Information Management

IBM

DB2 V10.1
Row and Column Access Control

George Baklarz, IBM

August 15, 2013

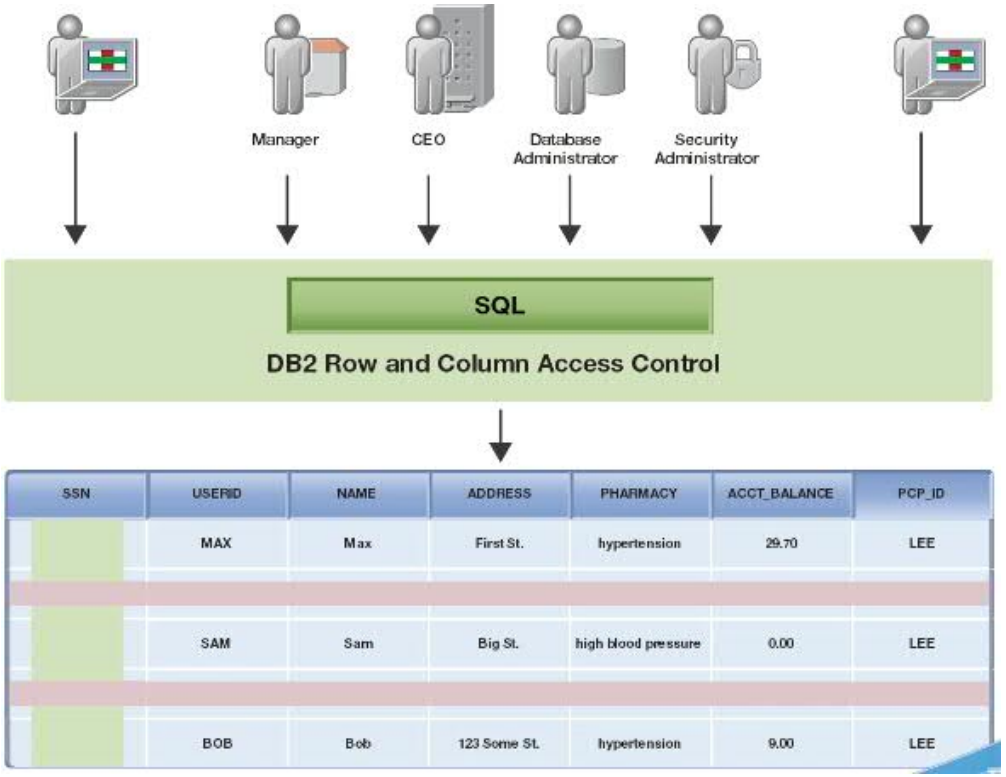
Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. 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.

