Database Security and Authorization

ACS 575: Database Systems

Instructor: Dr. Jin Soung Yoo , Professor
Department of Computer Science
Purdue University Fort Wayne

Reference

- Ramakrishnan et al., Database Management Systems, Ch 21
- Oracle Document
- SQL Server Document

Outline

- Architecture of database security
- Create login/user
- Security mechanisms
- Permissions
- Role-based authorization
- Integrity control
- View and security

Architecture of Access to Database

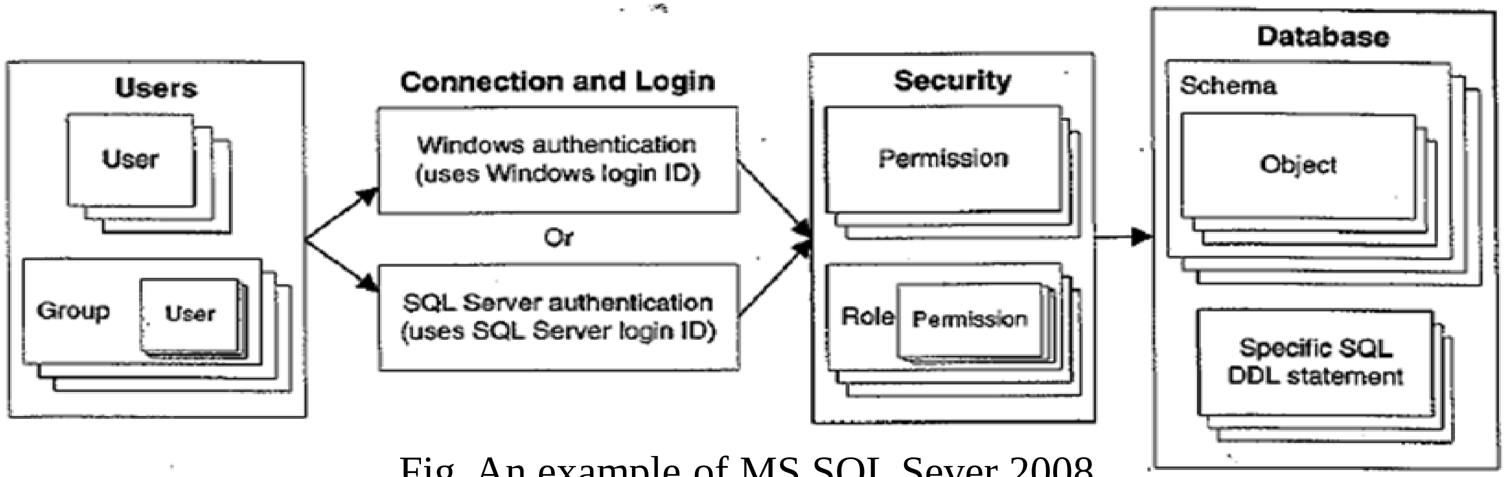

Users
User
Group
User
Connection and Login
Windows authentication (uses Windows login ID)
Or
SQL Server authentication (uses SQL Server login ID)
Security
Permission
Role
Permission
Database
Schema
Object
Specific SQL DDL statement

Fig. An example of MS SQL Sever 2008

- The user must connect and log on to the server using either an application program or a database client tool.
- The data the user has access to and the operations the user can perform depend on the ***permissions*** that have been granted to the user.
- You can grant ***object permissions*** so that user can perform specific actions on a specific database object, or you can grant ***database (system) permissions*** so the user can perform specific database operations.
- You can define a collection of permissions called a ***role***.

Authentication Mode

- User can log on to DMBS using one of two types of login authentication: *System authentication* or *Database server authentication*.
- **System authentication**
  - Authentication is handled by the security that’s integrated into the system authentication.
  - The login ID and password that the user enters to log on to the system are also used to log on to database server.
  - E.g., MS SQL Server uses it a default authentication mode.
- **Database server authentication** (Mixed mode)
  - The user must enter a database user ID and password to log on to database system. This user ID and password are separate from the system login id and password.
  - E.g., Oracle uses database server authentication mode.

Login vs. User in SQL Server

- A “login” grants the principal entry into the database server.
- A “user” grants a login entry into a single database.
- One login can be associated with many users (one per database)
- The user id and login id can be a same name or different names.

