 |
| C054 | HMO | P667 | \$15 | 3/1/2005 | 6/1/2005 |

Business Question: On 9/15/2005 client calls & complains: client saw an out-of-network specialist on 11/1/2004, our claims dept. denied client's claim for this visit on 11/15/2004. Client demands reimbursement.

SELECT * FROM POLICY FOR BUSINESS_TIME AS OF DATE '11/1/2004'
WHERE Empl='C054' AND PLCY='P667';

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End |
|------|------|------|-------|----------|----------|
| C054 | POS | P667 | \$10 | 9/1/2004 | 1/1/2005 |

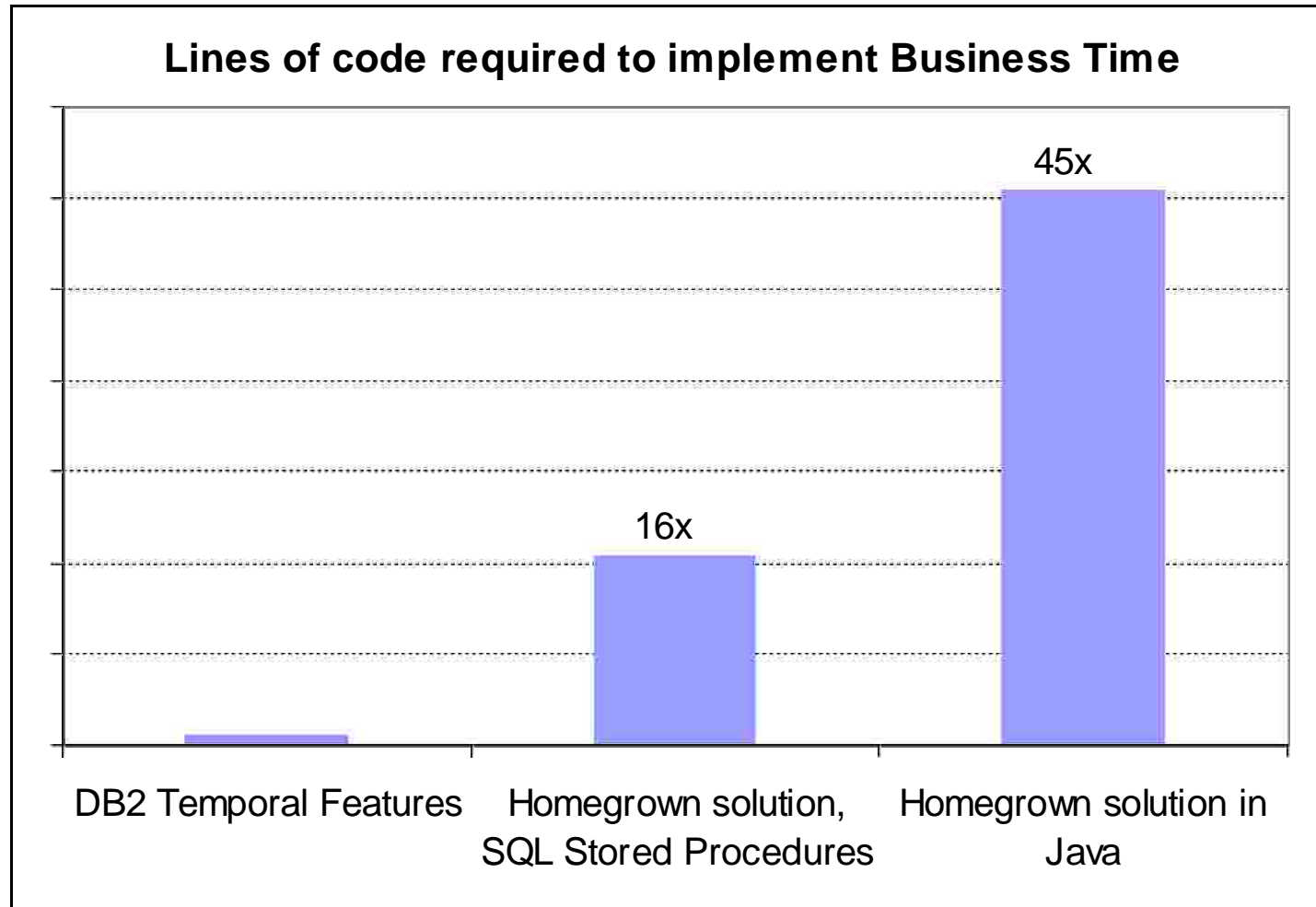
Business Answer: On 11/1/2004 client had POS coverage and is eligible to be reimbursed for out-of-network specialist consultation.