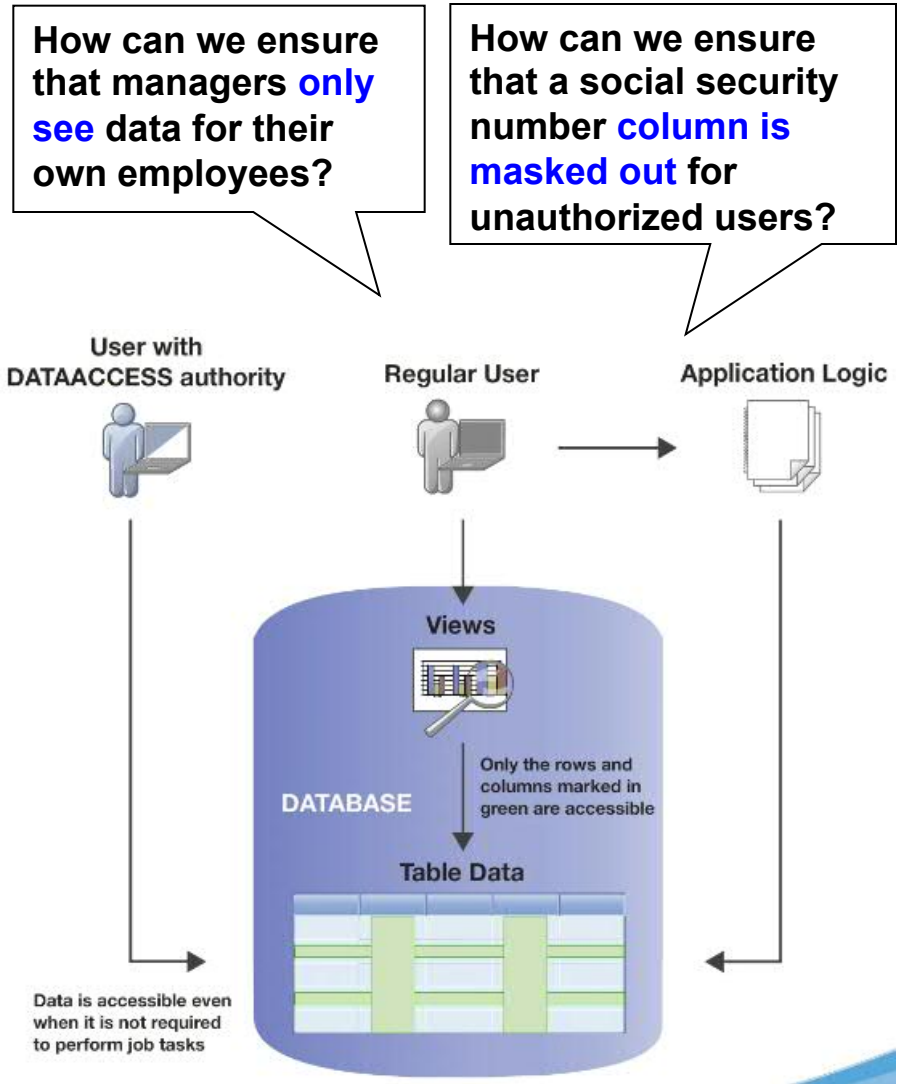
Row and Column Access Control

- **Additional layer of data security introduced in DB2 V10**
- **Complementary to table level authorization**
- **Allows access only to subset of data useful for job task**
- **Controls access to a table at the row, column, or both**
- **Two sets of rules**
  - Permissions for rows
  - Masks for columns



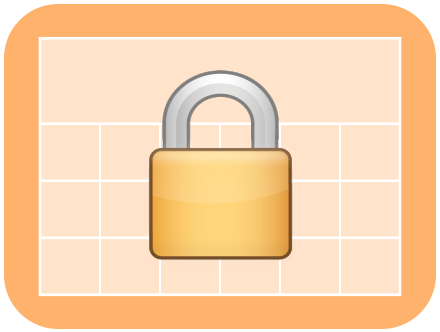
Why Use Row and Column Access Control?

- **Currently, data access is restricted via views or application logic**
- **Users with direct access to databases can bypass these layers**
  - Example: Users with `DATAACCESS` authority can still view all data
- **DB2 V10 provides a new way to control data access at row/column level**
  - Set up rich security policies
  - Prevents administrators with `DATAACCESS` authority from accessing all data in a database
  - No dependency on application logic
  - Facilitates table level multi-tenancy



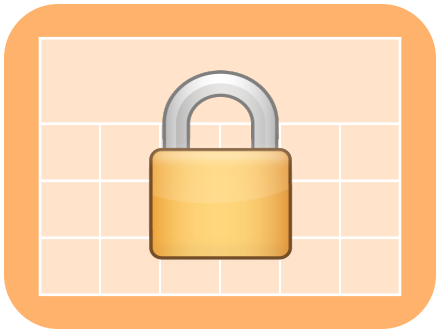
How is This Different from LBAC?

- **LBAC is a fixed label security model designed for the US government**
  - Hierarchical access scenarios
    - Great for large companies with well defined data and user classifications
    - Suited for such applications as those intelligence and defense communities
- **RCAC is a general purpose security model**
  - No data or user classification required
  - Best suited for commercial customers

