

## What DB2 brings to your Temporal Apps?

- Cost savings via up to 10x<sup>1</sup> Application Simplification
  - Complexity
  - Lines of code
- Up to 10x<sup>1</sup> Reduction in Time to Deployment
- Better ROI from consultant engagement

<sup>1</sup>: 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
- 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

The information on the new product is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information on the new product is for informational purposes only and may not be incorporated into any contract. The information on the new product is not a commitment, promise, or legal obligation to deliver any material, code or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion.

# Summary of DB2 Support

- 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

# Business Time Use Case: Health Insurance

	Date	Kind of Event	Event
1	11/1/2003	Future	Employee C054 elects <u>Indv</u> 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 <u>Fmly</u> 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 Indv Policy P667 with co-pay amount \$10 starting from 1/1/2004

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

Business Time

INSERT INTO POLICY

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


1/1/2004



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

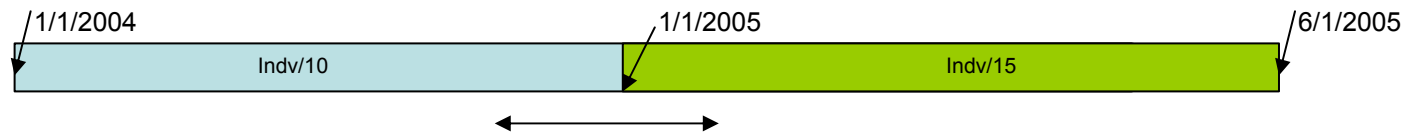
DELETE POLICY FOR PORTION OF BUSINESS\_TIME  
FROM CURRENT DATE TO '12/31/9999'  
WHERE Empl='C054' AND Plcy='P667';



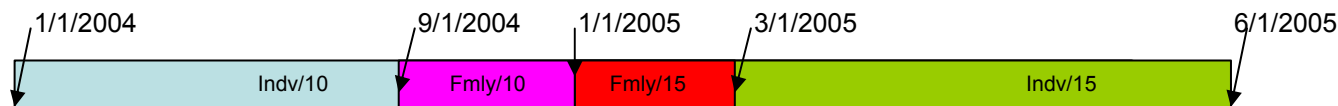
Empl	Type	Plcy	Copay	Eff-Beg	Eff-End
C054	Indv	P667	\$10	1/1/2004	1/1/2005
C054	Indv	P667	\$15	1/1/2005	6/1/2005

Date	Kind of Event	Event
1 11/1/2003	Future	Employee C054 elects Indv 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 Fmly from 9/1/2004 to 3/1/2005

