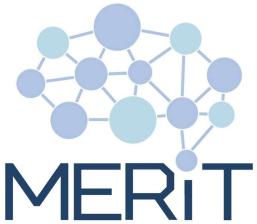
Co-funded by
the European Union

MERIT

Database Security

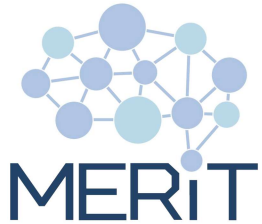
User Management and Access Control

Topics



- ✔ Methods for database security
- ✔ Discretionary access control method
- ✔ Mandatory access control
- ✔ Role base access control for multilevel security.
- ✔ Use of views in security enforcement.

METHODS FOR DATABASE SECURITY



Following are different techniques used for database security.

- ✔ Discretionary Access Control Method
- ✔ Mandatory Access Control Method
- ✔ Statistical Database Security
- ✔ Different Data Encryption Techniques

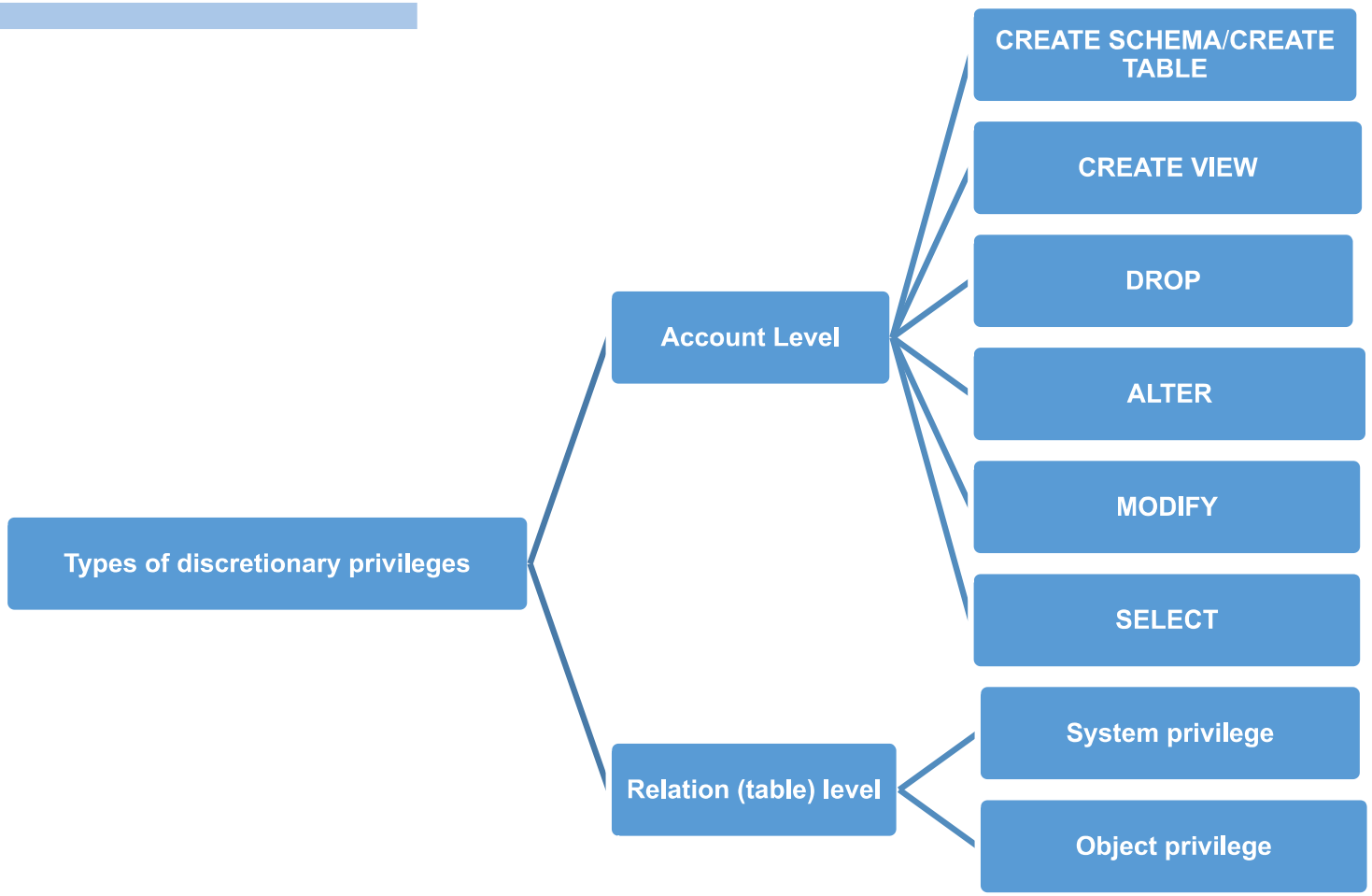
Discretionary access control method

Access to database from unauthorized users is secured by this methods.

