جامعة طرابلس
كلية تقنية المعلومات

قواعد البيانات المتقدمة Advanced Databases
ITSE312

د. عبدالسلام منصور الشريف

a.abdoessalam@uot.edu.ly

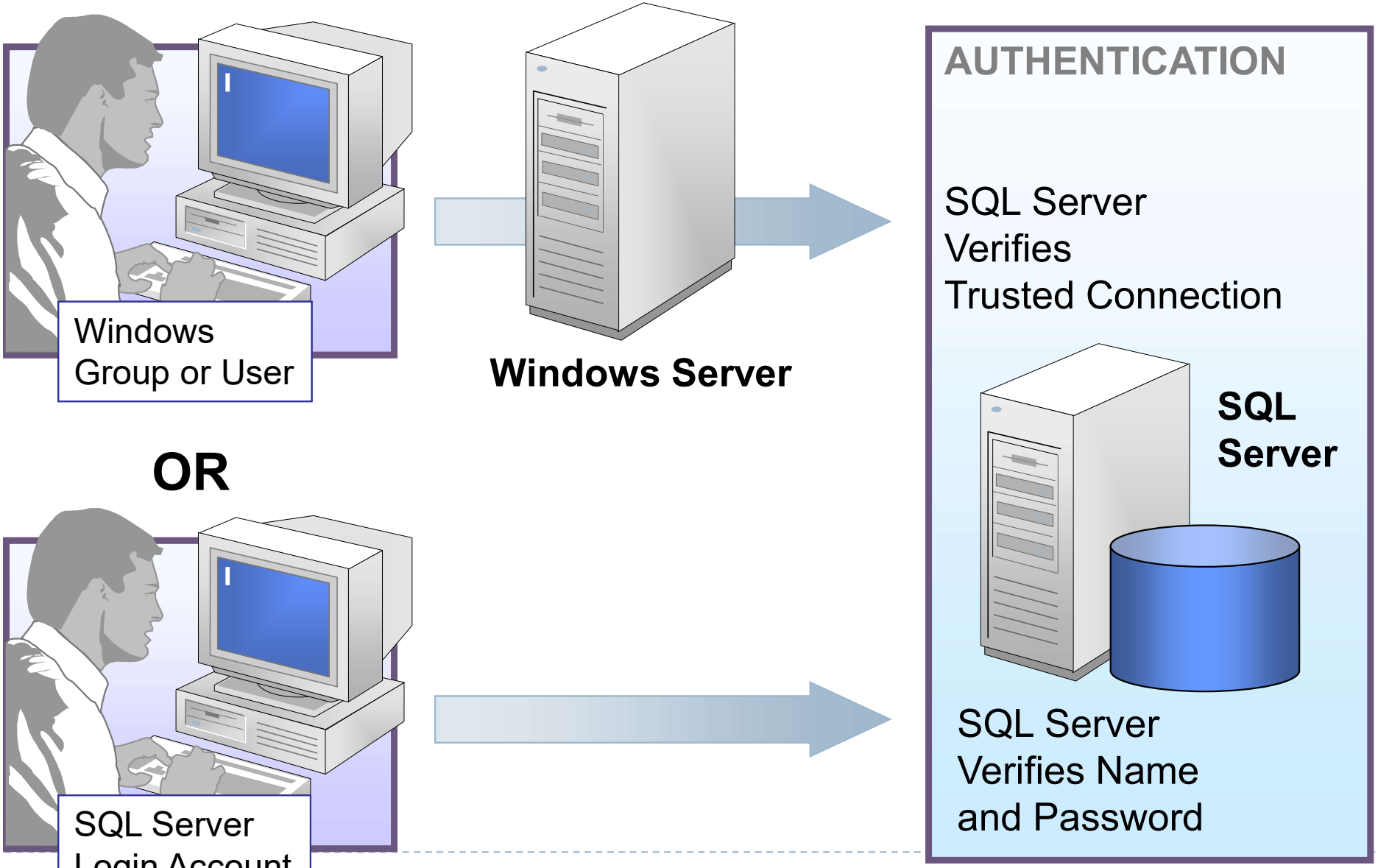
المحاضرة الحادية عاشر – لغة التحكم في البيانات I