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

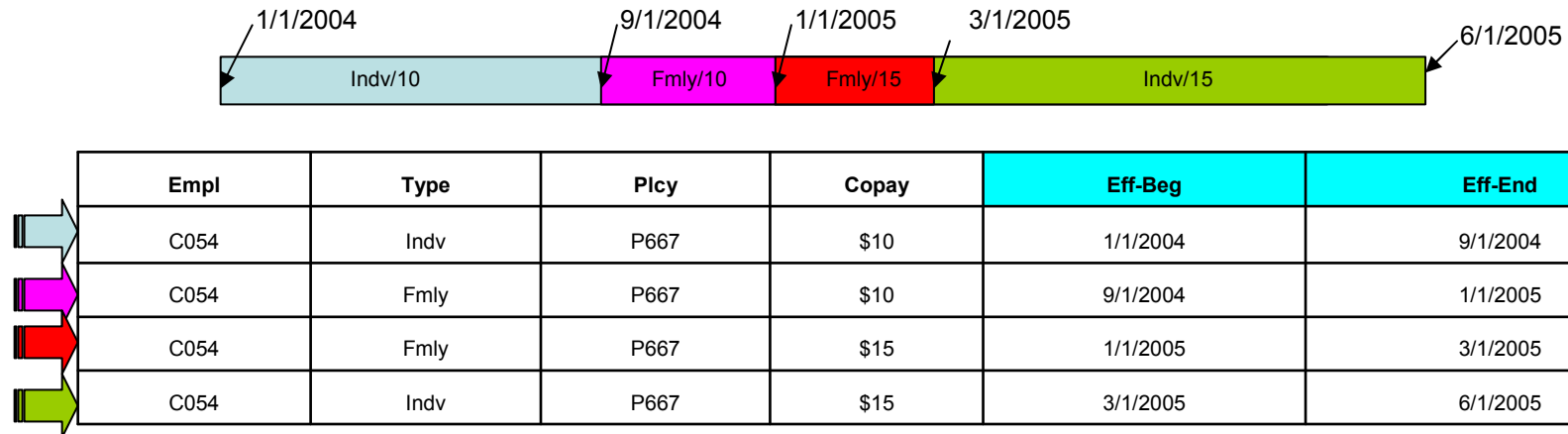


UPDATE POLICY FOR PORTION OF BUSINESS\_TIME  
FROM '9/1/2004' TO '3/1/2005'  
SET TYPE='Fmly'  
WHERE Empl='C054' AND Plcy='P667';



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

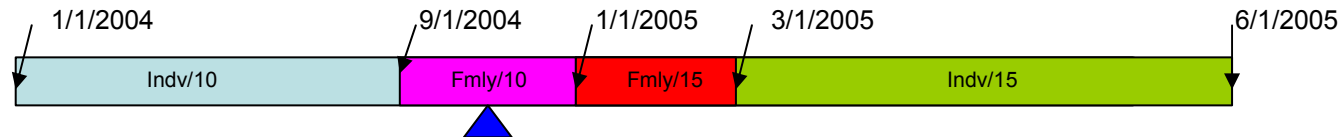
# KEY WITHOUT OVERLAPS



- 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 Indv 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 Fmly from 9/1/2004 to 3/1/2005



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

Business Question: On 9/15/2005 client calls & complains: client took daughter to see doctor on 11/1/2004, our claims dept. denied client's claim for this visit on 11/15/2004. Client is upset, demands reimbursement.

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

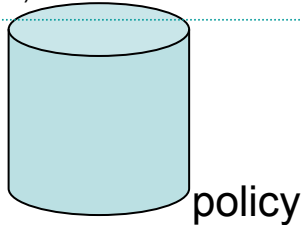
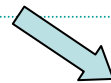
Empl	Type	Plcy	Copay	Eff-Beg	Eff-End
C054	Fmly	P667	\$10	9/1/2004	1/1/2005

Business Answer: On 11/1/2004 client had Fmly coverage and is eligible to be reimbursed for taking daughter to see doctor.

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

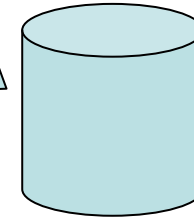
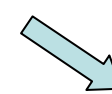
# Performance of zDB2 Business Time

```
UPDATE policy SET PREMIUM = 888, temp_begin  
= '2010-02-13', temp_end = '2010-02-15'  
WHERE policy_id = 1234  
AND (DATE('2010-02-13') >= BUS_BEGIN AND  
DATE('2010-02-13') < BUS_END)  
OR (DATE('2010-02-15') > BUS_BEGIN AND  
DATE('2010-02-15') <= BUS_END)  
OR (DATE('2010-02-13') <= BUS_BEGIN AND  
DATE('2010-02-15') >= BUS_END)
```



