Data Security

SQL - DCL

November, 2023

Security is an imp issue in database management because data should be protected from unauthorized access and updation…

SQL-DCL

- SQL defines an overall framework for DB security and SQL statements are used to specify security restrictions

- **SQL-DCL**
  - Mainly related with the security issues
  - Commands that control the user access to the DB objects
  - Who has access to the DB objects and what operations they can perform on them

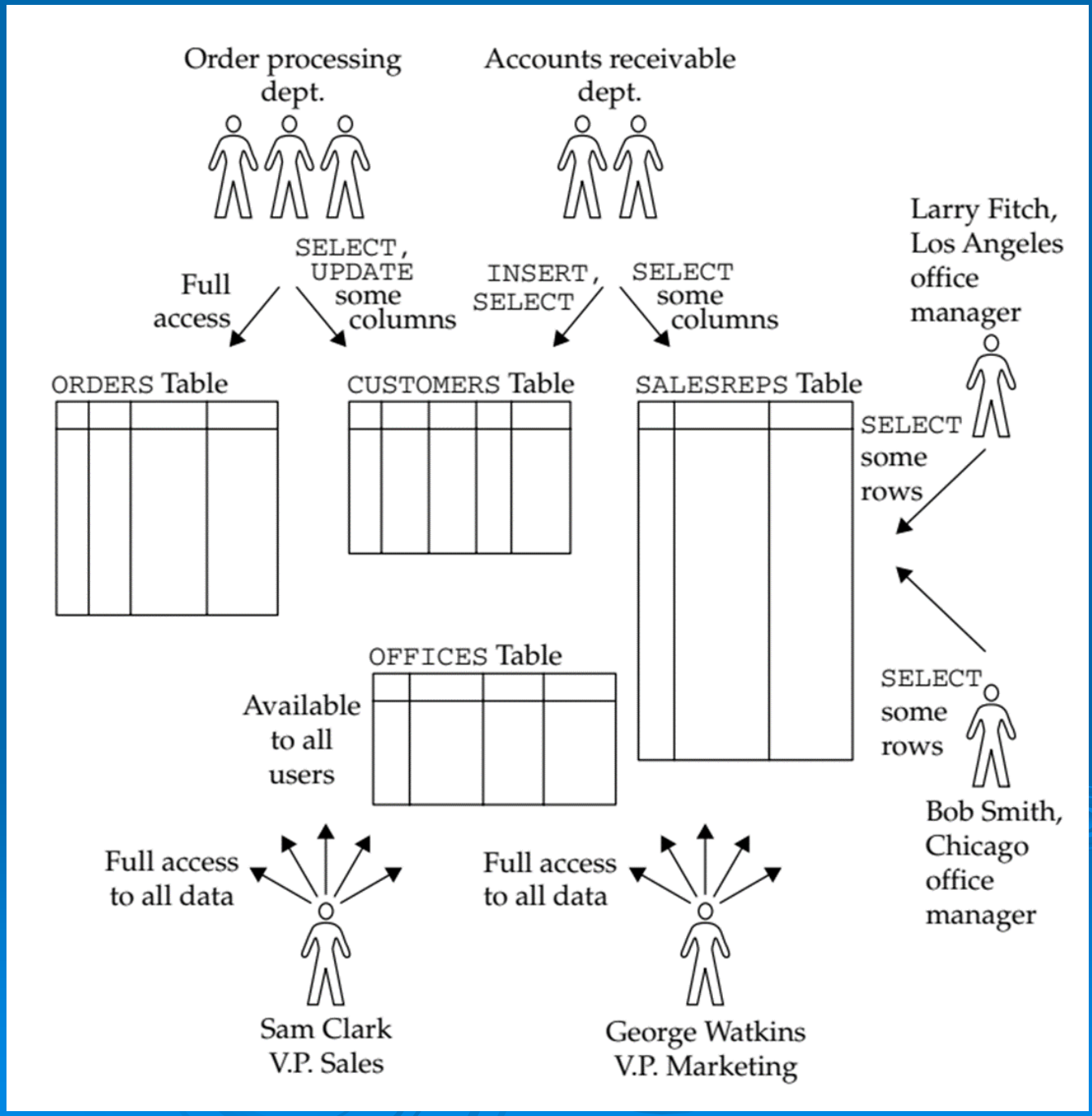
SQL Security Scheme

- Three central concepts
  - **Users:** the actors in the DB
  - **DB Objects:** the items to which SQL security protection can be applied
  - **Privileges:** the actions that a user is permitted to carry out for a given DB object