Data Control Language I

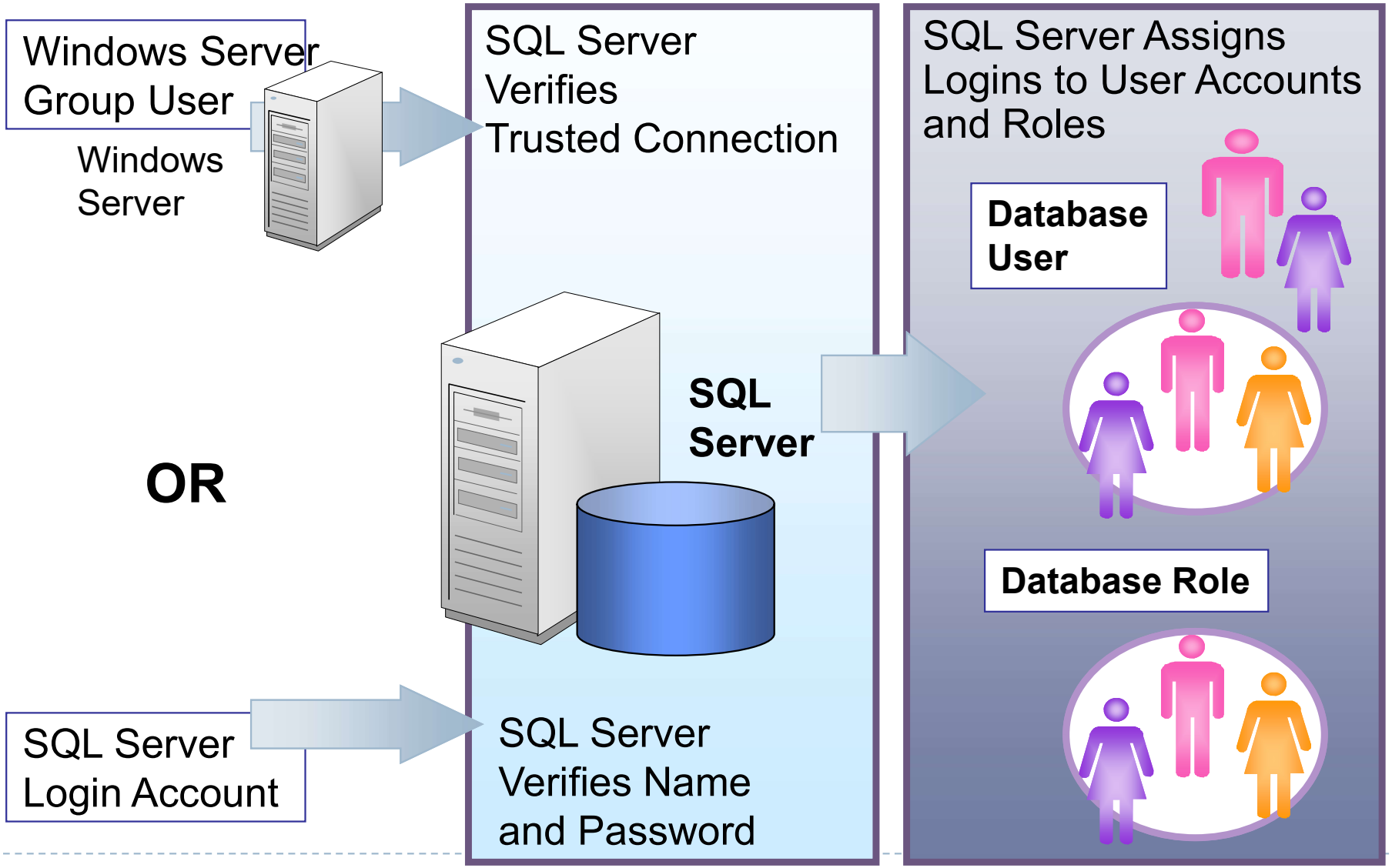
Contents

- Login Authentication
- Database User Accounts and Roles
- Types of Roles
- Permission Validation