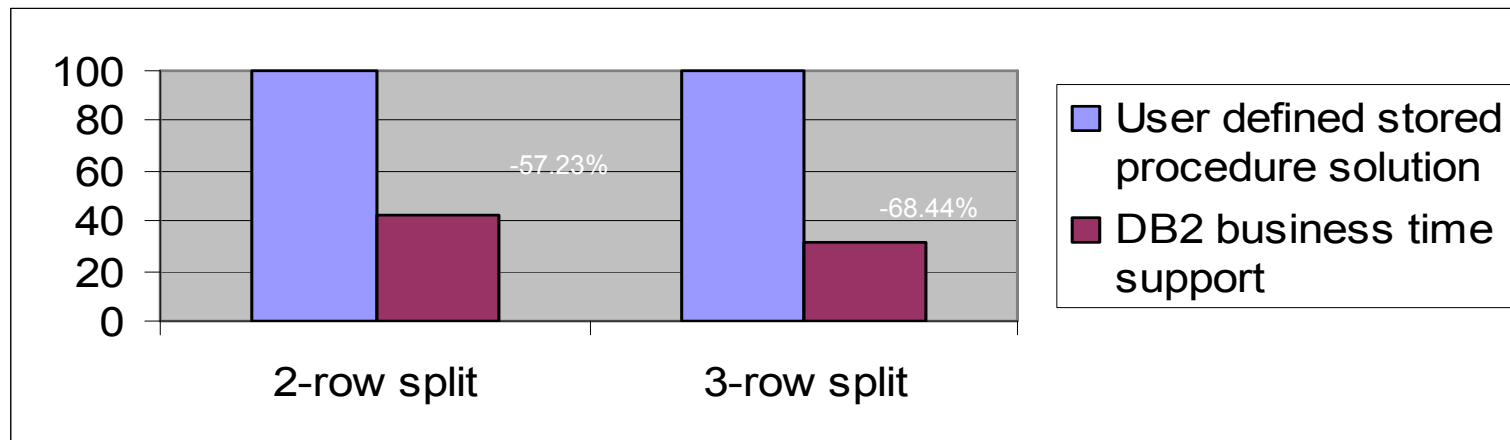
BUSINESS TIME PERIOD  
NOT DEFINED

```
UPDATE temporal_policy FOR PORTION OF  
BUSINESS_TIME FROM DATE('2010-02-13')  
TO DATE('2010-02-15') SET PREMIUM =  
888 WHERE policy_id = 1234
```



BUSINESS TIME PERIOD  
DEFINED

In DB2  
code path  
reduction



# 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 ('Peter Karlis', '75000')
```

Empname	Salary
Peter Karlis	75,000

Step 2

```
UPDATE EMPDB
SET Salary=Salary+5000
WHERE Empname='Peter Karlis'
```

Empname	Salary
Peter Karlis	80,000

Step 3

```
DELETE FROM EMPDB
WHERE Empname = 'Peter Karlis'
```

Empname	Salary
---------	--------

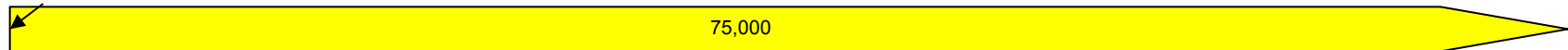
Significant part of compliance requirement may met by preserving rows updated/deleted.

Step	Date	Activity	CREATE TABLE EMPDB (Empname VARCHAR (40), Salary INTEGER, SysTmSta <b>TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN</b> , SysTmEnd <b>TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END</b> , ... <i>PERIOD SYSTEM_TIME (SysTmSta, SysTmEnd)</i> );
1	6/15/2007	New Employee Hired	
2	6/15/2008	Employee Gets Salary Raise	
3	9/15/2008	Employee quits	

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

INSERT INTO EMPDB VALUES ('Peter Karlis', '75000')

6/15/2007

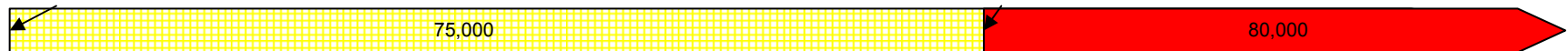


Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	12/31/9999

UPDATE EMPDB  
SET Salary=Salary+5000 WHERE Empname='Peter Karlis'

6/15/2007

6/15/2008



Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	6/15/2008
Peter Karlis	80000	6/15/2008	12/31/9999

DELETE FROM EMPDB WHERE Empname = 'Peter Karlis'

6/15/2007

6/15/2008

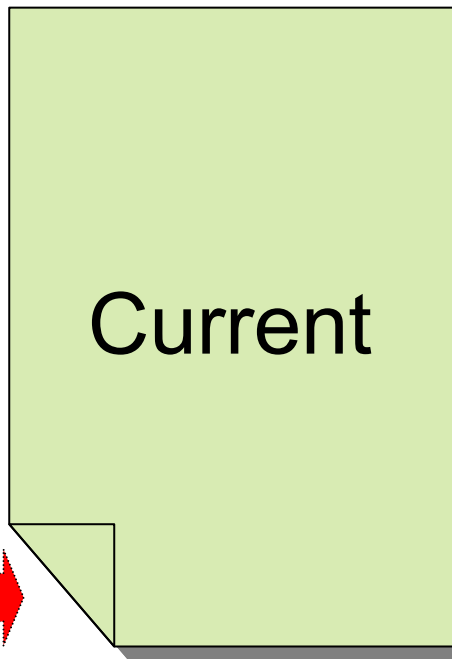
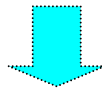
9/15/2008



Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	6/15/2008
Peter Karlis	80000	6/15/2008	9/15/2008

# Current and History

Current SQL Application



Current

History  
Generation



Jul 2008

Aug 2008

Sep 2008

Audit

History

Auditing SQL Application  
Using ASOF

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 ADD VERSIONING USE HISTOY TABLE EMPHIST
```