Business Question that cannot be answered: Did our claims department make an error denying the client's claim on 11/15/2004?

Recap: DB2 and Business Time

- Application controls business time stamps
- “AS OF”, “FROM/TO”, “BETWEEN” clauses for
 - Select
- DB2 translates AS OF/FROM-TO/BETWEEN times into predicates on business time column pair
- Novel “FOR PORTION OF FROM/TO” clause for
 - Insert, Update, Delete and Merge
- Novel but appropriate splitting of rows due to “FOR PORTION OF” updates and deletes
- Novel KEY ON BUSINESS_TIME WITHOUT OVERLAPS assists enforcement of business integrity

Reducing Development & Maintenance Cost



Using DB2 Temporal Tables, business time logic can be implemented with 16x or 45x fewer lines of code than using SQL stored procedures or Java, respectively, to implement equivalent logic. (IBM internal study)

Application-period Temporal Tables (Business Time)

- How much application development/maintenance cost can be saved?

| Metric | SQL implementation | JDBC implementation | DB2 Business Time |
|--|---|--|---|
| Update functionality | One SQL stored procedure for each table à 1 CALL stmt | JDBC code for each table à 1 method call | Provided by DB2 à 1 UPDATE stmt |
| Delete functionality | One SQL stored procedure for each table à 1 CALL stmt | JDBC code for each table à 1 method call | Provided by DB2 à 1 DELETE stmt |
| Overlap detection | 2 user-defined SQL triggers for each table | 2 user-defined SQL triggers for each table | Part of PK index for each table |
| Total number of lines of code to provide functionality | 210 | 610 | 13 |
| Total number of SQL statements | 27 | 24 | 3 |
| Time to develop and test | 5 weeks | 6 weeks | < 1 hour |

Application-period Temporal Table

- What is performance gain of using ATT over a homegrown (stored procedure-based) implementation?

New ATT functionality 20% to 80% faster than stored procedure-based solution.

What we heard from customers and how that was translated that into a product feature?

- From CTO of Money Center Bank: 40% of my time is spent on dealing with regulatory compliance
 - Translated into support of system time
- Insurance company CIO: We have packaged apps that we need to enable for compliance through history generation of data base changes made without changing the packaged app.
 - Translated into allowing normal SQL to work against tables with system time, and two table approach
- From the lead technologist of top Canadian bank: “We came to your lab 20 years ago and gave this requirement, where have you been all these years?”
 - Translated into
 - allowing existing tables to be altered into temporal tables and existing history tables to be made into DB2 history tables
 - allowing normal SQL to work against tables with business time
- VP at leading Wall St investment bank – purging data from history is not a strict time based process, it is complex and intricate
 - Translated into allowing SQL based deletion of rows from history table (under special authority)

System Time Use Case Employee Table

| Step | Date | Activity |
|------|-----------|----------------------------|
| 1 | 6/15/2007 | New Employee Hired |
| 2 | 6/15/2008 | Employee Gets Salary Raise |
| 3 | 9/15/2008 | Employee quits |

DML CREATE TABLE EMPDB
(Empname VARCHAR (40),
Salary INTEGER
);

| Empname | Salary |
|---------|--------|
| | |

Step 1 INSERT INTO EMPDB
VALUES ('John Smith', '75000')

| Empname | Salary |
|------------|--------|
| John Smith | 75,000 |

Step 2 UPDATE EMPDB
SET Salary=Salary+5000
WHERE Empname='John Smith'

| Empname | Salary |
|------------|--------|
| John Smith | 80,000 |

Step 3 DELETE FROM EMPDB
WHERE Empname = 'John Smith'

| Empname | Salary |
|---------|--------|
| | |

Significant part of compliance requirement may met by preserving rows updated/deleted.

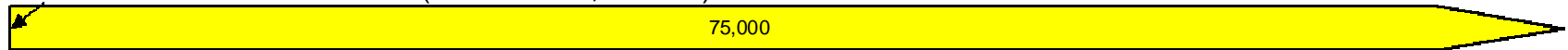
- | Step | Date | Activity |
|------|-----------|----------------------------|
| 1 | 6/15/2007 | New Employee Hired |
| 2 | 6/15/2008 | Employee Gets Salary Raise |
| 3 | 9/15/2008 | Employee quits |

```
CREATE TABLE EMPDB
(Empname VARCHAR (40),
Salary INTEGER,
SysTmSta TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
SysTmEnd TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END, ...
PERIOD SYSTEM_TIME (SysTmSta, SysTmEnd));
```

| Empname | Salary | SysTmSta | SysTmEnd |
|---------|--------|----------|----------|
|---------|--------|----------|----------|