Login Authentication



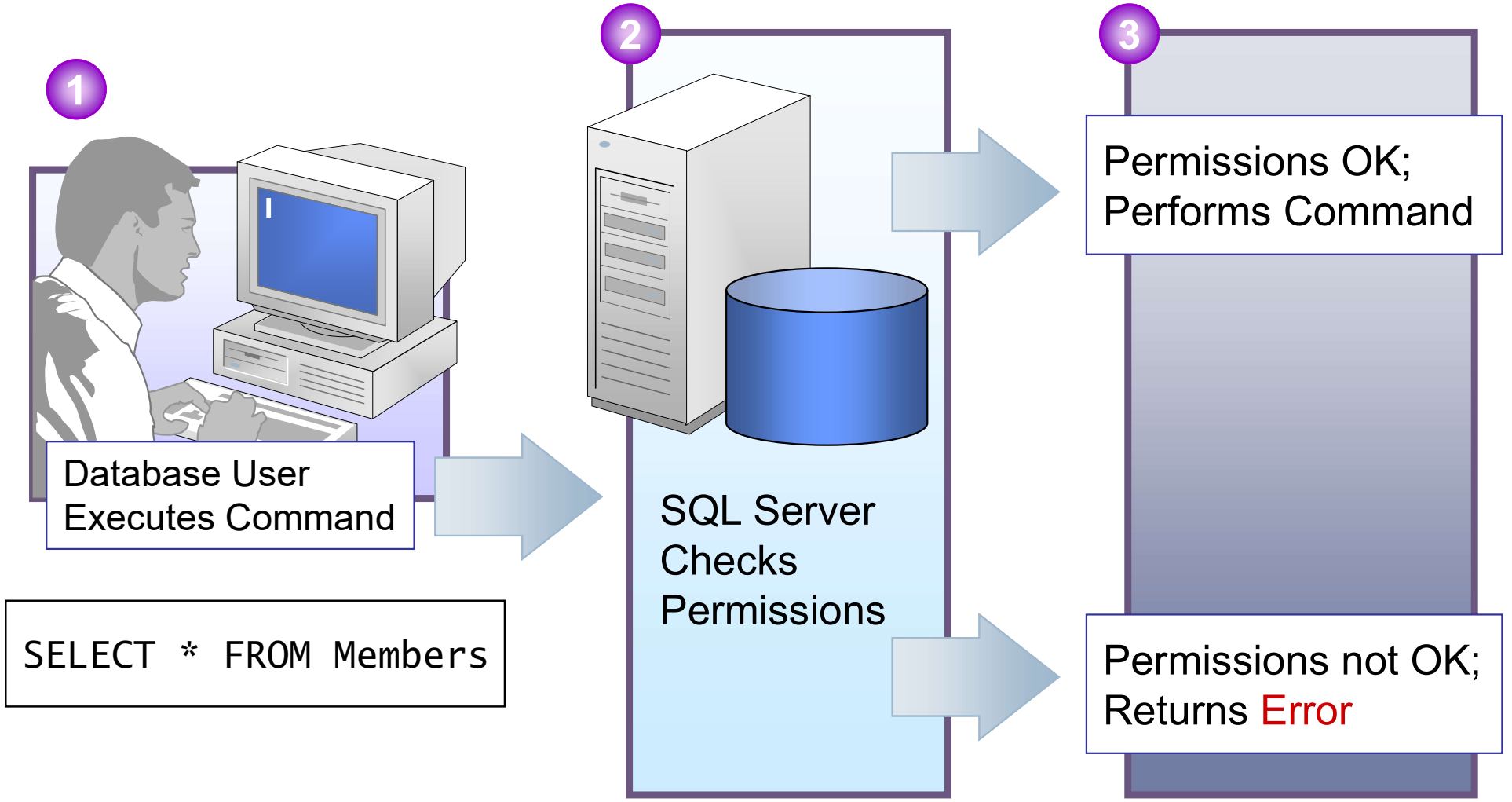
Database User Accounts and Roles



Types of Roles

- Fixed Server Roles
  - Group administrative privileges at the server level
- Fixed Database Roles
  - Group administrative privileges at the database level
- User-defined Database Roles
  - Represent work defined by a group of employees within an organization

Permission Validation



SQL Server Security Components

- To effectively protect SQL Server, you must be able to provide approved users with the access they need to specific SQL Server resources, without compromising those or other resources, a process that involves the use of three important types of components:

- **Principals**
  - Entities that can be authenticated to access the SQL Server resources. For example, your Windows login can be configured as a principal that allows you to connect to a SQL Server database. SQL Server supports three types of principals: logins, users, and roles.
    - Logins exist at the server level
    - Users exist at the database level
    - Roles can exist at either level.