Creation of Login ID (MS SQL Server)

- For Windows authentication

```
CREATE LOGIN login_name FROM WINDOWS
        [WITH [DEFAULT_DATABASE = database]
        [, DEFAULT_LANGUAGE = language]]
```

- For SQL Server authentication

```
CREATE LOGIN login_name
        WITH PASSWORD = 'password' [MUST_CHANGE]
        [, DEFAULT_DATABASE = database]
        [, DEFAULT_LANGUAGE = language]
        [, CHECK_EXPIRATION = {ON|OFF}
        [, CHECK_POLICY = {ON|OFF}
```

Creation of Database Users

- Each database maintains a list of the users that are authorized to access that database.
- Database administrator user can create database user account.

- The syntax of creating database user

```
CREATE USER username …;

DROP USER username …;

ALTER USER username….;
```

Creation of Database Users (MS SQL Server)

- The syntax of the CREATE USER statement

```
CREATE USER user_name
 [{FOR|FROM} LOGIN login_name]
 [WITH DEFAULT_SCHEMA = schema_name]
```

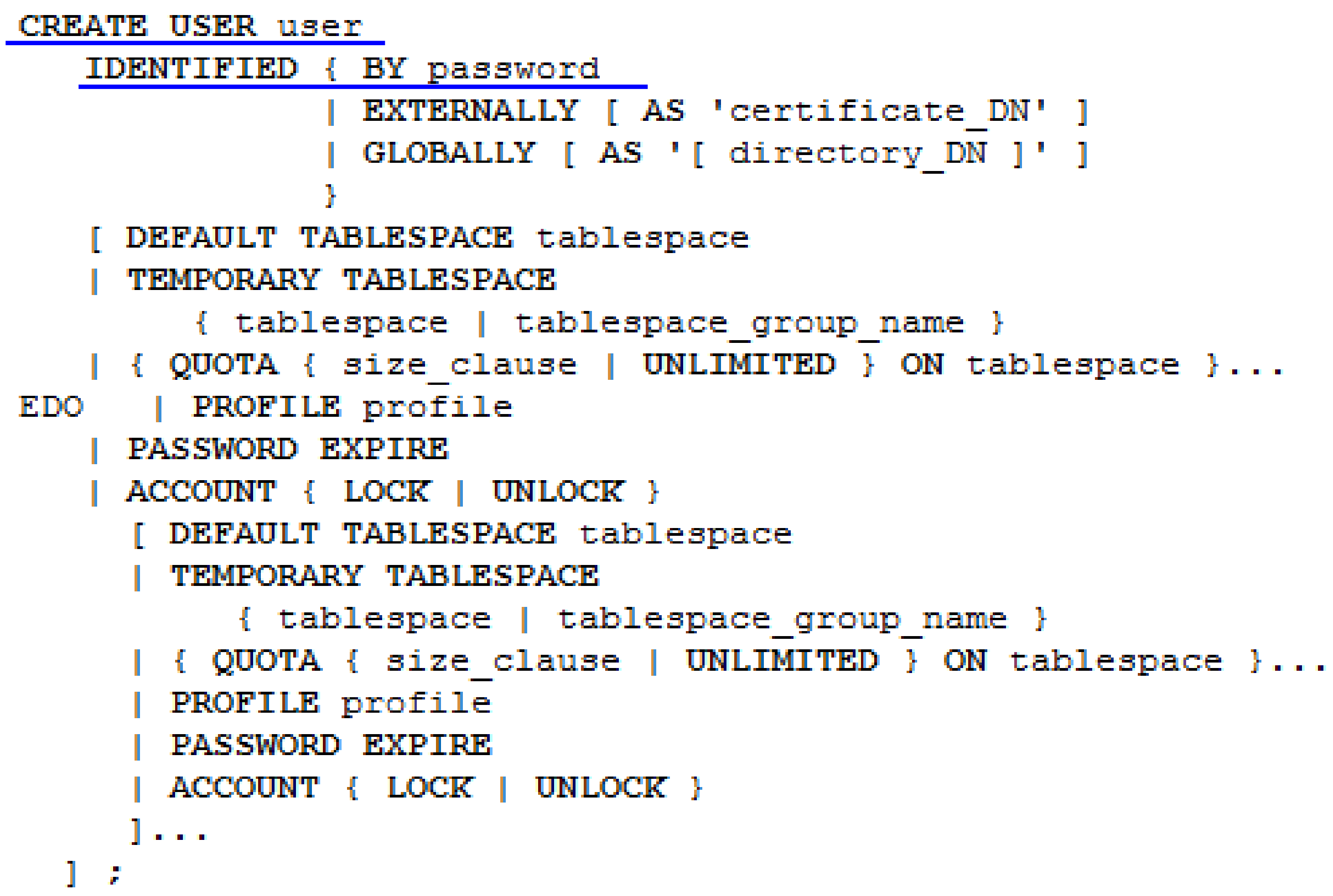
- The syntax of the ALTER USER statement

```
ALTER USER user_name WITH
 [NAME = new_user_name]
 [, DEFAULT_SCHEMA = schema_name]
```