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

```
INSERT INTO EMPDB
VALUES ('Peter Karlis', '75000')
```

Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	12/31/9999

Current Row

```
UPDATE EMPDB
SET Salary=Salary+5000
WHERE Empname='Peter Karlis'
```

Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	6/15/2008
Peter Karlis	80000	6/15/2008	12/31/9999

History Row

Current Row

```
DELETE FROM EMPDB
WHERE Empname = 'Peter Karlis'
```

Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	6/15/2008
Peter Karlis	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



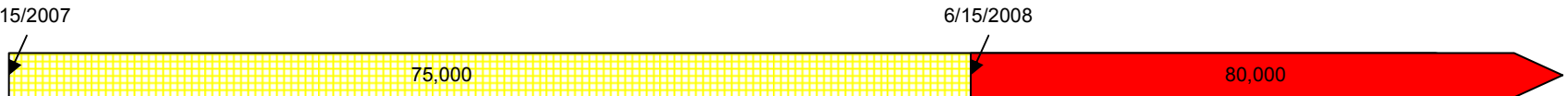
Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	12/31/9999

UPDATE EMPDB SET Salary=Salary+5000  
WHERE Empname='Peter Karlis'

history row

current row

Empname	Salary	SysTmSta	SysTmEnd
Peter Karlis	75000	6/15/2007	6/15/2008
Peter Karlis	80000	6/15/2008	12/31/9999



Current on 6/30/2008      SELECT Salary from EMPDB  
WHERE Empname='Peter Karlis'

Salary
80000

Prior-point-in-time      SELECT Salary from EMPDB  
FOR **SYSTEM TIME** AS OF '1/15/2008'  
WHERE Empname='Peter Karlis'

Salary
75000

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

## *No History Generation*

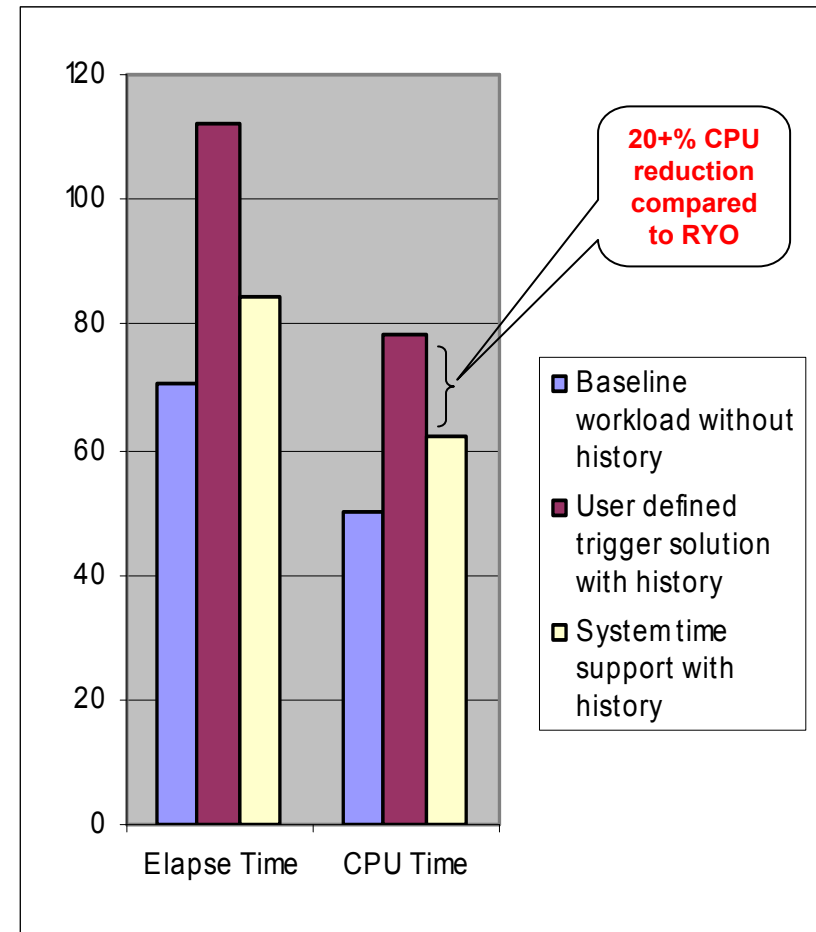
1. Foll. Txn mix run against in-house TPC-H database  
70% read (SELECT)  
30% write (10% INSERT + 20% UPDATE/DELETE)