SQL Security Scheme

- **Users**
  - Each time the DBMS retrieves, inserts, deletes, or updates data, it does so on behalf of some user
  - The DBMS permits/prohibits the action depending on which user is making the request
- **DB Objects**
  - Usually tables and views, but other objects such as forms, application programs, and entire DB can also be protected
- **Privileges**
  - A user may have permission to SELECT and INSERT rows in a certain table, but may lack permission to DELETE or UPDATE rows of the table
  - Different users may have a different set of privileges

A security scheme for the sample database



SQL-DCL

- Issues to be taken care by the security mechanism of the DBMS
  - How does DBMS recognize the users?
  - How will it distinguish the authorized users from intruders?
  - How will it reject the request made by unauthorized users?

- **GRANT and REVOKE statements**
  - These statements specify – which users are to be allowed to perform which commands, on which tables, views or columns

DB Users

- Identification and authentication mechanisms through userid and password
- Privileges of each user will be stored in the DB
- DBMS has to ensure that each user does only what he/she is supposed to do
- At least two kinds of special users:
  - A super user (DBA)
  - The owners of DB objects
- Users other than the special users need explicit permission to perform any operation on a DB object

DBA

- A special user who is created when the system is installed
- In many installations, the role of the DBA is shared by more than one person
- Each of these users might login as the DBA but using their own username and password
- In this case, DBA becomes a role and is shared by several people who have the DBA access privileges to the system
  - Users have the power to do the specialized tasks, but their activities can be traced
- Usually DBAs have all the powers that the system can give on all DB objects and for all commands
- He can pass most of these powers to other users of the system if he wishes so

Owners of DB Objects

- In many systems, the user who creates a DB object is its owner
- Owner is usually given full privileges for the object he owns
- Like DBA, the owner of a DB object can grant his power to other users

Privileges (1/2)

USAGE	Privilege to use a specific domain
SELECT	Privilege to access all columns of a specific table/view including any columns added later
INSERT(x)	Privilege to insert into a specific named column x of a specific table or view
INSERT	Privilege to insert into all columns of a specific table or view, including any columns added later
UPDATE(x)	Privilege to update a specific named column x of a specific table/view

Privileges (2/2)

UPDATE	Privilege to update into all columns of a specific table or view, including any columns added later
DELETE	Privilege to delete rows from a specific table or view, including any columns added later
REFERENCES(x)	Privilege to reference a specific named column x of a specified table/view in integrity constraints
REFERENCES	Privilege to reference all columns of a specified table/view in integrity constraints

Reference Privileges

- Restricts a user's ability to create a reference to a table from a foreign key in another table
- Be careful
- The reference privilege deals with a more subtle SQL security issue posed by the SQL capabilities of foreign keys and check constraints

A Scenario

- An employee can create a new table in the database, but does not have any access to the employee information in the SALESREPS table.
- You might assume that there is no way for him to determine the employee numbers being used or whether a new employee has been hired.
- This isn't strictly true.
- The employee could create a new table, with a column defined as a foreign key to the SALESREPS table.
- Employee can try to insert new rows in the new table with different values in the foreign key column.
- The INSERT statements that succeed tell the employee that he has discovered a valid employee no.

Reference Privileges

- To eliminate this backdoor access to data, the SQL standard specifies a new REFERENCES privilege.
- Like the SELECT, INSERT, and UPDATE privileges, the REFERENCES privilege can be granted for specific columns of a table.
- Only if a user has the REFERENCES privilege for a column is he or she allowed to create a new table that refers to that existing column in any way.
- In databases that don't yet implement the REFERENCES privilege but do support foreign keys or check constraints, the SELECT privilege is sometimes used for this purpose.

Ownership Privileges

- When you create a table with the CREATE TABLE statement, you become its owner and receive full privileges for the table
- Other users initially have no privileges on the newly created table and privileges must be explicitly granted to them
- When you create a view with the CREATE VIEW statement, you become its owner of the view, but you do not necessarily receive full privileges on it
  - To create the view successfully, you must already have SELECT privilege on each of the source tables for the view; therefore DBMS gives you the SELECT privilege for the view automatically
  - For each of the other privileges, the DBMS gives you the privileges on the view only if you hold that same privilege on every source table for the view

Other Privileges

- Oracle and the IBM mainframe DBs support an ALTER and an INDEX privilege for tables
  - A user with the ALTER privilege on a particular table can use the ALTER TABLE statement to modify the definition of table
  - A user with the INDEX privilege can create an index for the table with the CREATE INDEX statement
- Sybase and SQL server support an EXECUTE privilege for stored procedures, which determines whether a user can create tables in a specific table space
- DB2 supports a USE privilege for tablespaces, which determines whether a user can create tables in a specific table space

Granting Privileges

- In most multiuser DBMSs, you need to
  - Be a special user – DBA or owner; or
  - Get explicit permission from either the DBA or owner to perform any operation,

    even to run a simple query.

