IBM Information ON Demand 2010

Create Insight. Transform. Go Beyond.

TSB-2908A

New DB2 10 for z/OS Security Features Help Satisfy Your Auditors

James Pickel
Silicon Valley Laboratory
DB2 for z/OS Security
pickel@us.ibm.com

Housekeeping

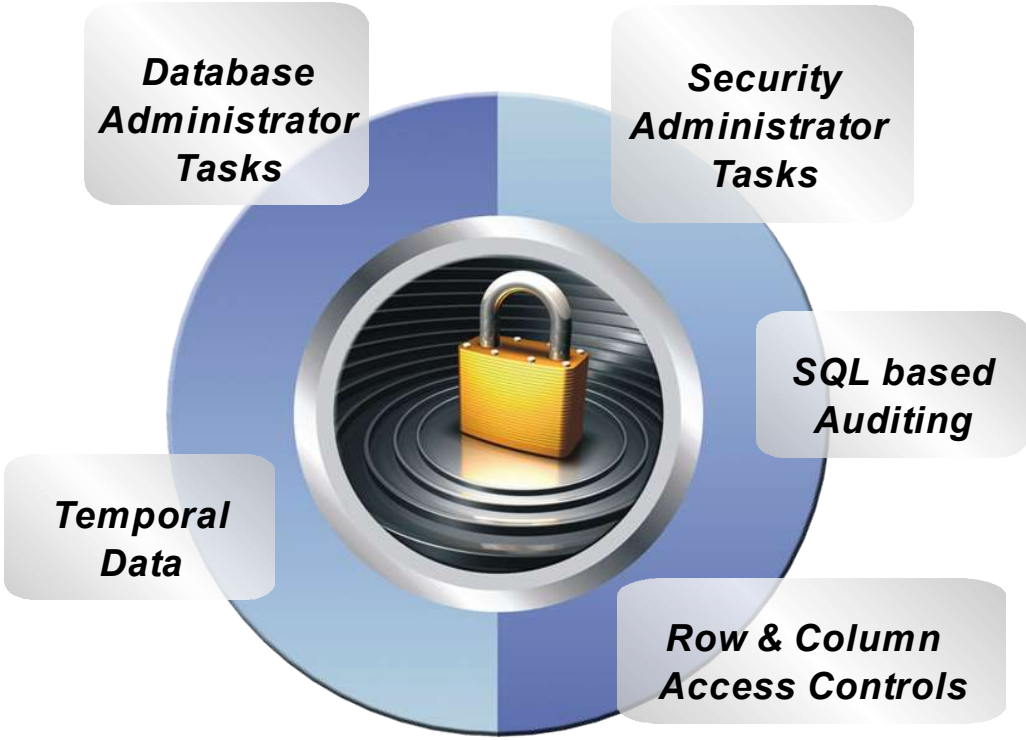
- We value your feedback - don't forget to complete your evaluation for each session you attend and hand it to the room monitors at the end of each session
- Overall Conference Evaluation will be provided at the General Session on Friday
- Visit the Expo Solutions Centre
- Please remember this is a 'non-smoking' venue!
- Please switch off your mobile phones
- Please remember to wear your badge at all times

IBM Disclaimer

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

Satisfy Your Auditor: Plan, Protect and Audit

- Unauthorized Data Access
  - Minimize the use of a superuser authorities such as SYSADM
  - A different group should manage access to restricted data than the owner of the data
- Data Auditing
  - Any dynamic access or use of a privileged authority needs to be included in your audit trail
  - Maintain historical versions of data for years or during a business period
- Data Privacy
  - All dynamic access to tables containing restricted data needs to be protected



Today's Mainframe:
The power of industry-leading security, the simplicity of centralised management

Minimize the use of SYSADM

- New granular system authorities and install security parameters

Prior to DB2 10

- SYSADM
- DBADM
- DBCTRL
- DBMAINT
- SYSCTRL
- PACKADM
- SYSOPR

New in DB2 10

- Install SECADM
- System DBADM
- ACCESSCTRL
- DATAACCESS
- SQLADM
- EXPLAIN

Prevent SYSADM and SYSCTRL from granting or revoking privileges

- New separate security install parameter
- New install SECADM has grant and revoke privilege