## *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

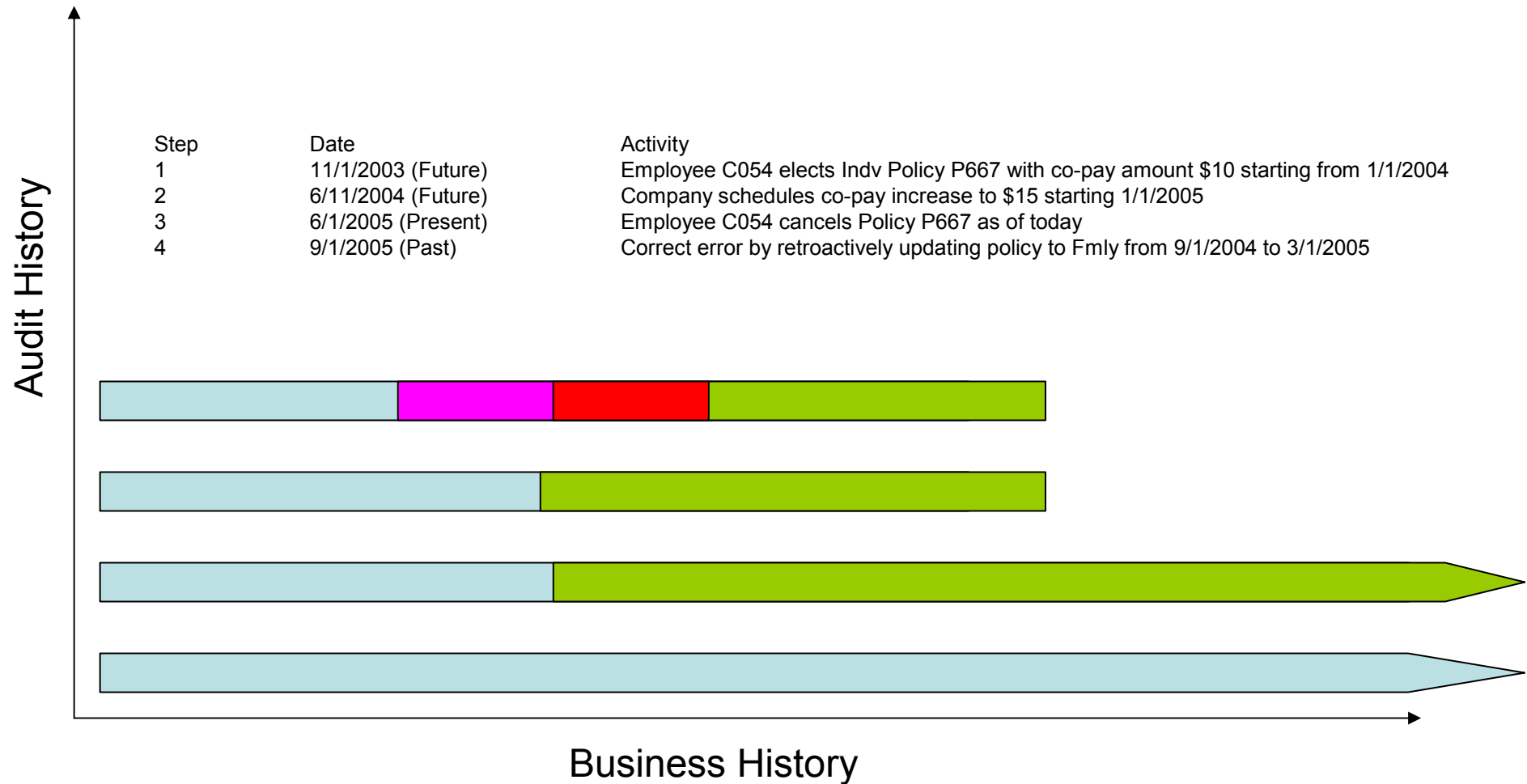
## *DB2 History Generation*

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





# 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
------	------	------	-------	---------	---------	---------	---------

# Bitemporal Use Case: Health Insurance - DML

	Date	Kind of Event	Event
1	11/1/2003	Future	Employee C054 elects Indv 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	Indv	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	Indv	P667	\$10	1/1/2004	12/31/9999	11/1/2003	6/11/2004
C054	Indv	P667	\$10	1/1/2004	1/1/2005	6/11/2004	12/31/9999
C054	Indv	P667	\$15	1/1/2005	12/31/9999	6/11/2004	12/31/9999

# DML: Bi-Temp/Health Insurance

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

Empl	Type	Plcy	Copay	Eff-Beg	Eff-End	Sys-Beg	Sys-End
				Business Time		System Time	

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




1/1/2004



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



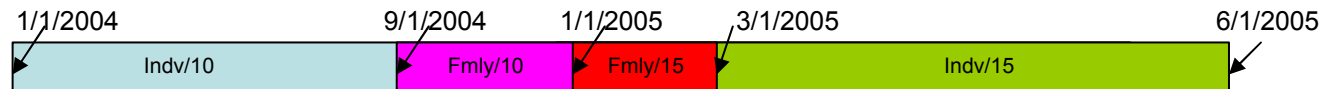
	Empl	Type	Plcy	Copay	Eff-Beg	Eff-End	Sys-Beg	Sys-End