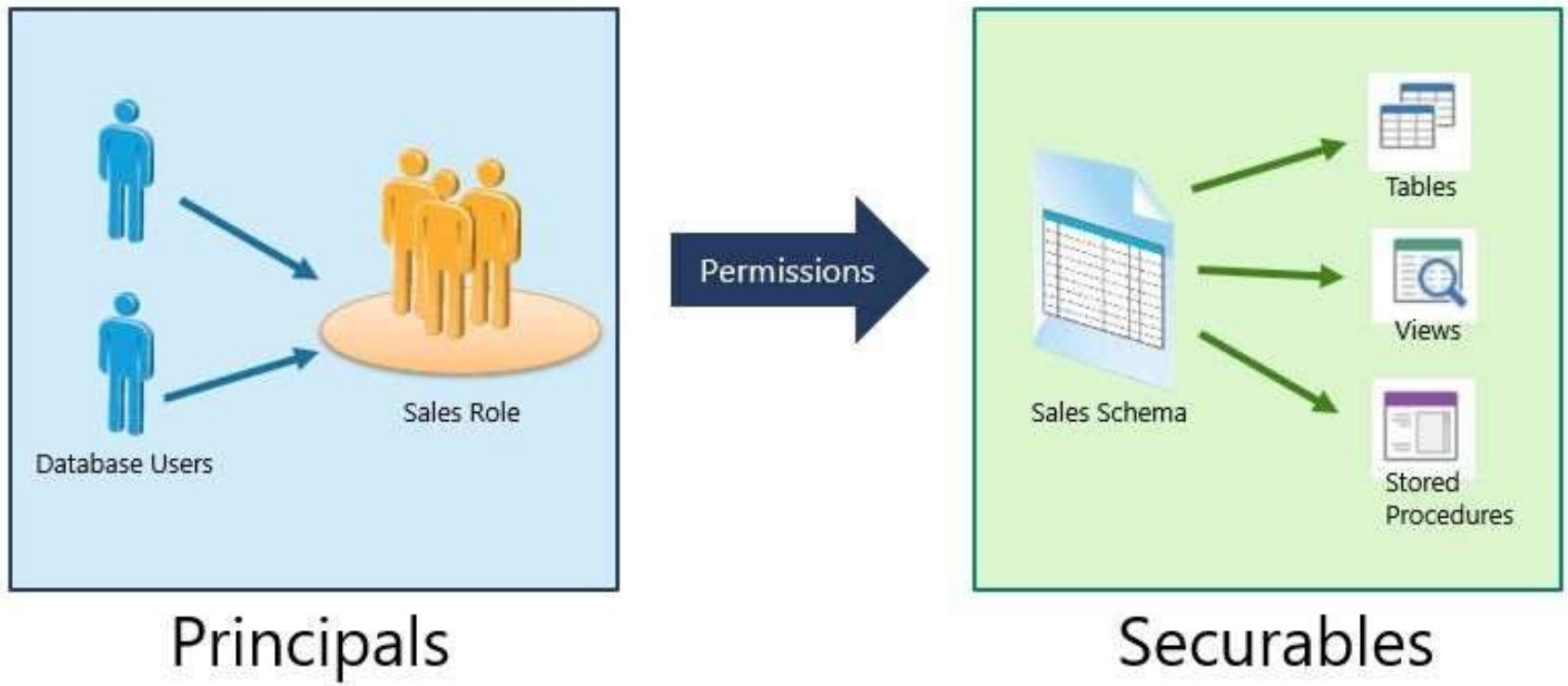
SQL Server Security Components

- **Securables**
  - SQL Server resources that can be accessed by a principal. Securables are the actual resources you're trying to protect, whether at the server level (e.g., availability groups), database level (e.g., full-text catalog), or the schema level (e.g., table or function).