Control cascading effect of revokes

- New revoke dependent privileges install parameter
- New revoke dependent privileges SQL clause

New System Authorities Details

- Install security administrator
  - **SECADM Authority**
  - Security related tasks (GRANT,REVOKE,ROLE..)
- System level database administrator
  - **DBADM Authority**
  - Perform DDL for all data bases
- System level data access
  - **DATAACCESS Authority**
  - Access data in all tables and can execute plans, packages, functions, and procedures
- System level access control
  - **ACCESSCTRL Authority**
  - Issue GRANT or REVOKE for any object

How to create new authority that can be granted without access to data or ability to grant access

- The new security administrator can grant new **system** DBADM authority to an authorization ID or a role

```
GRANT DBADM WITHOUT DATAACCESS
                WITHOUT ACCESSCTRL TO AdminID
```

- Admin can now CREATE, ALTER, or DROP objects
- Admin does **not** have implicit GRANT or REVOKE privileges
- Admin does **not** have implicit execute privileges on packages or plans
- Admin does **not** have implicit table privileges

New authority for monitoring and tuning SQL without ability to change or access data

- SQLADM Authority
  - Allows the user to
    - Issue SQL EXPLAIN statements
    - Issue START, STOP, and DISPLAY PROFILE commands
  - Perform actions involving:
    - EXPLAIN privilege
    - STATS privilege on all user databases
    - MONITOR2 privilege
  - Execute DB2-supplied stored procedures and routines
  - Cannot access data, perform DDL or execute

New privilege for programmers to validate their SQL before moving to production

- EXPLAIN privilege
  - Programmer can issue SQL EXPLAIN ALL statement without having the privileges to execute that SQL statement.
  - Programmer can issue SQL PREPARE and DESCRIBE TABLE statements without requiring any privileges on the object.
  - Programmer can specify new BIND EXPLAIN(ONLY) and SQLERROR(CHECK) options
  - Programmer can explain dynamic SQL statements executing under new special register, CURRENT EXPLAIN MODE = EXPLAIN

GRANT EXPLAIN to ProgGroup

Satisfy Your Auditor: Audit Policies provide needed flexibility and functionality

- New auditing capability allow you to comply without expensive external data collectors
  - New audit policies managed in catalog
  - Audit privileged users
  - Audit SQL activity against a table
  - Audit non-z/OS distributed identities

New Audit Policy Feature

- SECADM maintains system audit policies in new catalog table

- Auditor can audit access to specific tables for specific programs during day
  1) Audit policy does not require AUDIT clause to be specified to enable table auditing
  2) Audit policy generate records for all read and update access not just first access in transaction
  3) Audit policy includes additional records identifying the specific SQL statements reading or updating an audited UTS table
  4) Audit policy provides wildcarding of table names

- Auditor can can identify any unusual use of a privileged authority during day
  - Records each use of a system authority
  - Audit records written only when authority is used for access
  - External collectors only report users with a system authority

Satisfy Your Auditor:
Audit trace includes authenticated identities

- Support distributed identities introduced in z/OS V1R11
  - A distributed identity is a mapping between a RACF user ID and one or more distributed user identities, as they are known to application servers
- Support client certificates and password phrases introduced in z/OS V1R10
  - AT-TLS secure handshake accomplishes identification and authentication when the client presents its certificate as identification and its proof-of-possession as authentication
  - A password phrase is a character string made up of mixed-case letters, numbers, and special characters, including blanks, and is between 9 to 100 characters long.
- Support connection level security enforcement
  - Requires all connections use strong authentication to access DB2
  - TCPALVER parameter changed to support SERVER_ENCRYPT
  - All userids and passwords encrypted using AES, or connections accepted on a port which ensures AT-TLS policy protection or protected by an IPSec encrypted tunnel