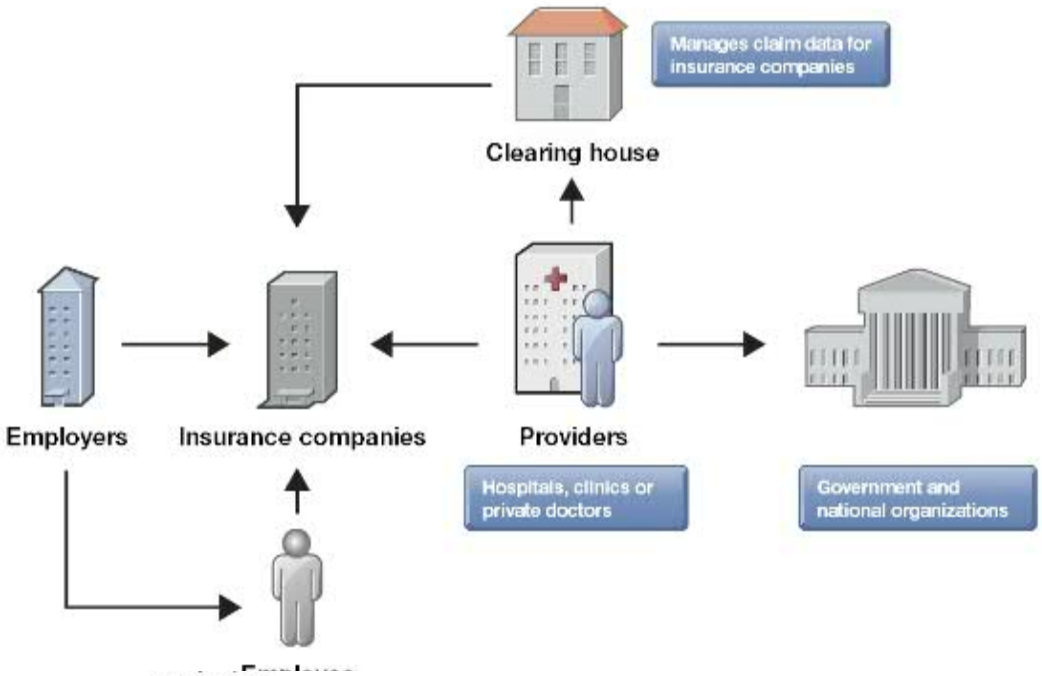


How is This Different from LBAC? (cont.)

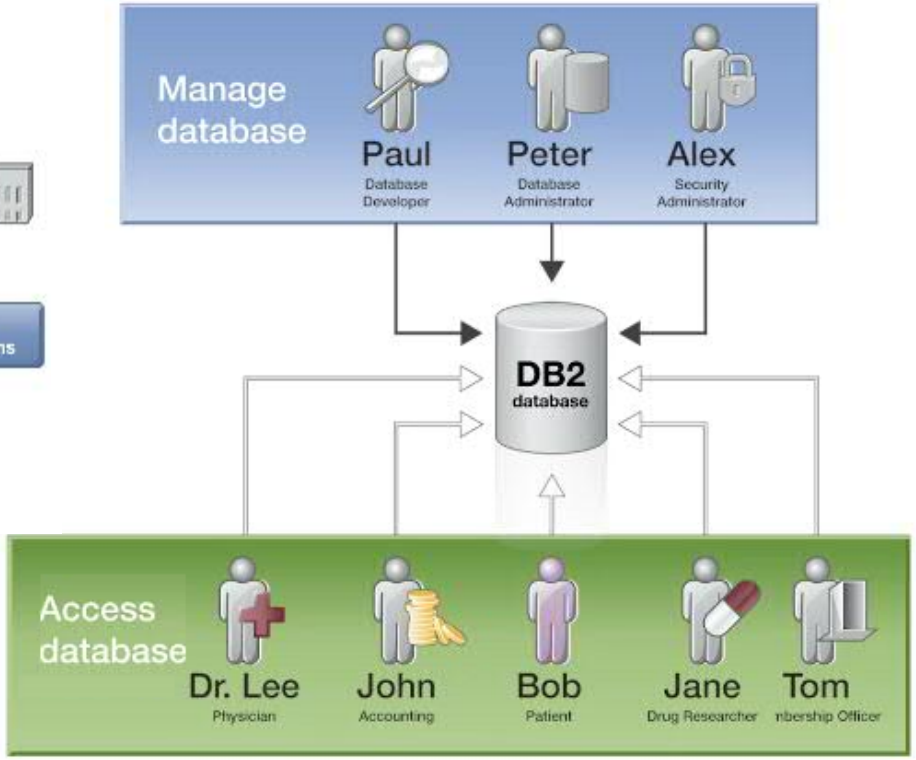
- **Why is RCAC more suited for commercial customers?**
  - Simpler mechanism
    - LBAC is powerful, but has a complex, lengthy implementation
  - No requirement to classify data and users
    - Can use existing roles and groups
  - Provides column masking (LBAC does not)
  - Transparent to the client application
  - Pushes security from application to database, giving performance advantages
  - Robust data privacy for supporting multi-tenancy
  - Very easy for use in compliance