- The syntax of the DROP USER statement

```
DROP USER user_name
```

Creation of Database Users (Oracle)



```
CREATE USER user
   IDENTIFIED { BY password
              | EXTERNALLY [ AS 'certificate_DN' ]
              | GLOBALLY [ AS '[ directory_DN ]' ]
              }
   [ DEFAULT TABLESPACE tablespace
   | TEMPORARY TABLESPACE
        { tablespace | tablespace_group_name }
   | { QUOTA { size_clause | UNLIMITED } ON tablespace }...
EDO   | PROFILE profile
   | PASSWORD EXPIRE
   | ACCOUNT { LOCK | UNLOCK }
     [ DEFAULT TABLESPACE tablespace
     | TEMPORARY TABLESPACE
          { tablespace | tablespace_group_name }
     | { QUOTA { size_clause | UNLIMITED } ON tablespace }...
     | PROFILE profile
     | PASSWORD EXPIRE
     | ACCOUNT { LOCK | UNLOCK }
     ]...
  ] ;
```

https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/CREATE-USER.html#GUID-F0246961-558F-480B-AC0F-14B50134621C

Outline

- Architecture of database security
- Create login/user
- ☞ **Security mechanisms**
  - Mandatory access control
  - Discretionary access control
- Permissions
- Role-based authorization
- Integrity control
- View and security

Main Objectives of Designing a Secure DB Application

- **Secrecy**: Users should not be able to see things they are not supposed to.
  - E.g., A student can’t see other students’ grades.
- **Integrity**: Users should not be able to modify things they are not supposed to.
  - E.g., Students not allowed to modify their grades
- **Availability**: Users should be able to see and modify things they are allowed to.
  - E.g., An instructor who wishes to change a grade should be allowed to do so.

Security Mechanisms

- A security mechanism allows us to enforce a chosen security policy.
- Two main security mechanisms at the DBMS level:
  - **Discretionary access control**
    - based on notion of privileges.
  - **Mandatory access control**
    - based on notion of security classes.

Mandatory Access Control

- **Mandatory access controls** (MAC) restrict the access of subjects to objects based on a system-wide policy
- Each DB object is assigned a security class.
- Each subject (user or user program) is assigned a security clearance.
- Example of security levels : Top Secret, Secrete, Confidential and Unclassified.
- According to security classes and clearances, govern who can read/write which objects.

Bell-LaPadula Model

- The security goal of **Bell-LaPadula Model** ensures that information do not flow to those not cleared for that level.

- Each object and subject is assigned a security class such as top secret (TS), secret (S), confidential (C), unclassified (U): TS > S> C > U
  - Subject S can read object O only if class(S) >= class(O) (Simple Security Property)
  - Subject S can write object O only if class(S) <= class(O) (*-Property)

Example

bid	bname	color	class
101	Salsa	Red	S
102	Pinto	Brown	C

- TS > S> C > U : Users with S and TS clearance will see both rows; a user with C will only see the 2nd row; a user with U will see no rows.
- Most commercial systems do not support mandatory access control. Versions of some DBMSs do support it; used for specialized (e.g., military) applications.

Discretionary Access Control

- **Discretionary access control** is based on the concept of
  - access rights or privileges for objects (tables and views), and systems
  - mechanisms for giving users privileges (and revoking privileges).
- Creator of a table or a view automatically gets all privileges on it.
- DMBS keeps track of who subsequently gains and loses privileges, and ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed.