**INSERT INTO EMPDB FOR PORTION OF SYSTEM_TIME
FROM TRANSACTION TIME TO MAXVALUE
VALUES ('John Smith', '75000')**

6/15/2007



| Empname | Salary | SysTmSta | SysTmEnd |
|------------|--------|-----------|------------|
| John Smith | 75000 | 6/15/2007 | 12/31/9999 |

**UPDATE EMPDB FOR PORTION OF SYSTEM_TIME
FROM TRANSACTION TIME TO MAXVALUE
SET Salary=Salary+5000 WHERE Empname='John Smith'**

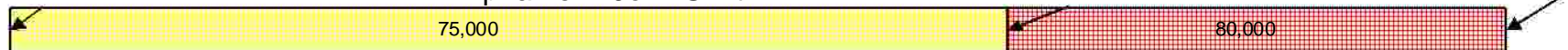
6/15/2007



| Empname | Salary | SysTmSta | SysTmEnd |
|------------|--------|-----------|------------|
| John Smith | 75000 | 6/15/2007 | 6/15/2008 |
| John Smith | 80000 | 6/15/2008 | 12/31/9999 |

**DELETE FROM EMPDB FOR PORTION OF SYSTEM_TIME
FROM TRANSACTION TIME TO MAXVALUE
WHERE Empname = 'John Smith'**

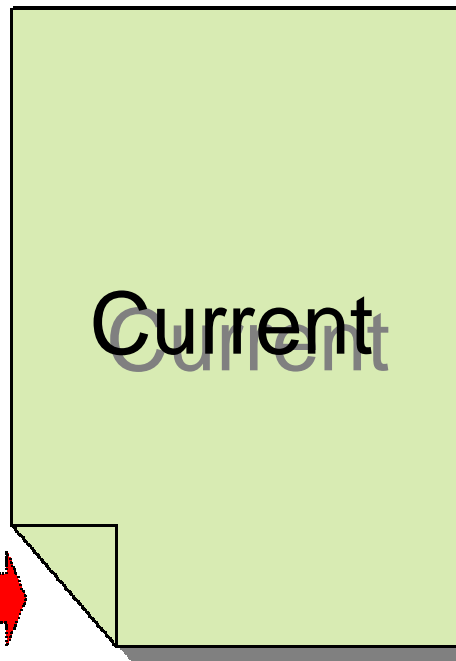
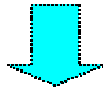
6/15/2007



| Empname | Salary | SysTmSta | SysTmEnd |
|------------|--------|-----------|-----------|
| John Smith | 75000 | 6/15/2007 | 6/15/2008 |
| John Smith | 80000 | 6/15/2008 | 9/15/2008 |

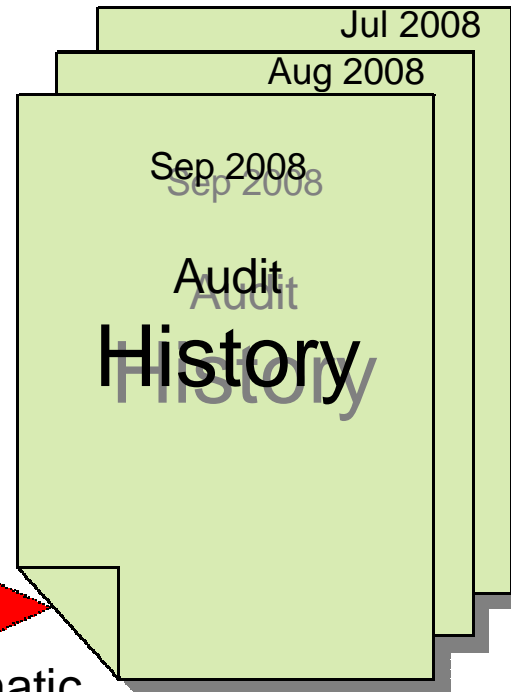
Current and History

Current SQL Application



Auditing SQL Application
Using ASOF

History
Generation



Transparent/automatic
Access to satisfy ASOF
Queries

```
CREATE TABLE EMPDB
(Empname VARCHAR (40),
Salary INTEGER,
SysTmSta TS(12) NN GENERATED ALWAYS AS ROW BEGIN,
SysTmEnd TS(12) NN GENERATED ALWAYS AS ROW END, ..
PERIOD SYSTEM_TIME (SysTmSta, SysTmEnd));
```

```
CREATE TABLE EMPHIST
(Empname VARCHAR (40),
Salary INTEGER,
SysTmSta TS(12)
SysTmEnd TS(12) .. )
;
```