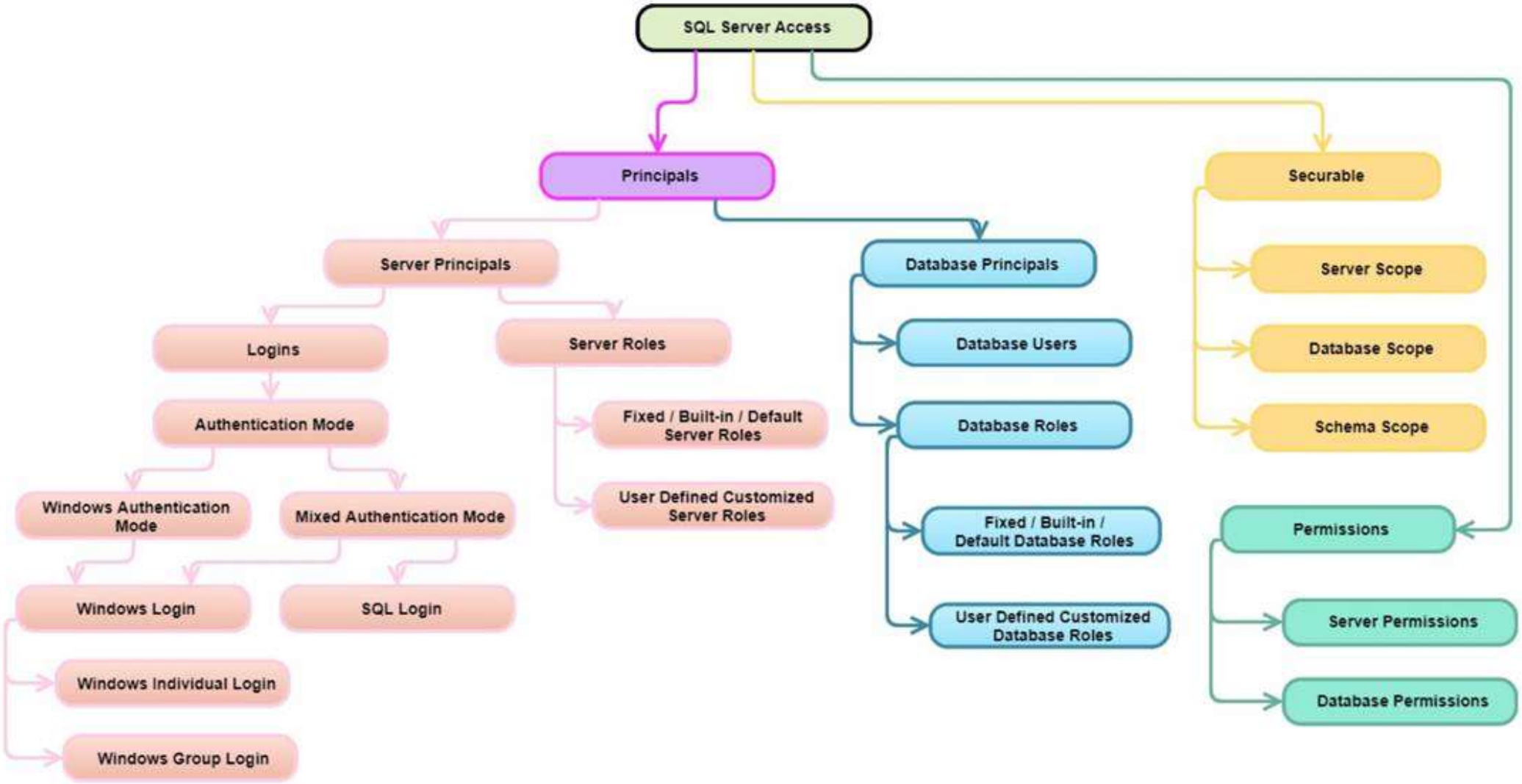
- **Permissions**
  - Types of access granted on a securable to a specific principal. For example, you can grant a Windows login (the principal) the ability to view data (the permission) in a specific database schema (the securable).