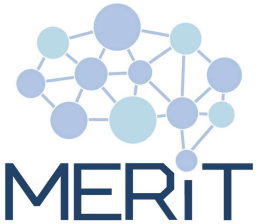
Granting and revoking privileges to database users at different levels such as on files, records, fields and on any specified mode such as read, insert, delete and update

A privilege is a right to execute a particular type of SQL statement or to access another user's object. e.g reading, writing, updating and deleting from database.

Discretionary access control method II

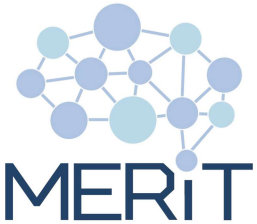


Account level privileges



- **CREATE SCHEMA/CREATE TABLE**: This privilege is used to create schema or base relation
- **CREATE VIEW**: This privilege is used to create view. This will be used for relation level and virtual –view relations.
- **DROP**: This privilege is used to delete relation or view.
- **ALTER**: This privilege is used to apply schema changes to relations such as adding/removing attribute column to a table.
- **MODIFY**: This privilege is used to insert, delete or update tuples in a database.
- **SELECT**: This privilege is use to retrieve information from database by using select query.

GRANTING AND REVOKING PRIVILEGES

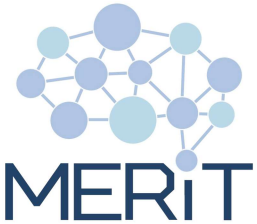


- These privileges follow authorization model and access matrix or authorization matrix model.

Access matrix model (M):

- Rows: Represents subjects(users/accounts/programs)
- Columns: Objects(relations/records/columns/views/operations)
- M(i,j): This position is the matrix represents the types of privileges (read, write, update) that subject i holds object j.

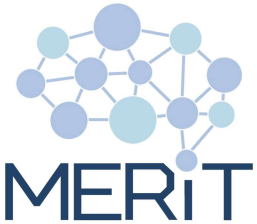
Example of privileges of relation



in SQL following types of privileges are granted to each relation (R)

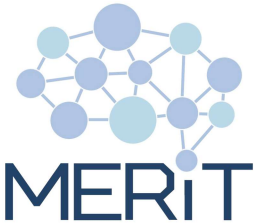
- SELECT: This is read privileges on relation (R). Data from relation is retrieved using this privilege.
- MODIFY: This privilege modify tuples from relation (R). This privilege gives various commands like UPDATE (updating attributes), DELETE (deleting attributes) and INSERT (inserting attributes) on relation (R).
- REFERENCES: This privilege modify the reference relation (R) using integrity constraints. It can be restricted for specific relation (R).

Relation (table) level privileges



- System privilege: It is right to perform particular action on a particular type of object. E.g. to create table, to delete rows of table using CREATE, ALTER and DROP command.
- Object privilege: It is right to perform particular action on particular type of table, relation, attribute, view, sequence, procedure, function or package. E.g. to select, delete , insert data into table using SELECT, INSERT, DELETE command.

GRANT COMMAND



Syntax:

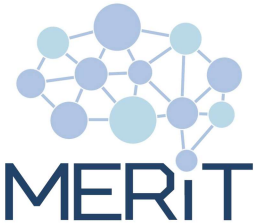
GRANT privilege_name

ON < tablename| view name>

TO {user list| role name}

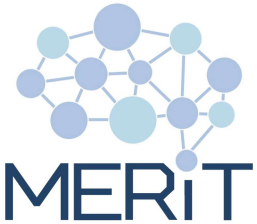
[WITH GRANT OPTION];

GRANT COMMAND II



- Privilege name: access right like SELECT, INSERT, UPDATE, DELETE etc. to be granted.
- Table name: relation name view name: specifies view name
- User list: specifies list of users to which privilege is to be granted
- Role name: specifies user role in database system
- WITH GRANT OPTION: specifies a user to grant access rights to other user.

Example:



GRANT SELECT, DELETE, UPDATE //access rights with commands

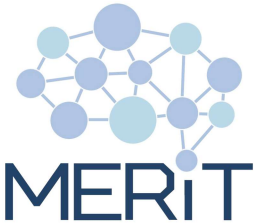
ON EMPLOYEE // table name

TO RAJESH //user name

WITH GRANT OPTION; // allows user to grant access rights to other users.