Satisfy Your Auditor:
Protect against unplanned and all dynamic access

- Define additional data controls at the table level
  - Security policies are defined using SQL providing flexibility
  - Separate security logic from application logic
- Security policies based on real time session attributes
  - Protects against SQL injection attacks
  - Determines how column values are returned
  - Determines which rows are returned

- No need to remember various view or application names
  - No need to manage many views; no view update or audit issues
- All access including adhoc query tools, report generation tools is protected
- Policies can be added, modified, or removed to meet current company rules without change to applications

New controls to protect access to individual rows

- Establish a row policy for a table
  - Filter rows out of answer set
  - Policy can use session information like the SQL ID is in what group or user is using what role to control when row is returned in result set
  - Applicable to SELECT, INSERT, UPDATE, DELETE, & MERGE
  - Defined as a row permission:
    CREATE PERMISSION policy-name ON table-name
    FOR ROWS WHERE search-condition
    ENFORCED FOR ALL ACCESS ENABLE;

Optimizer inserts search condition in all SQL statements accessing table. If row satisfies search-condition, row is returned in answer set

New controls to protect access to sensitive columns

- Establish a column policy for a table
  - Mask column values in answer set
  - Policy can use session information to mask value like the SQL ID is in what group or user is using what role
  - Applicable to the output of outermost subselect
  - Defined as column masks :

CREATE MASK mask-name ON table-name

FOR COLUMN column-name RETURN CASE-expression ENABLE;

Optimizer inserts case statement in all SQL accessing table to determine mask value to return in answer set

Define a column or row policy based on who is accessing the table

- **SESSION_USER**
  - Primary authorization ID of the process
- **CURRENT SQLID**
  - SQL authorization ID of the process
  - SET CURRENT SQLID = string-constant;
- **VERIFY_GROUP_FOR_USER (new BIF)**
  - Get authorization IDs for the value in SESSION_USER
    - Primary and secondary authorization IDs
  - Return 1 if any of those authorization IDs is in the argument list

```
WHERE
   VERIFY_GROUP_FOR_USER (SESSION_USER, 'MGR', 'PAYROLL') = 1
```

- **VERIFY_ROLE_FOR_USER (new BIF)**
  - Get the role for the value in SESSION_USER
  - Return 1 if the role is in the argument list

```
WHERE
   VERIFY_ROLE_FOR_USER (SESSION_USER, 'MGR', 'PAYROLL') = 1
```

Row and Column Access Control

- When activated row and column access controls:
  - Make row permissions and column masks become effective in all DML
    - All row permissions are connected with 'OR' to filter out rows
    - All column masks are applied to mask output
  - **All access to the table if no user-defined row permissions**

```
ALTER TABLE table-name
    ACTIVATE ROW LEVEL ACCESS CONTROL
    ACTIVATE COLUMN LEVEL ACCESS CONTROL;
```

- When deactivated row and column access controls:
  - Make row permissions and column masks become ineffective in DML
  - **Opens all access to the table**

```
ALTER TABLE table-name
    DEACTIVATE ROW LEVEL ACCESS CONTROL
    DEACTIVATE COLUMN LEVEL ACCESS CONTROL;
```

Example – A simple banking scenario

- Table: CUSTOMER

Account	Name	Phone	Income	Branch
1111-2222-3333-4444	Alice	111-1111	22,000	A
2222-3333-4444-5555	Bob	222-2222	71,000	B
3333-4444-5555-6666	Louis	333-3333	123,000	B
4444-5555-6666-7777	David	444-4444	172,000	C

- Table: CUSTOMER_CHOICE

Account	Phone_Choice
1111-2222-3333-4444	1
2222-3333-4444-5555	1
3333-4444-5555-6666	0
4444-5555-6666-7777	1

- Table: INTERNAL_INFO, stores information about bank employees including EMP_ID and HOME_BRANCH