Outline

- Architecture of database security
- Create login/user
- Security mechanisms
- ☞ **Permissions**
  - Object permissions
  - System permissions
- Role-based authorization
- Integrity control
- View and security

Permission

- The data the user has access to and the operations the user can perform depend on the ***permissions*** that have been granted to the user.
- Types of permissions
  - **Object permissions** are used so that user can perform specific actions on a specific database object
  - **System (/Database) permissions** are used so the user can perform specific database operations.

Grant or Revoke Permission

- The GRANT statement is used to grant permissions to database principal (e.g., user, group, role)

  GRANT object/system permissions …
  TO database principal

- The REVOKE statement is used to revoke permissions from database principal

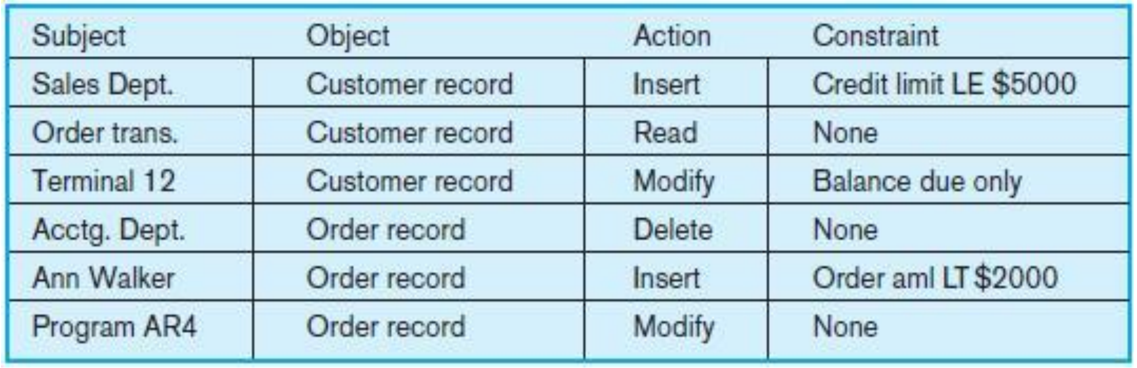
  REVOKE object/system permissions …
  FROM database principal

Object Permissions

Permission	Lets the user…	Applies to
`SELECT`	Select the data.	Tables, views, table-valued functions
`UPDATE`	Update existing data.	Tables, views, table-valued functions
`INSERT`	Insert new data.	Tables, views, table-valued functions
`DELETE`	Delete existing data.	Tables, views, table-valued functions
`EXECUTE`	Execute a procedure or function.	Stored procedures, scalar and aggregate functions
`REFERENCES`	Create objects that refer to the object.	Tables, views, functions
`ALL`	Have all applicable object permissions.	Tables, views, stored procedures, functions

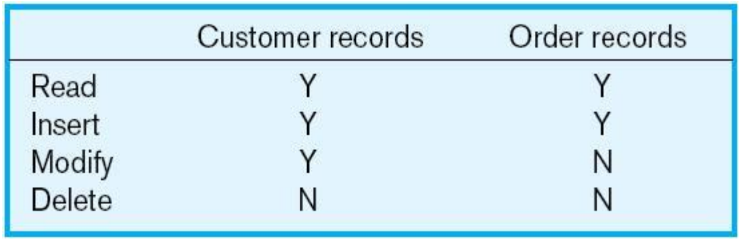
Example: Authorization Matrix

- Authorization matrix for
  - Subjects
  - Objects
  - Actions
  - Constraints



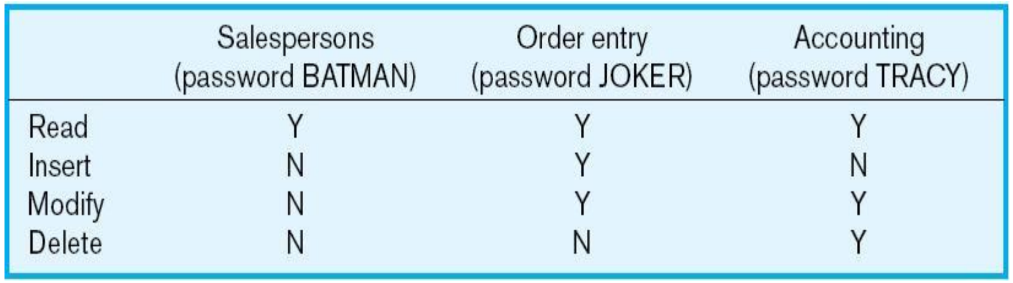
Subject	Object	Action	Constraint
Sales Dept.	Customer record	Insert	Credit limit LE $5000
Order trans.	Customer record	Read	None
Terminal 12	Customer record	Modify	Balance due only
Acctg. Dept.	Order record	Delete	None
Ann Walker	Order record	Insert	Order aml LT $2000
Program AR4	Order record	Modify	None