	C054	Indv	P667	\$10	1/1/2004	12/31/9999	11/1/2003	6/11/2004
	C054	Indv	P667	\$10	1/1/2004	1/1/2005	6/11/2004	12/31/9999
	C054	Indv	P667	\$15	1/1/2005	12/31/9999	6/11/2004	6/1/2005 
	C054	Indv	P667	\$15	1/1/2005	6/1/2005 	6/1/2005 	12/31/9999

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

Employee C054 elects Indv 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 Fmly from 9/1/2004 to 3/1/2005











Empl	Type	Plcy	Copay	Eff-Beg	Eff-End	Sys-Beg	Sys-End
C054	Indv	P667	\$10	1/1/2004	12/31/9999	11/12003	6/11/2004
C054	Indv	P667	\$10	1/1/2004	1/1/2005	6/11/2004	12/31/9999
C054	Indv	P667	\$15	1/1/2005	12/31/9999	6/11/2004	6/1/2005
C054	Indv	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='Fmly' WHERE Empl='C054' AND Plcy='P667';

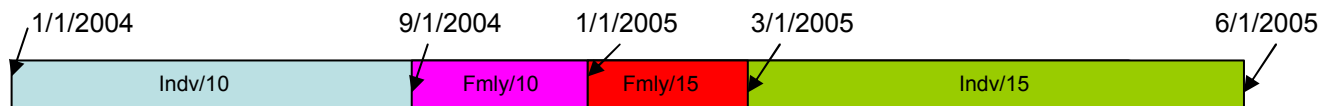


Empl	Type	Plcy	Copay	Eff-Beg	Eff-End	Sys-Beg	Sys-End
C054	Indv	P667	\$10	1/1/2004	12/31/9999	11/1/2003	6/11/2004
C054	Indv	P667	\$10	1/1/2004	1/1/2005	6/11/2004	9/1/2005
C054	Indv	P667	\$15	1/1/2005	12/31/9999	6/11/2004	6/1/2005
C054	Indv	P667	\$15	1/1/2005	6/1/2005	6/1/2005	9/1/2005
C054	Indv	P667	\$10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