Row and Column Access Control Banking Scenario

- Determine access control rules for customer service rep
  - Allow access to all customers of the bank (a row permission)
  - Mask all INCOME values (a column mask)
    - Return value 0 for incomes of 25000 and below
    - Return value 1 for incomes between 25000 and 75000
    - Return value 2 for incomes between 75000 and 150000
    - Return value 3 for incomes above 150000
  - All are in the CSR group (who)
- Create a row permission for customer service representatives

```
CREATE PERMISSION CSR_ROW_ACCESS ON CUSTOMER
    FOR ROWS WHERE
        VERIFY_GROUP_FOR_USER (SESSION_USER, 'CSR') = 1
ENFORCED FOR ALL ACCESS;
```

Banking Scenario

- Create a column mask on INCOME column for customer service representative

```
CREATE  MASK  INCOME_COLUMN_MASK  ON  CUSTOMER

   FOR  COLUMN  INCOME  RETURN

     CASE WHEN (VERIFY_GROUP_FOR_USER (SESSION_USER, 'CSR') = 1)

                  THEN  CASE  WHEN (INCOME > 150000) THEN  3
                              WHEN (INCOME > 75000)   THEN 2
                              WHEN (INCOME > 25000)   THEN 1
                              ELSE  0
                        END

             ELSE NULL
       END
ENABLE;
```

Banking Scenario

- Activate Row-level and Column-level Access Control

```
ALTER  TABLE  CUSTOMER
    ACTIVATE ROW LEVEL ACCESS CONTROL
    ACTIVATE COLUMN LEVEL ACCESS CONTROL;
COMMIT;
```

- What happens in DB2?
  - A default row permission is created implicitly to prevent all access to table CUSTOMER (WHERE 1=0)
  - All packages and cached statements that reference table CUSTOMER are invalidated

Banking Scenario

- **SELECT ACCOUNT, NAME, INCOME, PHONE FROM CUSTOMER;**

ACCOUNT	NAME	INCOME	PHONE
1111-2222-3333-4444	Alice	**0**	111-1111
2222-3333-4444-5555	Bob	**1**	222-2222
3333-4444-5555-6666	Louis	**2**	333-3333
4444-5555-6666-7777	David	**3**	444-4444

INCOME automatically masked by DB2!

Banking Scenario

- DB2 effectively evaluates the following revised query:

```
SELECT  ACCOUNT, NAME,

            CASE WHEN (VERIFY_GROUP_FOR_USER (SESSION_USER, 'CSR') = 1)
                        THEN CASE  WHEN (INCOME > 150000) THEN 3
                                   WHEN (INCOME > 75000)   THEN 2
                                   WHEN (INCOME > 25000)   THEN 1
                                   ELSE  0
                              END
                   ELSE NULL
             END  INCOME,
           PHONE
FROM  CUSTOMER
WHERE  VERIFY_GROUP_FOR_USER (SESSION_USER, 'CSR') = 1  OR
           1 = 0 ;
;
```

Satisfy Your Auditor:
Protect against unplanned and dynamic access

- Define additional data controls at the table level
  - Security policies are defined using SQL providing flexibility
  - Separate security logic from application logic
- Security policies based on real time session attributes
  - Protects against SQL injection attacks
  - Determines how column values are returned
  - Determines which rows are returned
- No need to remember various view or application names
  - No need to manage many views; no view update or audit issues
- All access including adhoc query tools, report generation tools is protected
- Policies can be added, modified, or removed to meet current company rules without change to applications

Problem of managing different versions of application data

- Application programmers and database administrators have struggled for years with managing different versions of application data.
- New regulatory laws require maintaining historical versions of data for years.
- Every update and delete of data requires copying old data to history tables.
- Existing approaches to application level versioning complicate table design, add complexity and are error prone for applications.

Temporal Table Overview

- Two types of time sequences of table rows are supported through the introduction of database defined time periods.

- **SYSTEM_TIME** is used for system maintained history for a new concept of “versioning” which archives old rows into a history table
- **BUSINESS_TIME** is a period that represents when a row is valid to the user or application. The BUSINESS_TIME period can be used to model data in the past, present, and future as the data values are controlled by the user/application

- A unique index can be defined for a BUSINESS_TIME period to enforce non-overlapping time periods for the instances of a particular object modeled in the temporal table.
- A bitemporal table includes both periods.