Scenario: Health Care Insurance Industry



- ✓ Ensure privacy of Protected Health Information
- ✓ Protect confidentiality of electronically transmitted data
- ✓ Control data access to privileged users only
- ✓ Log access of protected data



Create Permission

- **To create a permission governing access to rows**
  1) `CREATE` the permission with access rule defined by search condition
     - Choose to enforce for all DML or simply select
  2) `ENABLE` or `DISABLE` the permission
     - If enabled, this access rule will be implemented when row access control is `ACTIVATEd` for the affected table
  3) `ALTER` table to activate row access control

```
CREATE PERMISSION p_name ON table/view FOR ROWS
WHERE search condition ENFORCED FOR ALL ACCESS {disable/enable};

ALTER TABLE/VIEW table/view ACTIVATE ROW ACCESS CONTROL;
```

`WHERE` clause

`ACTIVATE` the row access control

Determines if permission will be `ENABLED` when access control is `ACTIVATED` for table

Scenario: Create Permission

- **Scenario has the following permissions attached**
  1. – Patients
     - Can only access their own data
  2. – Physicians
     - Can only access their own patients' data
  3. – Membership officers, Accounting, Drug researchers
     - Can access all data
  – Nobody else sees any data

Scenario: Create Permission (cont.)

1
2
3

```
CREATE PERMISSION row_access ON patient
FOR ROWS WHERE
(
   verify_role_for_user(SESSION_USER,'PATIENT') = 1
   AND patient.userid = SESSION_USER
)
OR
(
   verify_role_for_user(SESSION_USER,'PCP') = 1
   AND patient.pcp_id = SESSION_USER
)
OR
(
   verify_role_for_user(SESSION_USER,'MEMBERSHIP') = 1 OR
   verify_role_for_user(SESSION_USER,'ACCOUNTING') = 1 OR
   verify_role_for_user(SESSION_USER,'DRUG_RESEARCH') = 1
)
ENFORCED FOR ALL ACCESS
ENABLE;


ALTER TABLE patient ACTIVATE ROW ACCESS CONTROL;
```

Scenario: Update Table with Permissions


```
UPDATE patient SET pharmacy = 'codeine' WHERE name = 'Sam'
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
123 551 234	MAX	Max	First St.	hypertension	89.70	LEE
123 589 812	MIKE	Mike	Long St.	diabetics	8.30	JAMES
123 119 856	SAM	Sam	Big St.	codeine	12.50	LEE
123 191 454	DOUG	Doug	Good St.	influenza	7.68	JAMES
123 456 789	BOB	Bob	123 Some St.	hypertension	9.00	LEE

Successful `UPDATE` statement!

Scenario: Update Table with Permissions (cont.)


```
UPDATE patient SET pharmacy = 'codeine' WHERE name = 'Doug'
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
123 551 234	MAX	Max	First St.	hypertension	89.70	LEE
123 589 812	MIKE	Mike	Long St.	diabetics	8.30	JAMES
123 119 856	SAM	Sam	Big St.	codeine	12.50	LEE
123 191 454	DOUG	Doug	Good St.	influenza	7.68	JAMES
123 456 789	BOB	Bob	123 Some St.	hypertension	9.00	LEE

- **Unsuccessful `UPDATE` statement**

  `No row was found for FETCH, UPDATE or DELETE; or the result of a query is an empty table.`

- **If you cannot view a row, you cannot update it either**

Scenario: Select from Table with Permission


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
123 551 234	MAX	Max	First St.	hypertension	89.70	LEE
123 119 856	SAM	Sam	Big St.	codeine	12.50	LEE
123 456 789	BOB	Bob	123 Some St.	hypertension	9.00	LEE

- **Row Access Control**
  - Doctors can only see the data of their own patients

Scenario: Select from Table with Permission (cont.)