| Step | Date | Activity |
|------|-----------|----------------------------|
| 1 | 6/15/2007 | New Employee Hired |
| 2 | 6/15/2008 | Employee Gets Salary Raise |
| 3 | 9/15/2008 | Employee quits |

```
ALTER TABLE EMPDB ADD VERSIONING USE HISTORY TABLE EMPHIST
```

| Empname | Salary | SysTmSta | SysTmEnd |
|---------|--------|----------|----------|
|---------|--------|----------|----------|

```
INSERT INTO EMPDB FOR PORTION OF SYSTEM_TIME
FROM DATE TRANSACTION TIME TO MAXVALUE
VALUES ('John Smith', '75000')
```

| Empname | Salary | SysTmSta | SysTmEnd |
|------------|--------|-----------|------------|
| John Smith | 75000 | 6/15/2007 | 12/31/9999 |

Current Row

```
UPDATE EMPDB FOR PORTION OF SYSTEM_TIME
FROM DATE TRANSACTION TIME TO MAXVALUE
SET Salary=Salary+5000
WHERE Empname='John Smith'
```

| Empname | Salary | SysTmSta | SysTmEnd |
|------------|--------|-----------|------------|
| John Smith | 75000 | 6/15/2007 | 6/15/2008 |
| John Smith | 80000 | 6/15/2008 | 12/31/9999 |

History Row

Current Row

```
DELETE FROM EMPDB FOR PORTION OF SYSTEM_TIME
FROM DATE TRANSACTION TIME TO MAXVALUE
WHERE Empname = 'John Smith'
```

| Empname | Salary | SysTmSta | SysTmEnd |
|------------|--------|-----------|-----------|
| John Smith | 75000 | 6/15/2007 | 6/15/2008 |
| John Smith | 80000 | 6/15/2008 | 9/15/2008 |

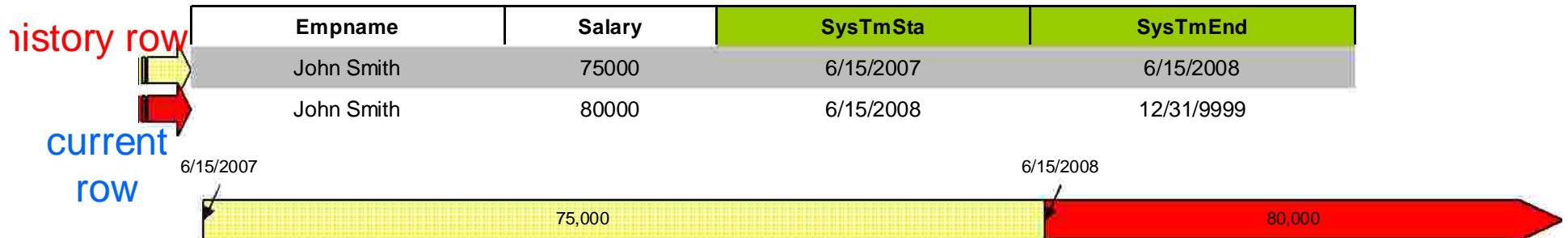
History Row

History Row

| Step | Date | Activity |
|------|-----------|----------------------------|
| 1 | 6/15/2007 | New Employee Hired |
| 2 | 6/15/2008 | Employee Gets Salary Raise |

| Empname | Salary | SysTmSta | SysTmEnd |
|------------|--------|-----------|------------|
| John Smith | 75000 | 6/15/2007 | 12/31/9999 |

UPDATE EMPDB FOR PORTION SYSTEM_TIME
 FROM DATE TRANSACTION TIME TO
 MAXVALUE
 SET Salary=Salary+5000
 WHERE Empname='John Smith'



Current on 6/30/2008
 SELECT Salary from EMPDB
 FOR SYSTEM_TIME AS OF CURRENT TIME
 WHERE Empname='John Smith'

| | |
|--------|-------|
| Salary | 80000 |
|--------|-------|

Prior-point-in-time
 SELECT Salary from EMPDB
 FOR SYSTEM_TIME AS OF '1/15/2008'
 WHERE Empname='John Smith'

| | |
|--------|-------|
| Salary | 75000 |
|--------|-------|

Recap: DB2 System Time

- Application on current data preserved
 - Insert, Update, Delete, Select unchanged
 - Triggers unchanged
 - Constraints unchanged
- DBMS preserves current application by moving history rows to separate table
- SELECTS w/o AS OF behave as if **FOR SYSTEM_TIME AS OF CURRENT TIME** transparently added
- DBMS translates AS OF times into predicates on system time column pair
- INSERT/UPDATE/DELETE behave as if **FOR PORTION OF SYSTEM_TIME FROM TRANSACTION TIME TO MAXVALUE** transparently added
- DBMS optimizer presents current and history tables as one, pulls answers from both as needed and presents as one answer set