SQL Server Security Components

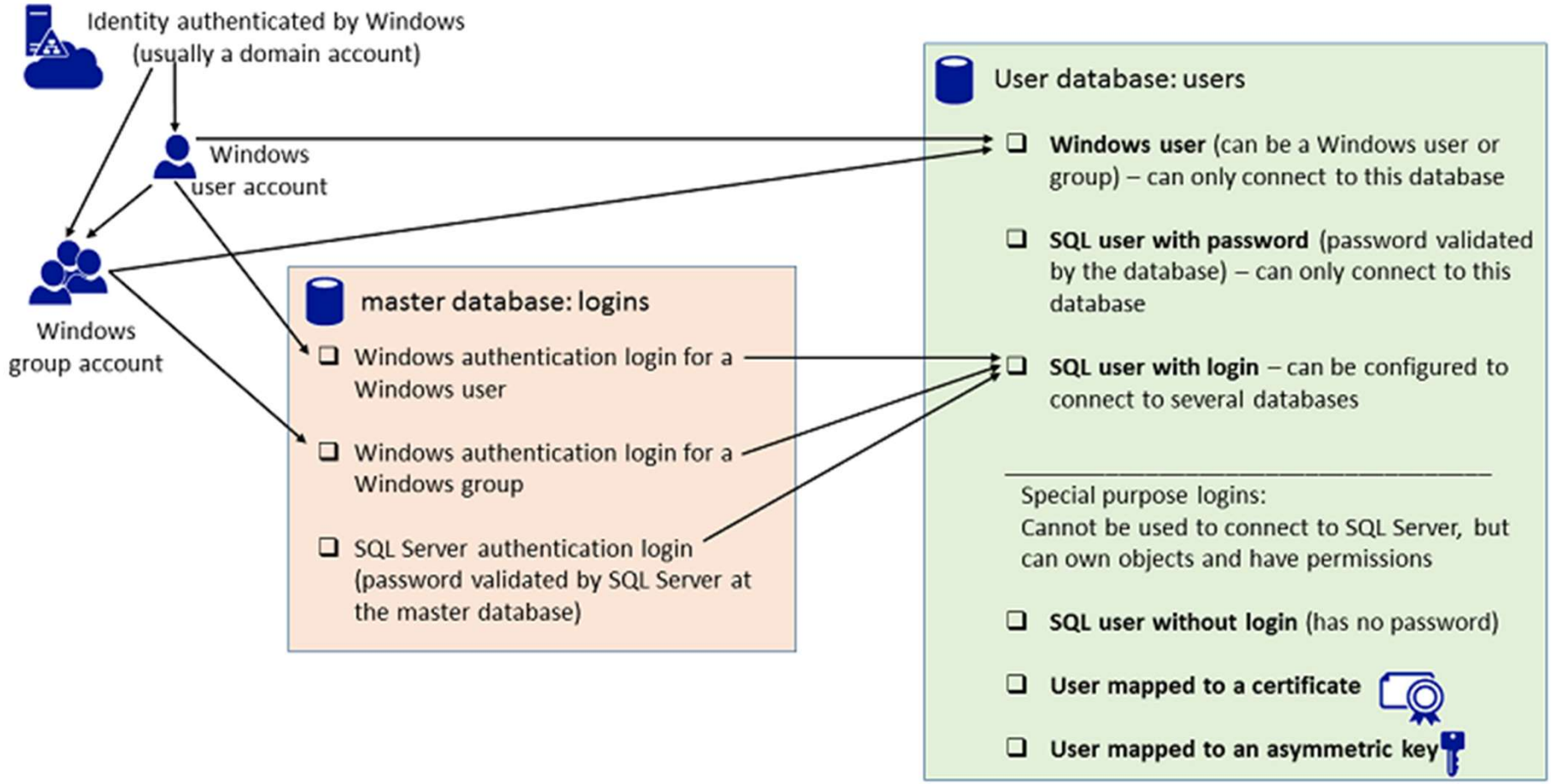
Principals vs Securables



SQL Server Security Components



SQL Server–Types of Users



SQL Server–Types of Users

- To determine the needed user:
  - First ask yourself, does the person or group that needs to access the database have a login?
  - Logins in the master database are common for the people who manage the SQL Server and for people who need to access many or all of the database on the instance of SQL Server.
  - For this situation, you will create a SQL user with login. The database user is the identity of the login when it is connected to a database. The database user can use the same name as the login, but that is not required.