New temporal tables manage different versions of data on new or existing tables

- DB2 provides a capability to specify table-level specifications to control the management of application data based upon time
- Application programmers can specify search criteria based on the time the data existed or was valid. This capability simplifies DB2 application development requiring data versioning
- Performance expectations
  - SELECT of current data would perform similar to tables that are not defined for data versioning
  - DELETE and UPDATE of current data would perform slower than for a table not defined with data versioning
  - SELECT of historical data requires retrieving data from two tables

Time Period Overview

- Period is two columns where two columns represent the beginning of the period and the end of the period
  - For a row, the beginning value is included in the period and end value is not included in the period
  - SYSTEM_TIME row begin column must be defined as:
    - TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN
  - SYSTEM_TIME row end column must be defined as:
    - TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END
  - BUSINESS_TIME row begin and row end columns can be defined as either:
    - TIMESTAMP(6) NOT NULL
    - DATE NOT NULL
  - When a period is defined for a table, DB2 generates a check constraint in which the end column value must be greater than the begin column value

Defining System Versioning on a Table

- System versioning is implemented by altering an existing or creating a table with a SYSTEM_TIME period, a history table, and defining the versioning relationship
- System period temporal table must have:
  - SYSTEM_TIME is defined as two TIMESTAMP(12) NOT NULL columns
  - First column defines the row begin time and the other is the row end time
  - A column defined as TIMESTAMP(12) for the transaction start id
  - SYSTEM_TIME period defined on the row begin and row end columns
    - example of period definition: PERIOD SYSTEM_TIME(col1, col2)
- History table must have:
  - Same number of columns as the system period temporal table
  - All columns must have the same corresponding names, data type, null attribute, ccsid, subtype, hidden attribute and fieldproc as the system period temporal table

Add System Versioning to a Table

- After the two tables are appropriately defined:
  - ALTER TABLE table-name ADD VERSIONING is specified on the table that is to be versioned, not the history table
- To query historical data, the table-reference of the FROM clause is extended to request historical data
  - DB2 rewrites the user's query to include data from the history table with a UNION ALL operator
- New FROM SYSTEM_TIME clauses:
  - table-name FOR SYSTEM_TIME AS OF timestamp-expression
  - table-name FOR SYSTEM_TIME FROM timestamp-expression1 TO timestamp-expression2
    - Note: the second timestamp-expression is not inclusive
  - table-name FOR SYSTEM_TIME BETWEEN timestamp-expression1 AND timestamp-expression2
    - Note: the second timestamp-expression is inclusive

SYSTEM_TIME Period Example (DDL)

```
CREATE TABLE      policy_info
(policy_id  CHAR(4)              NOT NULL,
coverage    INT                  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));

CREATE TABLE      hist_policy_info
(policy_id  CHAR(4)              NOT NULL,
coverage    INT                  NOT NULL,
sys_start   TIMESTAMP(12)        NOT NULL,
sys_end     TIMESTAMP (12)       NOT NULL,
create_id   TIMESTAMP(12));

ALTER TABLE policy_info
ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

SYSTEM_TIME Period INSERT Example

At timestamp '2009-01-05-12:12:12.001122000000',

INSERT INTO policy_info (policy_id, coverage)
VALUES('A123', 12000);

Hist_policy_info

Policy ID	Coverage	Sys_start	Sys_end

Policy_info

Policy ID	Coverage	Sys_start	Sys_end
A123	12000	2009-01-05-12.12.12.001122000000	9999-12-31-24.00.00.000000000000

SYSTEM_TIME Period UPDATE Example

At timestamp '2009-01-09-12:12:12.012345000000',

UPDATE policy_info SET coverage = 15000
WHERE policy_id = 'A123';

Hist_policy_info

Policy ID	Coverage	Sys_start	Sys_end
A123	12000	2009-01-05-12.12.12.001122000000	2009-01-09-12.12.12.012345000000

Policy_info

Policy ID	Coverage	Sys_start	Sys_end
A123	15000	2009-01-09-12.12.12.012345000000	9999-12-31-24.00.00.000000000000

SYSTEM_TIME Period SELECT Example