Performance Study: DB2 provided system time support vs. RYO trigger solution

No History Generation

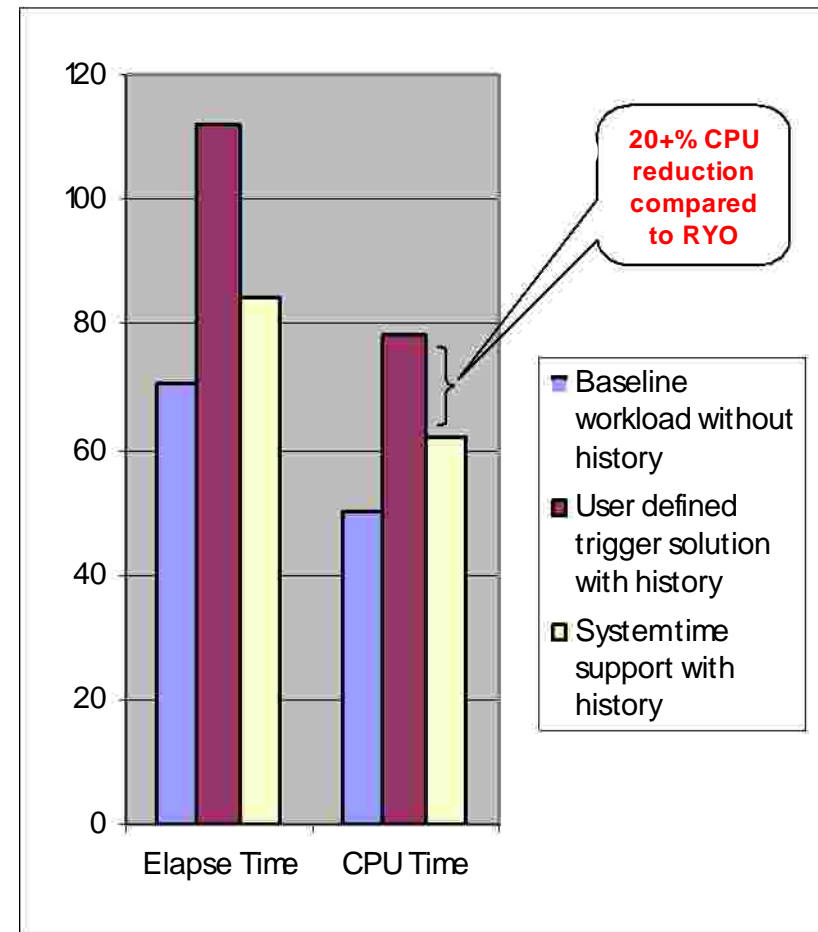
1. Foll. Txn mix run against in-house TPC-H database
70% read (SELECT)
30% write (10% INSERT + 20% UPDATE/DELETE)

RYO History Generation

2. Same Txn mix run against in-house TPC-H database enhanced with Update and Delete triggers creating historical rows
Historical Rows created in separate History table