(a) Authorization matrix



	Customer records	Order records
Read	Y	Y
Insert	Y	Y
Modify	Y	N
Delete	N	N

(b) Authorization table for subjects (salespersons)



	Salespersons (password BATMAN)	Order entry (password JOKER)	Accounting (password TRACY)
Read	Y	Y	Y
Insert	N	Y	N
Modify	N	Y	Y
Delete	N	N	Y

(c) Authorization table for objects (orders)

GRANT Object Permissions

- The Syntax of GRANT statement

```
GRANT {ALL|permissions [, ...]}
ON object_name [(column [, ...])]
TO database_principal [, ...]
[WITH GRANT OPTION]
```

  - You can only grant permissions for a single object with GRANT statement.
  - The object can be a table, a view, a stored procedure, or a user-defined function.
  - In the TO clause, principal names (user, group, role) are given.
  - The WITH GRANT OPTION clause gives a user permission to grant the permission to other users.

Example

- **GRANT** INSERT, SELECT **ON** sailors **TO** Horatio
  - Horatio can query the sailors table or insert tuples into it.
- **GRANT** DELETE **ON** sailors **TO** Yuppy **WITH GRANT OPTION**
  - Yuppy can delete tuples, and also authorize others to do so.
- **GRANT** UPDATE (rating) **ON** sailors **TO** Dustin
  - Dustin can update (only) the *rating* field of Sailors tuples.

- **GRANT** SELECT **ON** department **TO** PUBLIC
  - All users see the tuples of the department table

REVOKE Object Permissions

- Syntax of REVOKE statement

```
REVOKE [GRANT OPTION FOR] {ALL|permission [, ...]}
ON object_name [(column [, ...])]
FROM database_principal [, ...]
[CASCADE]
```

- In REVOKE statement with WITH GRANT OPTION,
  - GRANT OPTION FOR revokes the user’s permission to grant the permission to others
  - CASCADE revokes the permission from any other users who were given the permission by this user.

Example

- GRANT SELECT ON sailors TO Art WITH GRANT OPTION
  - Executed by Joe
- GRANT SELECT ON sailors TO Bob WITH GRANT OPTION
  - Executed by Art
- REVOKE SELECT ON sailors FROM Art CASCADE
  - Executed by Joe
- **What happens in Bob?**
  - Bob loses this privilege too.

Outline

- Architecture of database security
- Create login/user
- Security mechanisms
- Permissions
  - Object permissions
  - ☞ **System permissions**
- Role-based authorization
- Integrity control
- View and security

System (/Database) Privileges

- A system privilege is the right to perform a particular action or to perform an action on any schema objects of a particular type.
- Over 100 distinct system privileges
  - CREATE DATABASE LINK
  - ALTER DATABASE
  - CREATE ANY INDEX
  - CREATE ANY TABLE, and so on.
  - Reference: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/GRANT.html#GUID-20B4E2C0-A7F8-4BC8-A5E8-BE61BDC41AC3
- System privileges are very powerful. Only grant them when necessary to trusted database principles (i.e., users, roles)

Grant/ Revoke System Privileges

- GRANT CREATE ANY TABLE TO public;
- GRANT CREATE INDEX, CREATE SYNONYM TO Bob;
- GRANT CREATE FUNCTION, CREATE PROCEDURE TO Bob;
- GRANT CREATE ANY CLUSTER TO Smith;
- GRANT CREATE SYNONYM TO Jone;

- REVOKE CREATE INDEX, CREATE SYNONYM FROM Bob;
- REVOKE CREATE FUNCTION, CREATE PROCEDURE FROM Bob;

Outline

- Architecture of database security
- Create login/user
- Security mechanisms
- Permissions
- ☞ **Role-based authorization**
- Integrity control
- View and security