Meaning: RAJESH is authorized to perform SELECT, DELETE, UPDATE operation on EMPLOYEE table and grant those privileges to other users of employee table.

REVOKE COMMAND



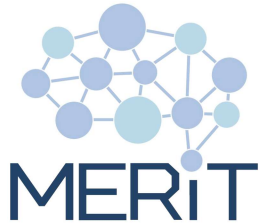
Syntax

REVOKE privilege_name

ON < tablename| view name>

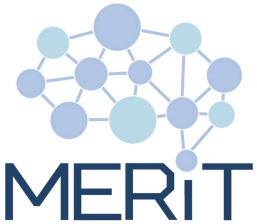
FROM {user list| role name} [restrict |cascade];

REVOKE COMMAND II



- Privilege name: access right like SELECT, INSERT, UPDATE, DELETE etc. to be removed.
- Table name: relation name view name: specifies view name
- User list: specifies list of users to which privilege is to be granted
- Role name: specifies user role in database system
- Restrict: the privilege will be removed only from specified user and not from other users to whom the privilege is granted by specified user.
- Cascade: The privilege will be removed from user and from other dependents users also.

Example



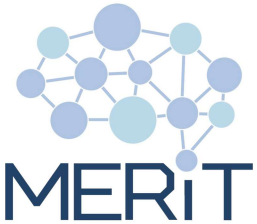
REVOKE DELETE, UPDATE (emp-sal) //access rights with commands to be removed

ON EMPLOYEE // table name

FROM RAJESH //user name

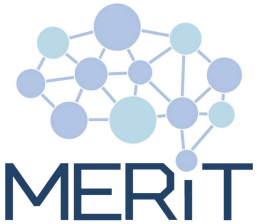
Explanation: DELETE and UPDATE rights on emp-sal will be removed from user rajesh on table EMPLOYEE.

AUDIT TRAILS



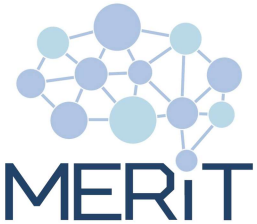
Audit trail is a log of all changes (insert/delete/update) performed on database along with the user information and time. This is used to track fraud in database as it gives security to the database.

A typical audit trail contains following entries



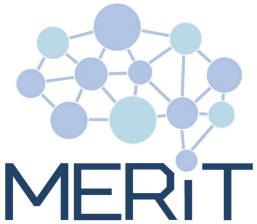
- Request (source text)
- Terminal (from which operation was performed)
- User(who performed the operation)
- Date
- Time
- Tuples (on which tuples of relation R)
- Attributes (of relation R)
- Old value (of tuple/ attribute/ relation R)
- New value (of tuple/ attribute/ relation R)

Advantages of DAC



- **User-friendly:** Users can manage their data and quickly access data of other users.
- **Flexible:** Users can configure data access parameters without administrators.
- **Easy to maintain:** Adding new objects and users doesn't take much time for the administrator.
- **Granular:** Users can configure access parameters for each piece of data.

Disadvantages of DAC



- **Low level of data protection —** DAC can’t ensure reliable security because users can share their data however they like.
- **Obscure —** There’s no centralized access management, so in order to find out access parameters, you have to check each ACL

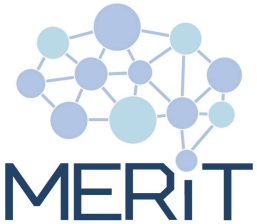
Mandatory Access Control

- Mandatory Access Control is a method of limiting access to resources based on the sensitivity of the information that the resources contains and the authorization of the user to access information with that level of sensitivity.
- This model of access control where the operating system provides users with access based on data confidentiality and user clearance levels. In this model, access is granted on a need know basis.
- Before gaining the access you have to know the need of information
- It is most secure because this model is non- discretionary control model and implemented on zero trust principle.

Advantages of MAC

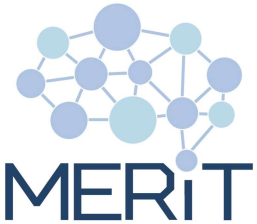
- **High level of data protection:** An administrator defines access to objects, and users can't edit that access.
- **Granular:** An administrator sets user access rights and object access parameters manually.
- **Immune to Trojan Horse attacks:** Users can't declassify data or share access to classified data.

Disadvantages of MAC



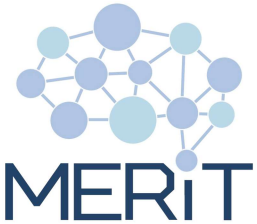
- **Maintainability:** Manual configuration of security levels and clearances requires constant attention from administrators.
- **Scalability:** MAC doesn't scale automatically.
- **Not user-friendly:** Users have to request access to each new piece of data; they can't configure access parameters for their own data.