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
123 551 234	MAX	Max	First St.	hypertension	89.70	LEE
123 589 812	MIKE	Mike	Long St.	diabetics	8.30	JAMES
123 119 856	SAM	Sam	Big St.	codeine	12.50	LEE
123 191 454	DOUG	Doug	Good St.	influenza	7.68	JAMES
123 456 789	BOB	Bob	123 Some St.	hypertension	9.00	LEE

- **Row Access Control**
  - Accounting, drug researchers and membership officers can see all data

Scenario: Select from Table with Permission (cont.)

- Row Access Control -> Patients can only see their own data


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
123 456 789	BOB	Bob	123 Some St.	hypertension	9.00	LEE

- Row Access Control -> Database Administrators cannot see any data


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID

Scenario: Create Column Mask

- **Scenario has the following permissions attached**
  - Account balance column
    - Accounting can see the account balance
    - Everyone else sees `0.00`
  - SIN number column
    - Patients can see full SIN number
    - Everyone else sees `'xxx xxx '` + last three digits of SIN

1

2

Create Column Mask

- **To create a mask for a column**
  1) `CREATE` the mask with visibility of column value determined by case expression
  2) `ENABLE` or `DISABLE` the permission, determining if this access rule will be implemented when column access control is enabled for the affected table
  3) `ALTER` table to `ACTIVATE` row access control

```
CREATE MASK m_name on t_name FOR COLUMN c_name RETURN
case-expression {disable/enable}

ALTER TABLE/VIEW table/view ACTIVATE COLUMN ACCESS CONTROL;
```

Result of case expression is returned in substitute of column value

Determines if mask will be enabled when access control is `ACTIVATE`d for table

`ACTIVATE` column access control

Scenario: Create Column Mask (cont.)

1

```
CREATE MASK acct_balance_mask ON patient FOR
COLUMN acct_balance RETURN
   CASE
      WHEN verify_role_for_user(SESSION_USER,
         'ACCOUNTING') = 1
         THEN acct_balance
      ELSE 0.00
   END
ENABLE;
```

2

```
CREATE MASK sin_mask ON patient FOR
COLUMN sin RETURN
   CASE
      WHEN verify_role_for_user(SESSION_USER,
         'PATIENT') = 1
         THEN sin
      ELSE
         'XXX XXX ' || SUBSTR(sin,8,3)
   END
ENABLE;

ALTER TABLE patient ACTIVATE COLUMN ACCESS CONTROL;
```

Scenario: Select from Table with Mask


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
XXX XXX 234	MAX	Max	First St.	hypertension	89.70	LEE
XXX XXX 812	MIKE	Mike	Long St.	diabetics	8.30	JAMES
XXX XXX 856	SAM	Sam	Big St.	codeine	12.50	LEE
XXX XXX 454	DOUG	Doug	Good St.	influenza	7.68	JAMES
XXX XXX 789	BOB	Bob	123 Some St.	hypertension	9.00	LEE

- **Column Access Control**
  - Accountants can see account balances
  - Accountants cannot see SIN numbers
- **Row Access Control**
  - Accountants can see all rows

Scenario: Select from Table with Mask (cont.)


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
XXX XXX 234	MAX	Max	First St.	hypertension	0.00	LEE
XXX XXX 812	MIKE	Mike	Long St.	diabetics	0.00	JAMES
XXX XXX 856	SAM	Sam	Big St.	codeine	0.00	LEE
XXX XXX 454	DOUG	Doug	Good St.	influenza	0.00	JAMES
XXX XXX 789	BOB	Bob	123 Some St.	hypertension	0.00	LEE

- **Column Access Control**
  - Drug researchers cannot see account balances
  - Drug researchers cannot see SIN numbers
- **Row Access Control**
  - Drug researchers can see all rows

Scenario: Select from Table with Mask (cont.)