```
SELECT policy_id, coverage FROM policy_info
FOR SYSTEM_TIME AS OF '2009-01-08-00:00:00.000000000000';
```

returns a row of
('A123', 12000)

Hist_policy_info

Policy ID	Coverage	Sys_start	Sys_end
A123	12000	2009-01-05-12.12.12.001122	2009-01-09-12.12.12.012345000000

Policy_info

Policy ID	Coverage	Sys_start	Sys_end
A123	15000	2009-01-09-12.12.12.012345000000	9999-12-31-24.00.00.000000000000

Adding a Business Period to a Table

- To create a temporal table with BUSINESS_TIME period, the table needs to be defined with:
  - A begin and end column defined as TIMESTAMP(6) NOT NULL or DATE NOT NULL
  - BUSINESS_TIME time defined on the two above columns
    - example of period specification: PERIOD BUSINESS_TIME(col1, col2)
- The temporal table with BUSINESS_TIME period can also be defined to have a unique key that is unique for a period of time. For example:
  - UNIQUE (col1, BUSINESS_TIME WITHOUT OVERLAPS)
  - PRIMARY KEY (col1,col2, BUSINESS_TIME WITHOUT OVERLAPS)
- BUSINESS_TIME WITHOUT OVERLAPS clause specified on CREATE, ALTER, or CREATE UNIQUE INDEX statements.
  - BUSINESS_TIME WITHOUT OVERLAPS must be the last expression specified
  - It will add, in ascending order, the end column and begin column of the period BUSINESS_TIME to the key and have special support to enforce that there are no overlaps in time given the rest of the key expressions.
- The enforcement of uniqueness over a period of time is the important functionality delivered with temporal tables with BUSINESS_TIME periods

BUSINESS_TIME Period

To query a temporal table with a BUSINESS_TIME period, table-reference of the FROM clause is extended with a specification on the BUSINESS_TIME period:

- Business period is a date or timestamp expression depending on period definition
- table-name FOR BUSINESS_TIME AS OF expression
- table-name FOR BUSINESS_TIME FROM expression1 TO expression2
  - note that the second expression is not inclusive
- table-name FOR BUSINESS_TIME BETWEEN expression1 AND expression2
  - note that the second expression is inclusive

BUSINESS_TIME UPDATE and DELETE Semantics

- Temporal tables with BUSINESS_TIME period, the UPDATE and DELETE statement have been enhanced to allow updating and deleting based upon a period of time
- New clause is specified as follows:
  - FOR PORTION OF BUSINESS_TIME FROM expression1 TO expression2
  - expression is a date or timestamp expression depending on period definition
- New update or delete clause updates/deletes rows as usual if the period for the row is totally contained in the period specified in the FROM and TO clause
- When a row is not totally contained within the period, only the part of the row that is contained between the FROM and TO values are updated or deleted
  - This may cause additional inserts of rows into the table as the original row is split into multiple rows to reflect the old values for the period that are not affected by the update or delete

BUSINESS_TIME Period Example

```
CREATE TABLE      policy_info
(policy_id      CHAR(4)      NOT NULL,
coverage        INT          NOT NULL,
bus_start       DATE         NOT NULL,
bus_end         DATE         NOT NULL,
PERIOD          BUSINESS_TIME(bus_start, bus_end);

CREATE UNIQUE INDEX ix_policy
ON policy_info (policy_id, BUSINESS_TIME WITHOUT
  OVERLAPS);
```

BUSINESS_TIME Period INSERT Example

```
INSERT INTO policy_info VALUES
('A123', 12000, '01-01-2008', '07-01-2008');
```

Policy_info

DB Action	Policy ID	Coverage	Bus_strt	Bus_end
Insert	**A123**	**12000**	**01-01-2008**	**07-01-2008**

BUSINESS_TIME Period INSERT Example