Role-Based Authorization

- ***Roles*** are named groups of related privileges
- Roles can then be granted to users and to other roles.
  - Useful for quickly and easily granting permissions to users.
- Each role name must be unique, different all user name and all other role names

Pre-defined / User-defined Roles

- **Pre-defined database roles**
  - Database systems have pre-defined roles.
  - These roles can’t be deleted and the permissions associated with them can’t be modified.
- **User-defined database roles**

Pre-defined Roles in Oracle

Role Name	Description
CONNECT	Includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	Includes the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	All system privileges WITH ADMIN OPTION
EXP_FULL_DATABASE	Provides the privileges required to perform full and incremental database exports. Includes: SELECT ANY TABLE, BACKUP ANY TABLE, EXECUTE ANY PROCEDURE, EXECUTE ANY TYPE, ADMINISTER RESOURCE MANAGER, and INSERT, DELETE, and UPDATE on the tables SYS.INCVID, SYS.INCFIL, and SYS.INCEXP. Also the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
DELETE_CATALOG_ROLE	Provides DELETE privilege on the system audit table (AUD$)
…	…

Reference: https://docs.oracle.com/en/database/oracle/oracle-database/19/dbseg/configuring-privilege-and-role-authorization.html#GUID-A5B26A03-32CF-4F5D-A6BE-F2452AD8CB8A

Pre-defined Roles in MS SQL Server

Role Name	Description
sysadmin	Can perform any activity on the server. By default, all members of the Windows BUILTIN\Administrators group are members of the this role.
securityadmin	Can manage login IDs and passwords for the server and can grant, deny, and revoke database permissions.
dbcreator	Can create, alter, drop, and restore databases.
db_owner	Has all permissions for the database.
db_accessadmin	Can add or remove login IDs for the database.
db_securityadmin	Can manage object permissions, database permissions, roles, and role memberships.
db_ddladmin	Can issue all DDL statements except GRANT, REVOKE, and DENY.
db_datawriter	Can insert, delete, or update data from any user table in the database.
db_datareader	Can select data from any user table in the database.
…	…

Reference: https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-ver16

User defined Roles

- Unlike the fixed database roles, you can create your own user-defined roles, you can modify the permissions associated with those roles, and you can delete the roles when necessary
- Example

```
CREATE ROLE all_customer;

GRANT select, update ON customer TO all_customer;
GRANT select ON item_table TO all_customer;
GRANT all_customer TO fred, mary, joe;

DROP ROLE all_customer;
```

Outline

- Architecture of database security
- Create login/user
- Security mechanisms
- Permissions
- Role-based authorization
- ☞ **View and security**
- Integrity control

Views and Security

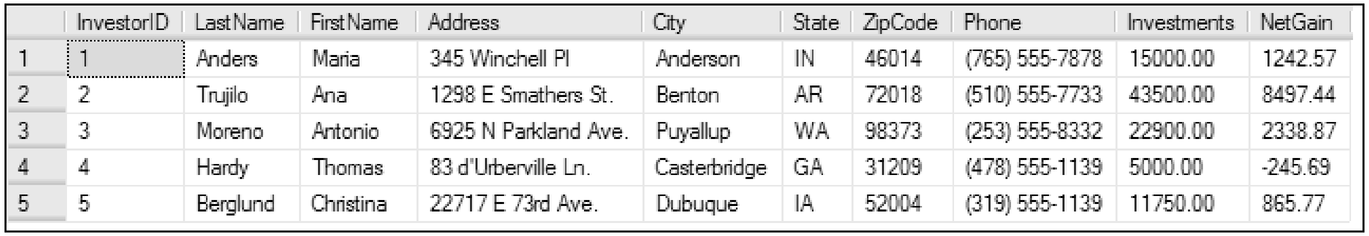
- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
  - E.g., Given ActiveSailors, but not Sailors or Reserves, we can find sailors who have a reservation, but not the *bid*'s of boats that have been reserved.
- Together with GRANT/REVOKE commands, views are a very powerful access control tool.

Benefits of using Views

Benefit	**Description**
Design independence	If the database design changes, you can modify the view so the queries that use it don't need to be changed.
Data security	Views can be used to limit access to the data that specific users are allowed to see.
Flexibility	Custom views can accommodate different data requirements.
Simplified queries	Views can hide the complexity of retrieval operations.
Updatability	With some restrictions, a view can be used to update data in a base table.