DB2 History Generation

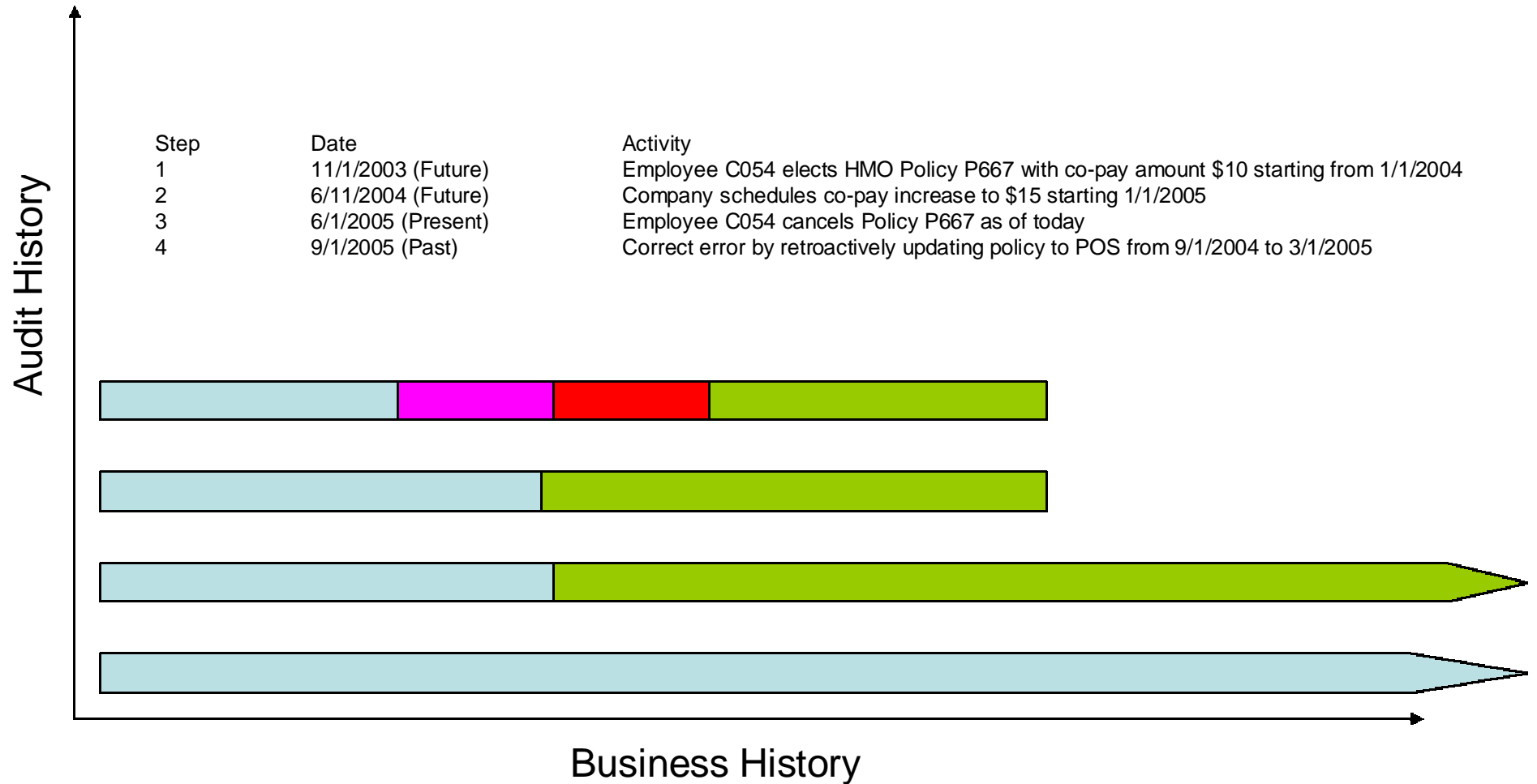
- § Same Txn mix run against enhanced TPC-H schema supporting SYSTEM TIME period, transaction ID, History table and versioning



Background of the development of the Temporal DB feature - what we heard from your customers and how that was translated that into a product feature?

- From a VP of a top investment bank: “Just about all our trades get recorded correctly, but once in a while we find that a trader has entered a trade incorrectly. We need the ability to audit our changes and also to make retroactive updates to customer positions. We require two dimensional tombstoning of data: business history and audit history
 - Translated into bitemporal support
- From lead application developer at major insurance company: “We need audit history for to be able to catch and correct errors, our compliance mandate requires us to keep the history of the evolution of all policies we have issued.”
 - Reinforced the need for bitemporal support

Bi-temporal Use Case Health Insurance



DDL: Bi-Temp/Health Insurance

```
CREATE TABLE Policy
  Empl VARCHAR(4) NOT NULL,
  Type VARCHAR(4),
  Plcy VARCHAR(4) NOT NULL,
  Copay VARCHAR(4),
  Eff-Beg DATE,
  Eff-End DATE,
  Sys-Beg TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
  Sys-End TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END, ...
  PERIOD BUSINESS_TIME (Eff-Beg, Eff-End),
  PERIOD SYSTEM_TIME (Sys-Beg, Sys-End),
  PRIMARY KEY (Empl,Plcy BUSINESS_TIME WITHOUT OVERLAPS)
);
```

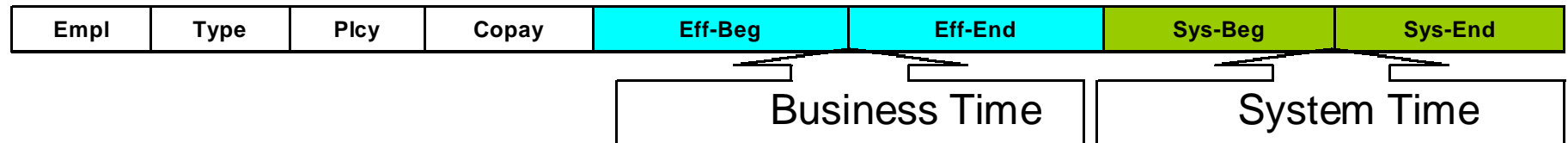
```
CREATE TABLE PolicyHistory (Bk ....);
```

```
ALTER TABLE Policy
  ADD VERSIONING ON SYSTEM_TIME
  USING PolicyHistory;
```

| | | | | | | | |
|------|------|------|-------|---------|---------|---------|---------|
| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|---------|---------|---------|---------|