Granting Privileges

- **General form of GRANT:**

  GRANT { ALL | privilege-list }
  ON { table name [(column-commalist)] | view name [(column-commalist)] }
  TO { PUBLIC | user list }
  [WITH GRANT OPTION]

- The user who grants privileges on a specific object must already hold the privilege on the specified item
- Granting a privilege on a view does not imply granting any privileges on the underlying relations

Granting Privileges - Examples

- **Give order processing users full access to the ORDERS table (id: OPUSER)**

```
GRANT SELECT, INSERT, DELETE, UPDATE
ON ORDERS
TO OPUSER
```

- **Allow Neeta to insert or delete an office**

```
GRANT INSERT, DELETE
ON OFFICES
TO NEETA
```

- **Grant all privileges to Neeta on the SALESREPS table**

```
GRANT ALL
ON SALESREPS
TO NEETA
```

Granting Privileges - Examples

- **Give all users SELECT access to the OFFICES table**

```
GRANT SELECT
ON OFFICES
TO PUBLIC
```

This GRANT statement grants access to all present and future authorized users.

- **Let order processing users (id: OPUSER) change company names and salesperson assignments**

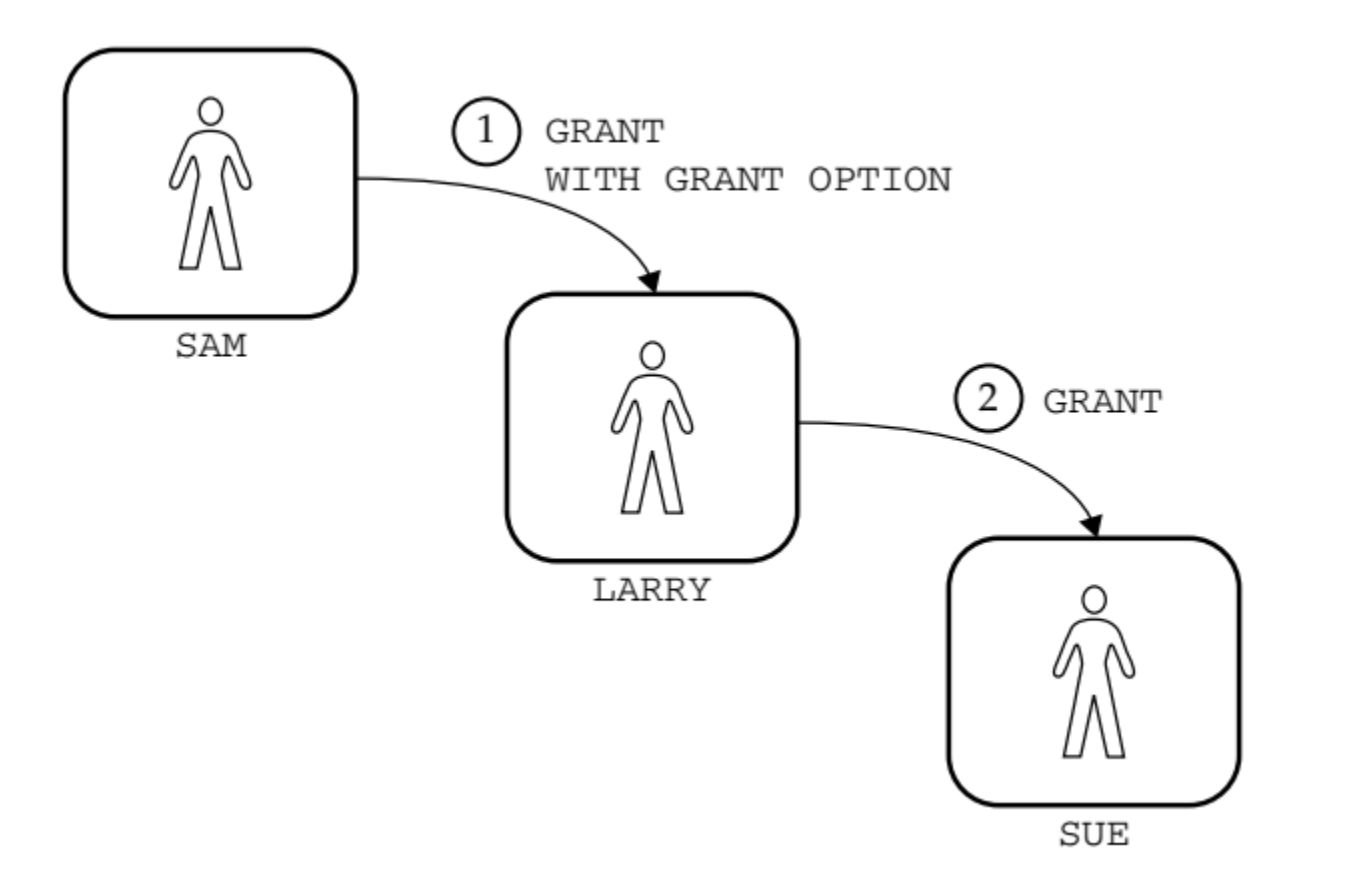
```
GRANT UPDATE (COMPANY, CUST_REP)
ON CUSTOMERS
TO OPUSER
```

- Some DBMS brands allow the column list for the SELECT privilege, although ANSI/ISO standard does not permit a column list for the SELECT privilege
  - If standard does not permit, effectively we can do so by designing a view and then grant permission on that view

Passing Privileges

- Once a user has been granted certain privileges with the GRANT OPTION, that user may grant those privileges and the GRANT OPTION to other users
- Those other users can, in turn, continue to grant both the privileges and the GRANT OPTION
- Take great care when giving other users the GRANT option
- GRANT OPTION applies only to the specific privileges named in the GRANT statement
- If you want to grant certain privileges with the GRANT OPTION and grant other privileges without it, you must use two separate GRANT statements

Passing Privileges - Example



Passing Privileges - Examples

- Neeta creates views XYZ and ABC, so she owns them

```
GRANT SELECT
ON XYZ
TO RAVI