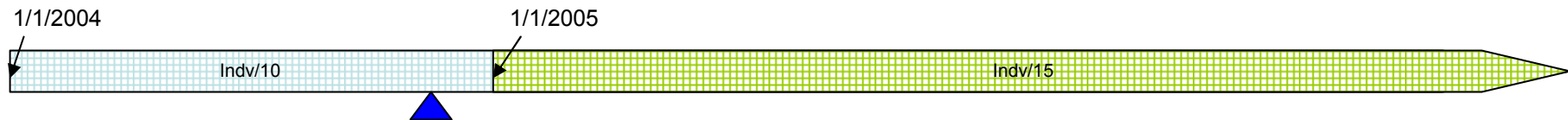
Step	Date	Activity
1	11/1/2003 (Future)	Employee C054 elects Indv Policy P667 with co-pay amount \$10 starting from 1/1/2004
2	6/11/2004 (Future)	Company schedules co-pay increase to \$15 starting 1/1/2005
3	6/1/2005 (Present)	Employee C054 cancels Policy P667 as of today
4	9/1/2005 (Past)	Correct error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


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

On 9/15/2005 client calls & complains: client took daughter to see doctor on 11/1/2004, our claims dept. denied client's claim for this visit on 11/15/2004. Did our claims department 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	Indv	P667	\$10	1/1/2004	1/1/2005	6/11/2004	9/1/2005

No, claims dept did not make an error, the state of the policy was diff on processing date,

# Savings in productivity

Item	User application solution	DB2 supplied solution
System Time Support	2 triggers for each table	Provided by DB2
Business Time Support	2 stored procedures for each table	Provided by DB2 via SQL statements
Period overlap detection	1 trigger for each table	Part of PK index for each table
Total number of lines of code	650	13
Total number of SQL statements	27	3
Time to develop and test	6 weeks	<1 hour



# Road Map

- Payload Columns: Who, What, When, Why: Option to record userID & applicationID
- Logical Transaction Time
- Temporal Referential Integrity
- Provide option to keep current and history rows in a single table
- Support for temporal in ORM frameworks, e.g., Hibernate and JPA
- Allow MQTs to be temporal
- Ability for triggers to distinguish "internal" inserts (from row splits) from "external" inserts
- Period operators: OVERLAPS, BEFORE, AFTER, etc.
- Enforcement of "no gaps" between business time periods
- Dummy updates that don't modify a row should not produce history
- Don't Care Columns
- Versioning for DB2 catalog tables
- Coalesce: Collapse rows with adjacent business periods but otherwise identical values
- Allow SELECT FROM FINAL TABLE for temporal I/U/D
- Support for inclusive-inclusive periods
- Change default BUSTIMESENSITIVE and SYSTIMESENSITIVE bind options to NO
- Business time support for hash organized temporal tables
- User supplied values for row begin and row end populated by call to exit routine
- Support for user defined calendars (allow BigInt & VARCHAR datatypes for periods)
- Allow triggers to reference the FROM and TO values when fired by an UPDATE or DELETE FOR PORTION OF BUSINESS\_TIME FROM t1 TO t2
- User-defined period names
- Multiple business-time periods per table
- Qualifying period names - To support period specifications at a more granular level you might want to qualify the name of the period with the table/view name or correlation name for the table reference.
- Drop period - Allow a period in a table to be dropped.
- Create table LIKE to create history table - Create another variant of 'CREATE TABLE ... LIKE table2' which copies/inherits attributes as appropriate for the definition of a history table. This would be a bit different from the normal rules of inheriting attributes.
- Allow definition of unique constraint on a history table.
- History Table pruning - Different options for removing data from a history table.
- Temporal JOIN
- Temporal GROUPBY
- TEMPORAL Aggregates
- DELETE FPO in cursor operations
- MERGE FPO
- Trigger in statements that have FPO
- Attaching Periods to views
- Diagnostics for FPO (e.g. how many rows inserted)
- Temporal business object

*We welcome clients who would like to work with IBM and direct the future of database temporal support.  
Contact Bala Iyer ([balaayer@us.ibm.com](mailto:balaayer@us.ibm.com)) through your IBM client advocate to get started.*