DML: Bi-Temp/Health Insurance

| | Date | Kind of Event | Event |
|---|-----------|---------------|---|
| 1 | 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |



INSERT INTO POLICY
VALUES ('C054','HMO', 'P667', '\$10', '1/1/2004', '12/31/9999')

1/1/2004



| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|------------|-----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 11/1/2003 | 12/31/9999 |

Bitemporal Use Case: Health Insurance - DML

| | Date | Kind of Event | Event |
|---|-----------|---------------|---|
| 1 | 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 | 6/11/2004 | Future | Company schedules co-pay increase to \$15 starting 1/1/2005 |

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|------------|-----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 11/1/2003 | 12/31/9999 |



UPDATE POLICY FOR PORTION OF BUSINESS_TIME
FROM DATE '1/1/2005' TO '12/31/9999'
SET Copay='\$15'
WHERE Plcy='P667';



| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|------------|------------|-------------|-------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 11/1/2003 | 6/11/2004 ● |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 ● | 6/11/2004 ● | 12/31/9999 |
| C054 | HMO | P667 | \$15 | 1/1/2005 ● | 12/31/9999 | 6/11/2004 ● | 12/31/9999 |

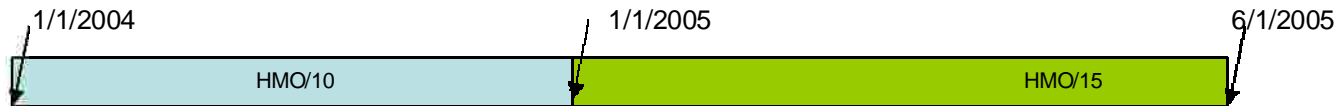
Bitemporal Use Case: Health Insurance - DML

| | Date | Kind of Event | Event |
|---|-----------|---------------|---|
| 1 | 11/1/2003 | Future | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 | 6/11/2004 | Future | Company schedules co-pay increase to \$15 starting 1/1/2005 |
| 3 | 6/1/2005 | Present | Employee C054 cancels Policy P667 as of today |

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|------------|-----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 1/11/2003 | 6/11/2004 |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 | 6/11/2004 | 12/31/9999 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 12/31/9999 | 6/11/2004 | 12/31/9999 |



**DELETE POLICY FOR PORTION OF BUSINESS_TIME
FROM CURRENT DATE TO '12/31/9999'
WHERE Empl='C054' AND Plcy='P667';**

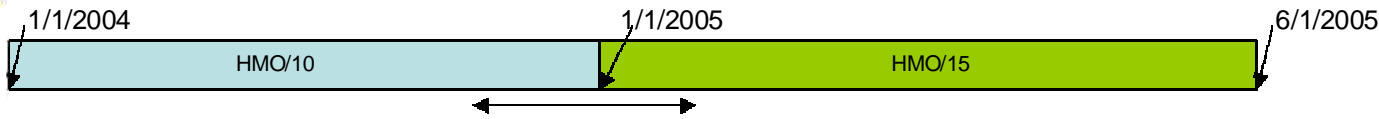


| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|------------|------------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 11/1/2003 | 6/11/2004 |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 | 6/11/2004 | 12/31/9999 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 12/31/9999 | 6/11/2004 | 6/1/2005 ● |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 6/1/2005 ● | 6/1/2005 ● | 12/31/9999 |

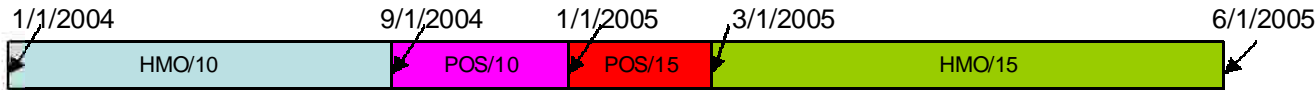
- 1 11/1/2003 Future
- 2 6/11/2004 Future
- 3 6/1/2005 Present
- 4 9/1/2005 Past

Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004
 Company schedules co-pay increase to \$15 starting 1/1/2005
 Employee C054 cancels Policy P667 as of today
 Correct error by retroactively updating policy to POS from 9/1/2004 to 3/1/2005

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|------------|-----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 11/12003 | 6/11/2004 |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 | 6/11/2004 | 12/31/9999 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 12/31/9999 | 6/11/2004 | 6/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 6/1/2005 | 6/1/2005 | 12/31/9999 |



UPDATE POLICY FOR PORTION OF BUSINESS_TIME
 FROM '9/1/2004' TO '3/1/2005'
 SET TYPE='POS' WHERE Empl='C054' AND Plcy='P667';