Example

- Data in a table of investors

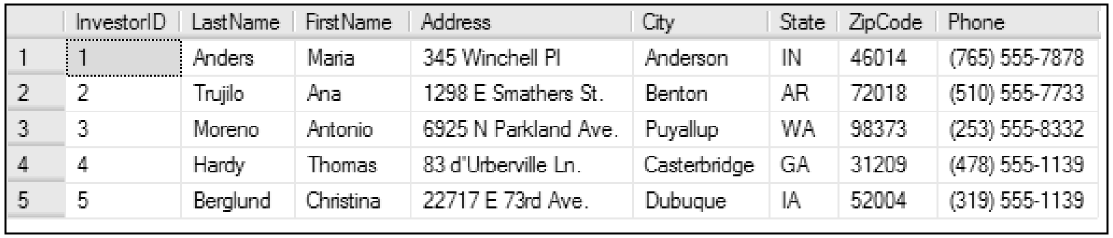


	InvestorID	LastName	FirstName	Address	City	State	ZipCode	Phone	Investments	NetGain
1	1	Anders	Maria	345 Winchell Pl	Anderson	IN	46014	(765) 555-7878	15000.00	1242.57
2	2	Trujilo	Ana	1298 E Smathers St.	Benton	AR	72018	(510) 555-7733	43500.00	8497.44
3	3	Moreno	Antonio	6925 N Parkland Ave.	Puyallup	WA	98373	(253) 555-8332	22900.00	2338.87
4	4	Hardy	Thomas	83 d'Urberville Ln.	Casterbridge	GA	31209	(478) 555-1139	5000.00	-245.69
5	5	Berglund	Christina	22717 E 73rd Ave.	Dubuque	IA	52004	(319) 555-1139	11750.00	865.77

- A view that restricts access to certain columns

```
CREATE VIEW InvestorsGeneral   AS
SELECT InvestorID, LastName, FirstName, Address,
   City, State, ZipCode, Phone  FROM Investors
```

- Data retrieved by the view



	InvestorID	LastName	FirstName	Address	City	State	ZipCode	Phone
1	1	Anders	Maria	345 Winchell Pl	Anderson	IN	46014	(765) 555-7878
2	2	Trujilo	Ana	1298 E Smathers St.	Benton	AR	72018	(510) 555-7733
3	3	Moreno	Antonio	6925 N Parkland Ave.	Puyallup	WA	98373	(253) 555-8332
4	4	Hardy	Thomas	83 d'Urberville Ln.	Casterbridge	GA	31209	(478) 555-1139
5	5	Berglund	Christina	22717 E 73rd Ave.	Dubuque	IA	52004	(319) 555-1139

CREATE VIEW statement

CREATE VIEW view_name [(column_name_1 [, column_name_2]...)]
[WITH
{ENCRYPTION|SCHEMABINDING|ENCRYPTION,SCHEMABINDING}]
AS
select statement
[WITH CHECK OPTION]

- You can use the WITH ENCRYPTION clause to keep users from examining the SQL code that defines the view.
- You can use the WITH SCHEMABINDING clause to bind a view to the *database schema*. Then, you can't drop the tables on which the view is based or modify them in a way that affects the view.
- You can use the WITH CHECK OPTION clause to prevent a row from being updated through a view if it would no longer be included in the view.

Privileges Required to Create View

- Some DBMS requires certain privileges for view creation.
- For example, in Oracle, to create a view, need the following privileges:
  - CREATE VIEW (or CREATE ANY VIEW) system privilege
  - SELECT, INSERT, UPDATE or DELETE object privileges on all base objects underlying the view

    (or SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE or DELTE ANY TABLE)

Update or Drop Views

- When you delete a view, any permissions that are connected to the view are also deleted.
- If you delete a table, you should also delete any views that are based on that table.
- If you specify the WITH SCHEMABINDING option when you create or modify a view, you won’t be able to delete the base tables without first deleting the view.
- If you specify WITH CHECK OPTION when you create a view, an error will occur if you try to modify a row in such a way that it would no longer be included in the view.
- If you don’t specify WITH CHECK OPTION, a change you make through the view can cause the modified rows to be excluded from the view.