Compare and contrast DAC and MAC



DAC	MAC
A type of access control on which the owner of a resource restricts access to the resource based on the identity of the users.	A type of access control that restricts the access to the resources based on the clearance of the subjects.
Stands of Discretionary access Control	Stands for Mandatory Access Control
Resource owner determines who can access and what privileges they have	Provides access to the users depending on the clearance level of users. Access is determined by the system
More flexible	Less flexible
Not as secure as MAC	More secure
Easier to implement	Comparatively less easier to implement

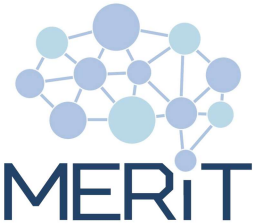
Role Based Access Control (RBAC) For Multilevel Security



Roles are collection of privileges or access rights that are combined in a centralized unit which manages users or objects of a database.

There are two types of roles namely application role and user role.
Application role: It is used for granting all the necessary privileges to run a given application.
User role: It is used for creation of database user groups with common privilege requirements.

RBAC



- ✔ Roles can be created using CREATE ROLE and deleted using DROP ROLE command. GRANT and REVOKE commands under Discretionary Access control are used to assign and revoke privileges from roles.
- ✔ RABC is alternative to Discretionary Access control and Mandatory Access control.

RBAC

- CREATION OF ROLES:

Syntax:

- CREATE ROLE rolename [IDENTIFIED BY password]

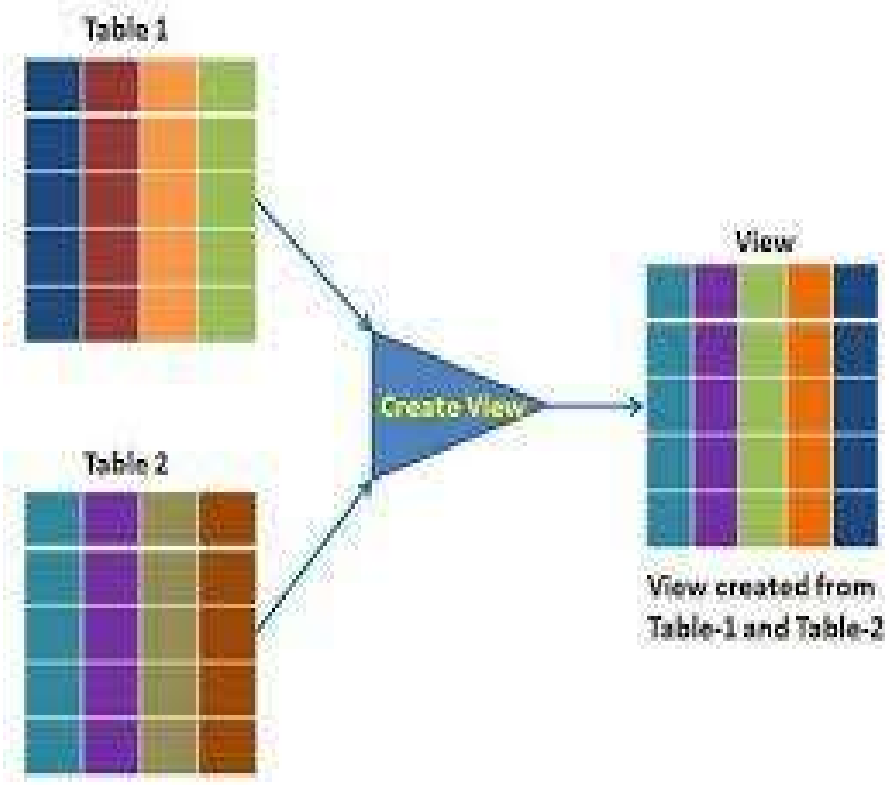
Example to create role of “developer” with password dev@123

CREATE ROLE DEVELOPER [IDENTIFIED BY dev@123]

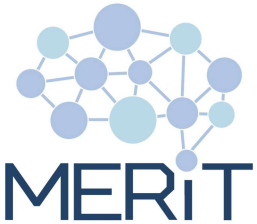
DROPPING ROLES:
Syntax
DROP ROLE role_name;
Example DROP ROLE DEVELOPER;

USE OF VIEWS IN SECURITY ENFORCEMENT

A **view** is the result set of a *stored* query on the data, which the database users can query just as they would in a persistent database collection object. Views can represent a subset of the data contained in a table.