```
INSERT INTO policy_info VALUES
('A123',12000, '07-01-2008', '01-01-2009');
```

Policy_info

DB Action	Policy ID	Coverage	Bus_strt	Bus_end
	A123	**12000**	**01-01-2008**	**07-01-2008**
Insert	**A123**	**12000**	**07-01-2008**	**01-01-2009**

BUSINESS_TIME Period INSERT Example

INSERT INTO policy_info VALUES ('A123',14000, '06-01-2008','08-01-2008');

ERROR, policy is not unique over time.

Policy_info

DB Action	Policy ID	Coverage	Bus_strt	Bus_end
	A123	**12000**	**01-01-2008**	**07-01-2008**
	A123	**12000**	**07-01-2008**	**01-01-2009**

BUSINESS_TIME Period UPDATE Example

```
UPDATE policy_info
FOR PORTION OF BUSINESS_TIME
FROM '06-01-2008' TO '08-01-2008'
SET coverage = 14000
WHERE policy_id = 'A123';
```

Policy_info

DB Action	Policy ID	Coverage	Bus_strt	Bus_end
Insert	**A123**	**12000**	**01-01-2008**	**06-01-2008**
Update	**A123**	**14000**	**06-01-2008**	**07-01-2008**
Update	**A123**	**14000**	**07-01-2008**	**08-01-2008**
Insert	**A123**	**12000**	**08-01-2008**	**01-01-2009**

BUSINESS_TIME Period DELETE Example

```
DELETE FROM policy_info
FOR PORTION OF BUSINESS_TIME
FROM '06-15-2008' TO '08-15-2008'
WHERE policy_id = 'A123';
```

Policy_info

DB Action	Policy ID	Coverage	Bus_strt	Bus_end
	A123	12000	01-01-2008	06-01-2008
Delete				
Insert	A123	14000	06-01-2008	06-15-2008
Delete				
Delete				
Insert	A123	12000	08-15-2008	01-01-2009

BUSINESS_TIME Period Example (DML)…

SELECT policy_id, coverage FROM policy_info
FOR BUSINESS_TIME AS OF '06-01-2008';

returns a row of
('A123', 14000)

Policy_info

DB Action	Policy ID	Coverage	Bus_strt	Bus_end
	A123	**12000**	**01-01-2008**	**06-01-2008**
Select	**A123**	**14000**	**06-01-2008**	**06-15-2008**
	A123	**12000**	**08-15-2008**	**01-01-2009**

Satisfy Your Auditor: Temporal Table Benefits

- SYSTEM_TIME period
  - Provides database managed row begin and row end timestamp maintenance providing non-overlapping time periods with no gaps in time
  - Provides automatic management of data movement from system period temporal table to history table
  - Provides automatic rewrite of queries on system period temporal table to include a UNION ALL to history table
  - Minimizes performance impact of temporal support when no rows requested from a historical perspective
- BUSINESS_TIME period
  - Provides a clause for a unique index to prevent overlapping time periods for a business key
  - Provides database management of the temporal data when an UPDATE or a DELETE statement is executed with a FOR PORTION OF BUSINESS_TIME clause that does not exactly align with respect to time for existing rows
- BITEMPORAL table
  - Combines all of the features provided for SYSTEM_TIME period and BUSINESS_TIME period

Help Satisfy Your Auditors using DB2 10

- More granular authorities
- Improved SQL statement level auditing
- New SQL row and column access controls
- New temporal data support
  - System period supporting data versioning
  - Business period supporting business time

Data Management Communities for DB2

- IDUG – the worldwide community of DB2 users
  - Membership is FREE – join today! www.idug.org
- Data Management Community – share and interact with peers around the world
  - www.ibm.com/software/data/management/community.html
- developerWorks – IBM's resource for developers & IT professionals
  - www.ibm.com/developerworks/data/products/db2
- Information Champions – recognizes individuals who have made the most outstanding contributions to the Information Management community
  - www.ibm.com/software/data/champion