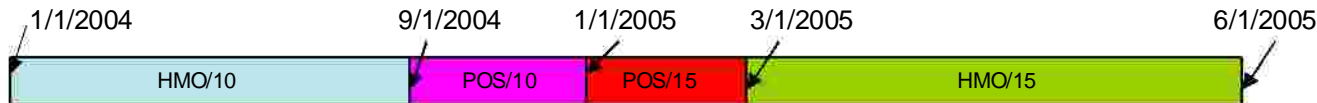


| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|------------|-----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 11/1/2003 | 6/11/2004 |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 | 6/11/2004 | 9/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 12/31/9999 | 6/11/2004 | 6/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 6/1/2005 | 6/1/2005 | 9/1/2005 |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 9/1/2004 | 9/1/2005 | 12/31/9999 |
| C054 | POS | P667 | \$10 | 9/1/2004 | 1/1/2005 | 9/1/2005 | 12/31/9999 |
| C054 | POS | P667 | \$15 | 1/1/2005 | 3/1/2005 | 9/1/2005 | 12/31/9999 |
| C054 | HMO | P667 | \$15 | 3/1/2005 | 6/1/2005 | 9/1/2005 | 12/31/9999 |

| Step | Date | Activity |
|------|--------------------|---|
| 1 | 11/1/2003 (Future) | Employee C054 elects HMO Policy P667 with co-pay amount \$10 starting from 1/1/2004 |
| 2 | 6/11/2004 (Future) | Company schedules co-pay increase to \$15 starting 1/1/2005 |
| 3 | 6/1/2005 (Present) | Employee C054 cancels Policy P667 as of today |
| 4 | 9/1/2005 (Past) | Correct error by retroactively updating policy to POS from 9/1/2004 to 3/1/2005 |

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|------------|-----------|------------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 12/31/9999 | 11/1/2003 | 6/11/2004 |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 | 6/11/2004 | 9/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 12/31/9999 | 6/11/2004 | 6/1/2005 |
| C054 | HMO | P667 | \$15 | 1/1/2005 | 6/1/2005 | 6/1/2005 | 9/1/2005 |
| C054 | HMO | P667 | \$10 | 1/1/2004 | 9/1/2004 | 9/1/2005 | 12/31/9999 |
| C054 | POS | P667 | \$10 | 9/1/2004 | 1/1/2005 | 9/1/2005 | 12/31/9999 |
| C054 | POS | P667 | \$15 | 1/1/2005 | 3/1/2005 | 9/1/2005 | 12/31/9999 |
| C054 | HMO | P667 | \$15 | 3/1/2005 | 6/1/2005 | 9/1/2005 | 12/31/9999 |

Business Question: On 9/15/2005 client calls & complains: client saw an out-of-network specialist on 11/1/2004, our claims dept. denied client's claim for this visit on 11/15/2004. Client needs to be reimbursed. Did Claims Dept. make an error?



SELECT * FROM POLICY FOR BUSINESS_TIME AS OF DATE '11/1/2004' FOR SYSTEM_TIME AS OF '11/15/2004' WHERE Empl='C054' AND Plcy='P667';

| Empl | Type | Plcy | Copay | Eff-Beg | Eff-End | Sys-Beg | Sys-End |
|------|------|------|-------|----------|----------|-----------|----------|
| C054 | HMO | P667 | \$10 | 1/1/2004 | 1/1/2005 | 6/11/2004 | 9/1/2005 |

No, Claims Dept did not make an error

Summary

- Simple, elegant but powerful SQL based temporal function in zDB2 V10
 - Offers real reduction in cost for one and two dimensional History tracking applications common in Insurance, Brokerage, Banking, Retail, Telecom, ..
 - “ .. simply use instead of re-implementing the whole darn thing each time”
- Extensions being considered for the future
 - Support of Timestamp with Timezone datatype
 - Non disruptive schema change propagation from current to history
 - Support for (Inclusive, Inclusive) period convention
 - Column reference in AS OF clause
 - View and Time Machine support
 - Support of no gaps constraint
 - Temporal RI
 - Controlled propagation of temporal rows without change in system time
 - Replication
 - Oltp to Warehouse migration
 - Period generation based on business transactions rather than DB2 transactions
 - Space Filling/Unforgiving UPDATE FOR PORTION OF so gaps in time are filled or disallowed
 - Coalesce
 - Interoperability with column and row access control
 - ASOF Query optimization
 - Union Optimization
 - Augmented History
 - Suppressing history generation when update does not make any modifications
 - Designating don't care columns whose updates don't trigger history generation
 - Efficient History purge