GRANT on Views

- CREATE VIEW ActiveSailors
  AS
  SELECT *
  FROM Sailors, Reserves
  WHERE ….

- **GRANT** SELECT **ON** ActiveSailors **TO** Guppy, Yuppy
  - Guppy and Yuppy can select tuples in ActiveSailors.
  - This does NOT allow they query Sailors and Reserves directly!

REVOKE on Views

- If the creator of a view loses the SELECT privilege on an underlying table, the view is dropped!
- If the creator of a view loses a privilege held with the grant option on an underlying table, (s)he loses the privilege on the view as well

Outline

- ◻ Architecture of database security
- ◻ Create login/user
- ◻ Security mechanisms
- ◻ Permissions
- ◻ Role-based authorization
- ◻ View and security
- ☞ **Integrity control**

Integrity Controls

- Protect data from unauthorized use
- Domains–set allowable values
- Assertions–enforce database conditions
- Triggers – prevent inappropriate actions, invoke special handling procedures, write to log files

Domains

- A domain can be used to create a user-defined data type.
- Examples

```
CREATE DOMAIN PriceChange AS DECIMAL
     CHECK (VALUE BETWEEN 0.001 and 0.15);

CREATE TABLE PRICING
 ( …
      PriceIncrease PriceChange NOT NULL
      …
 );
```

Assertion

- When a constraint involves two or more tables, the table constraint mechanism is sometimes hard and results may not come as expected. To cover such situation SQL supports the creation of assertions that are constraints not associated with only one table.
- Example - An assertion to demand that Boston based departments do not employ trainers:

```
CREATE ASSERTION no_transfers_in_boston AS
      CHECK ( NOT EXISTS
         ( SELECT 'trainer in Boston'
           FROM EMP e, DEPT d
           WHERE e.DEPTNO = d.DEPTNO
                AND e.JOB    = 'TRAINER'
                AND d.LOC    = 'BOSTON') )
```

Conclusion

- Architecture of database security
- Create login/user
- Security mechanisms
  - Object permissions
  - System permissions
- Permissions
  - Object permissions
  - System permissions
- Role-based authorization
- Integrity control
- View and security

Appendix: Schema Object

Schema

- A **database** is the main container, it contains the data and all the schemas within it.
- **Schemas** are like folders within a database, and mainly used to group logical objects together which leads to ease of setting permissions by schema.
  - E.g., Group homogeneous set of objects in separated containers (aka schemas), so that all tables, views, stored procedures related to accounting information will be contained in the “Payroll” schema, where all objects related to employees and the like could be contained in the “HumanResources” schema…
- <u>An advantage of using schema is that you can grant permissions to all the objects in a schema by granting permissions to the schema.</u>

Schema

- The syntax of the CREATE SCHEMA statement

```
CREATE SCHEMA schema_name [AUTHORIZATION owner_name]
    [table definition]...
    [view definition]...
    [grant statement]...
    [revoke /deny statement]...
```

  - The tables, view, functions, and procedures of a database can be stored in “schemas”.

- The syntax of the DROP SCHEMA statement

```
DROP SCHEMA schema_name
```

Schema Creation – SQL Server Example

- This example creates schema ”Sprockets” owned by Annik that contains table NineProng, grants SELECT to Mandar, and denies SELECT to Prasanna.
- Note that Sprockets and NineProngs are created in a single statement.

```
CREATE SCHEMA Sprockets AUTHORIZATION Annik
    CREATE TABLE NineProngs (source int, cost int, partnumber
int)
    GRANT SELECT ON SCHEMA::Sprockets TO Mandar
    DENY SELECT ON SCHEMA::Sprockets TO Prasanna ;
```

Schema Creation – Oracle Example

- This example creates a schema named oe for the sample order entry user oe, creates the table new_product, creates the view new_product_view, and grants the SELECT object privilege on new_product_view to the sample human resources user hr.

```
CREATE SCHEMA AUTHORIZATION oe
    CREATE TABLE new_product (color VARCHAR2(10)
            PRIMARY KEY, quantity NUMBER)
    CREATE VIEW new_product_view
           AS SELECT color, quantity FROM new_product
               WHERE color = 'RED'
    GRANT select ON new_product_view TO hr ;
```