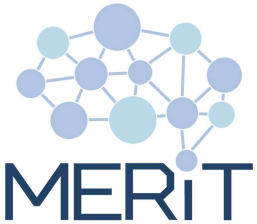


Advantages of views



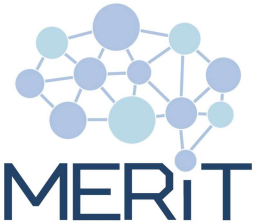
- ✔ A view can limit the degree of exposure of the underlying tables to the outer world.
- ✔ Security: Each user can be given permission to access the database only through a small set of views that contain the specific data the user is authorized to see, thus restricting the user's access to stored data
- ✔ Query simplicity: A view can draw data from several different tables and present it as a single table, turning multi-table queries into single-table queries against the view.

Advantages of views



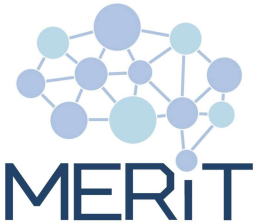
- ✔ **Structural simplicity: Views can give a user a "personalized" view of the database structure, presenting the database as a set of virtual tables that make sense for that user.**
- ✔ **Consistency: A view can present a consistent, unchanged image of the structure of the database, even if the underlying source tables are split, restructured, or renamed.**
- ✔ **Data Integrity: If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets the specified integrity constraints.**
- ✔ **Logical data independence: View can make the application and database tables to a certain extent independent. If there is no view, the application must be based on a table.**

Disadvantages of views



- When a table is dropped, associated view become irrelevant.
- Since the view is created when a query requesting data from view is triggered, it’s a bit slow.
- When views are created for large tables, it occupies more memory.

Creation of views



Syntax:

- CREATE [TEMP| TEMPORARY] VIEW view_name AS
- SELECT column1, column2…
- FROM table_name
- WHERE [condition];
- TEMP| TEMPORARY keyword says that views are created in temporary space. Temporary views are automatically dropped at the end of current session.

Example of view

sno	sname	sbdate	saddress	sdiv	scourse
1	Rajesh	12/06/1958	Kothrud	A	BCS
2	Kavita	14/09/1952	Deccan	B	BCA
3	Sadhana	18/02/1950	Aundh	C	MCA
4	Radha	28/09/1965	SB Road	C	BCS
5	Kalpana	01/01/1960	Deccan	D	BCA

Example of view II

Create a view for student no, name and division.

- CREATE VIEW S_view1 AS
- SELECT sno, sname, sdiv
- FROM STUDENT;

S_view1 will be as follows:

sno	sname	sdiv
1	Rajesh	A
2	Kavita	B
3	Sadhana	C
4	Radha	C
5	Kalpana	D

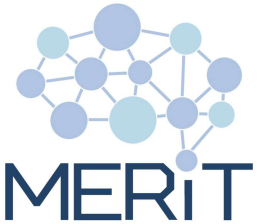
Example of view III

Create a view for student of C division.

- CREATE VIEW S_view2 AS
- SELECT *
- FROM STUDENT
- WHERE sdiv='C';
- S_view2 wil

sno	sname	sbdate	saddress	sdiv	scourse
3	Sadhana	18/02/1950	Aundh	C	MCA
4	Radha	28/09/1965	SB Road	C	BCS

Example of view IV



A view can be updated with the CREATE OR REPLACE VIEW statement.

Syntax

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2,                                    ...
FROM table_name
WHERE condition;
```

Definition of view can be changed by using ALTER VIEW command.

User can also change name of the view using RENAME command

```
ALTER VIEW S_view1 RENAME TO St_view1;
```

Dropping views

A view is deleted with the DROP VIEW statement.
Syntax: DROP VIEW [IF EXISTS] view_name;

The view_name specifies the name of the view is to be dropped.

IF EXISTS will check that view exists or not. If view does not exists and user tries to drop it then database shows error. To overcome this error IF EXISTS will be used.
Example
DROP VIEW IF EXISTS S_view1;

SUMMARY

- **Discretionary Access Control (DAC):**
  Grants and revokes user privileges based on ownership and access matrices. Flexible but less secure.

- **Mandatory Access Control (MAC):**
  Enforces access based on data sensitivity and user clearance. More secure, but harder to manage.

- **Role-Based Access Control (RBAC):**
  Centralizes access via roles assigned to users. Simplifies management for multilevel security systems.

- **Views for Security:**
  Restrict data access through customized views, enhancing data integrity and logical independence.

- **Audit Trails:**
  Record all changes with user, time, and data context for accountability and fraud detection.
  Together, these mechanisms form the foundation for secure, manageable, and scalable database access control.