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
XXX XXX 234	MAX	Max	First St.	hypertension	0.00	LEE
XXX XXX 856	SAM	Sam	Big St.	codeine	0.00	LEE
XXX XXX 789	BOB	Bob	123 Some St.	hypertension	0.00	LEE

- **Column Access Control**
  - Doctors cannot see account balances
  - Doctors cannot see SIN numbers
- **Row Access Control**
  - Doctors can only see the rows of their own patients

Scenario: Select from Table with Mask (cont.)


```
SELECT * FROM patient
```

SIN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
123 456 789	BOB	Bob	123 Some St.	hypertension	0.00	LEE

- **Column Access Control**
  - Patients cannot see account balances
  - Patients can see SIN numbers
- **Row Access Control**
  - Patients can only see their own data

Using Views with RCAC-Protected Tables

- **Views can be created on RCAC-protected tables**
  - When querying the view, data is returned based on RCAC rules defined on base table

```
CREATE VIEW patient_info_view AS
   SELECT p.sin, p.name, c.choice
   FROM patient p, patientchoice c
   WHERE p.sin = c.sin
         AND c.choice = 'drug-research'
         AND c.value = 'opt-in';
```


```
SELECT * FROM patient_info_view;
```

SIN	NAME	CHOICE
XXX XXX 856	Sam	drug-research
XXX XXX 789	Bob	drug-research

Using UDFs and Triggers with RCAC-Protected Tables

- UDFs must be defined as SECURED when referenced from within row and column access control definitions

```
ALTER FUNCTION ACCBALDISPLAY SECURED;
```

```
CREATE MASK EXAMPLEHMO.ACCT_BALANCE_MASK ON PATIENT FOR
COLUMN ACCT_BALANCE RETURN
CASE WHEN VERIFY_ROLE_FOR_USER(SESSION_USER,'ACCOUNTING') = 1
   THEN ACCBALDISPLAY(ACCT_BALANCE)
   ELSE 0.00
END
ENABLE;
```

- UDFs invoked on columns protected with a column mask must be defined as SECURED. For instance, the SELECT statement below will fail unless CALC is defined as a secure UDF

```
SELECT CALC(ACC_BALANCE) FROM PATIENT;
```

- Triggers must be defined as SECURED if the subject table is protected with row or column access control

```
ALTER TRIGGER T_LOG_CHANGES SECURED;
```

SQL Statements for Managing RCAC rules

```
CREATE [OR REPLACE] PERMISSION p_name ON t_name FOR ROWS
WHERE search condition ENFORCED FOR ALL ACCESS DISABLE/
ENABLE
```

```
CREATE [OR REPLACE] MASK m_name ON t_name FOR COLUMN
c_name RETURN case expression DISABLE/ENABLE
```

```
ALTER MASK/PERMISSION c_name/p_name DISABLE/ENABLE
```

```
DROP MASK/PERMISSION c_name/p_name
```

```
ALTER TABLE t_name ACTIVATE/DEACTIVATE ROW/COLUMN ACCESS
CONTROL
```

New Built-in Scalar Functions

- **`verify_role_for_user (user, role1, role2, …)`**
  - The result is 1 if any of the roles associated with the user are in list of role1, role2, etc.
  - Else 0
- **`verify_group_for_user (user, group1, group2, …)`**
  - The result is 1 if any of the groups associated with the user are in list of group1, group2, etc.
  - Else 0
- **`verify_trusted_context_role_for_user (user, role1, role2, …)`**
  - The result is 1 if when the role acquired through a trusted connection is in (or contains) any of the roles in the list of role1, role2, …
  - Else 0

Restrictions and Considerations

- **You cannot create a mask on a column which**
  - Is an `XML` object
  - Is a `LOB` column or a distinct type column that is based on a `LOB`
  - Is a column referenced in an expression that defines a generated column
- **`UDFs` and `TRIGGERs` must be altered with `SECURE` keyword**
  - Compromise between security vs. integrity
- **Automatic Data Movement**
  - Row permissions are automatically activated on these target tables
    - `MQT`, `History` Tables for `Temporal` tables, and detached table partitions for range-partitioned tables
- **`db2look` can extract row permission definitions in order to mimic elsewhere**
- **`EXPLAIN` facility shows access plans with `RCAC` in place**
  - Can override with `NORCAC` option

Summary of Row and Column Access Control

- **Allows access only to subset of data useful for job task**
  - Leverages existing groups and roles
- **Same output regardless of access method**
  - Data Studio, views, applications, etc.
- **Data-centric and transparent to client application**
- **Less powerful mechanism than LBAC, but simpler to implement**
- **Ideal for**
  - Commercial applications
  - Function-specific access control (vs. hierarchical)
  - Use in compliance
- **`SECADM` is the sole manager of security policy**
- **Two sets of rules**
  - Row access is restricted via permissions
  - Column access is restricted via masks