SQL Server–Authenticated Logins

- SQL Server–authenticated logins are authenticated by the Database Engine instance rather than through the host operating system or a domain controller.

- SQL Server–authenticated login passwords are stored within the master database.

SQL Server–Authenticated Logins

- To create an authenticated login use:

```
CREATE LOGIN login_name
    { WITH PASSWORD = 'password'| FROM WINDOWS }
```

- To create an authenticated login *data_entry* with password:

```
CREATE LOGIN data_entry
      WITH PASSWORD = 'testme';
```

- To create an authenticated login from Windows user:

```
CREATE LOGIN [DESKTOP-126\Abdoessalam]
      FROM WINDOWS;
```

SQL Server–Authenticated Logins

- After creating a login, the login can connect to SQL Server, but only has the permissions granted to the public role.
- Consider performing some of the following activities.
  - To connect to a database, create a database user for the login.
  - Create a user-defined server role by using CREATE SERVER ROLE. Use ALTER SERVER ROLE ...ADD MEMBER to add the new login to the user-defined server role.
  - Use sp_addsrvrolemember to add the login to a fixed server role.
  - Use the GRANT statement, to grant server-level permissions to the new login or to a role containing the login.

SQL Server–Database Users

- In order for the SQL server login to access a database, a database user must be created:

```
CREATE USER user_name FROM LOGIN login_name
```

- To create a database user *data_entry user for the current database*:

```
CREATE USER data_entry FROM LOGIN data_entry;
```

SQL Server–Database Users

- Creating a user grants access to a database but does not automatically grant any access to the objects in a database.

- After creating a user, common actions are to add users to database roles that have permission to access database objects, or grant object permissions to the user.

SQL Server - Permissions

- Every SQL Server securable has associated permissions that can be granted to a principal.
- Permissions in the Database Engine are managed at the server level assigned to logins and server roles.
- Permissions at the database level assigned to database users and database roles.

How do Permissions work?

- Permissions have a parent/child hierarchy:
  - If you grant SELECT permission on a database, that permission includes SELECT permission on all (child) schemas in the database.
  - If you grant SELECT permission on a schema, it includes SELECT permission on all the (child) tables and views in the schema.
  - The permissions are transitive:
    - If you grant SELECT permission on a database, it includes SELECT permission on all (child) schemas, and all (grandchild) tables and views.
  - Permissions also have covering permissions. The CONTROL permission on an object, normally gives you all other permissions on the object.

How do Permissions work?

- Because both the parent/child hierarchy and the covering hierarchy can act on the same permission, the permission system can get complicated. For example, let's take a `testtable` table , in a `testschema` , in a `ITDatabase`:

  - **CONTROL** permission on the `testtable` table includes all the other permissions on the table, including ALTER, SELECT, INSERT, UPDATE, DELETE, and some other permissions.

  - **SELECT** on the `testschema` that owns the table includes the SELECT permission on the `testtable` table.

How do Permissions work?

- To allow users to have permissions on objects use:

```
GRANT { ALL [ PRIVILEGES ] }
      [ ON [ class :: ] securable ]
      TO principal [ ,...n ]
      [ WITH GRANT OPTION ] [ AS principal ]
```

- To give the user *data_entry* a select permission on the testable use:

```
GRANT SELECT ON OBJECT::testschema.testable
      TO data_entry;
```

Grant Permissions

```
GRANT SELECT ON OBJECT::testtable TO data_entry;

GRANT CONTROL ON OBJECT::testtable TO data_entry;

GRANT SELECT ON SCHEMA::testschema TO data_entry;

GRANT CONTROL ON SCHEMA::testschema TO data_entry;

GRANT SELECT ON DATABASE::ITDatabase TO data_entry;

GRANT CONTROL ON DATABASE::ITDatabase TO data_entry;
```