```

```
GRANT SELECT
ON ABC
TO RAVI
WITH GRANT OPTION
```

Now RAVI can issue a command

```
GRANT SELECT
ON ABC
TO SEEMA
```

- Alternate way: Ravi might construct a view for Seema and give her access to that view

```
CREATE VIEW CCB AS
    SELECT * FROM ABC
    WHERE OFFICE = 2
```

```
GRANT SELECT
ON CCB
TO SEEMA
```

  - Ravi is the owner of CCB view, but he does not own ABC view. To maintain security, the DBMS requires that Ravi not only have the SELECT privilege on ABC, but also has GRANT OPTION for SELECT priviledge before allowing him to grant the SELECT priviledge on CCB to Seema

Revoking Privileges

- Used to take away the privileges that was granted

  REVOKE { ALL | privilege-list }
  ON { table name [(column-commalist)] | view name [(column-commalist)] }
  FROM { PUBLIC | user list }
  [**RESTRICT** | **CASCADE**]

- The user issuing the REVOKE command, should be the user who granted the privileges in the first place
- If the CASCADE option is provided, revocation of a privilege from a user causes privileges granted by the user to be also revoked

Revoking Privileges - Examples

- Take away all privileges granted earlier on the OFFICES table

```
REVOKE ALL
ON OFFICES
FROM PUBLIC
```

- The usual REVOKE statement takes away both the privileges and the ability to grant those privileges to others

```
REVOKE SELECT, UPDATE
ON ORDERS
FROM RAVI
```

- Another version:

```
REVOKE GRANT OPTION FOR SELECT, UPDATE
ON ORDERS
FROM RAVI CASCADE
```

Revoke and Grant Option

- When you grant privileges with the GRANT OPTION and later revoke these privileges, most DBMS brands will automatically revoke all privileges derived from the original grant
- If two different users grant the same privilege on the same object to a user and one of them later revokes the privilege, the second user's grant will still allow the user to access the object
- The time sequence of GRANT and REVOKE statements, rather than just the privileges themselves, can determine how far the effects of a REVOKE statement will cascade
- Granting and revoking privileges with GRANT OPTION must be handled very carefully, to ensure that the results are those you intend