To begin with ....

1. Database server Host (or simply Host): the physical machine on which the DBMS Engine (a program) is running
2. The Database server Host is controlled by the OS (e.g. Windows, Linux)
3. Note that multiple *instances* of the DBMS Engine may be running on the same Host
Database: a set of data managed by a DBMS
A DBMS Engine can manage *multiple* Databases

Principals
- Windows Level
  - Windows login
  - Windows group
- SQL Server Level
  - SQL Server Login
  - SQL Server Roles (fixed)
- Database Level
  - Database User
  - Database Role
  - Application Role

Securables
- Resources to which access is regulated
- Can be arranged in a hierarchy

Permissions
- Every securable has associated permissions that can be granted to a principal

SQL Server
Securing an SQL Server involves three areas:
- the platform (host) and the network,
- principals and securables, and
- applications that access the database

Ref:
Securables

- Securables are the resources to which the SQL Server Database Engine authorization system regulates access.
- Some securables can be contained within others, creating nested hierarchies called "scopes" that can themselves be secured.
- The securable scopes are server, database, and schema.

Privileges

- **ALTER**
  Confers the ability to change the properties, except ownership, of a particular securable. When granted on a scope, ALTER also bestows the ability to alter, create, or drop any securable that is contained within that scope. For example, ALTER permission on a schema includes the ability to create, alter, and drop objects from the schema.
- **ALTER ANY <Server Securable>**, where Server Securable can be any server securable.
  Confers the ability to create, alter, or drop individual instances of the Server Securable. For example, ALTER ANY LOGIN confers the ability to create, alter, or drop any login in the instance.
- **ALTER ANY <Database Securable>**, where Database Securable can be any securable at the database level.
  Confers the ability to CREATE, ALTER, or DROP individual instances of the Database Securable. For example, ALTER ANY SCHEMA confers the ability to create, alter, or drop any schema in the database.
- **CREATE <Server Securable>**
  Confers to the grantee the ability to create the Server Securable
- **CREATE <Database Securable>**
  Confers to the grantee the ability to create the Database Securable
- **CREATE <Schema-contained Securable>**
  Confers the ability to create the schema-contained securable. However, ALTER permission on the schema is required to create the securable in a particular schema.
- **CONTROL**
  - ownership like capabilities on the grantee
  - grantee can also grant permission to other principals
  - implies all permissions on all securables under the scope of the securable on which it is granted
**Hierarchical permission model**

- For example: Alter permission on schema_1 implies Alter permission on all securables (such as schema_1.table_1) defined under schema_1.
- Alter on table_1 is the implied permissions; Alter on schema_1 is the covering permission

---

**Creating users in SQL Server 2008**

CREATE USER user_name
{ { FOR | FROM }
{ LOGIN login_name | CERTIFICATE cert_name | ASYMMETRIC KEY asym_key_name }
WITHOUT LOGIN ]
[ WITH DEFAULT_SCHEMA = schema_name ]

---

**Arguments**

- user_name Specifies the name by which the user is identified inside this database. user_name is a sysname. It can be up to 128 characters long.
- LOGIN login_name Specifies the SQL Server login for which the database user is being created. login_name must be a valid login in the server. When this SQL Server login enters the database it will acquire the name and ID of the database user that is being created.
- CERTIFICATE cert_name Specifies the certificate for which the database user is being created.
- ASYMMETRIC KEY asym_key_name Specifies the asymmetric key for which the database user is being created.
- WITH DEFAULT_SCHEMA = schema_name Specifies the first schema that will be searched by the server when it resolves the names of objects for this database user.
- WITHOUT LOGIN Specifies that the user should not be mapped to an existing login

---

**Creating users in SQL Server 2008**

Example: create a user mapped to SQL server login

- The following example first creates a server login named Bob with a password, and then creates a corresponding database user for the SimpleDB database:

```
CREATE LOGIN Bob
  WITH PASSWORD = '340SUuxwp7Mdxo7Khy';
USE SimpleDB;
CREATE USER Bob FOR LOGIN Bob
GO
```
Creating users in SQL Server 2008

1. CREATE LOGIN Bob
2.  WITH PASSWORD = '340$Uuxwp7Mcxo7Khy';
3. USE SimpleDB;
4. CREATE USER Bob FOR LOGIN Bob
GO

Example: create a user mapped to SQL server login

USE SimpleDB;
CREATE CERTIFICATE CarnationProduction 50
  WITH SUBJECT = 'Carnation Production Facility Supervisors',
  EXPIRY_DATE = '11/11/2011';
GO
CREATE USER TomFord FOR CERTIFICATE CarnationProduction50;
GO

Note: The term subject refers to a field in the metadata of the certificate as defined in the X.509 standard. The subject can be up to 128 characters long.

Permission Assignment Commands

- **GRANT** –
  - Grants a permission on a securable
  - WITH GRANT OPTION: gives the ability to grant the permission to other principals

- **DENY** –
  - Denies a previously granted permission (we will explore this in detail)

- **REVOKE** –
  - Removes a previously granted or denied permission.
  - CASCADING: removes the revoked permission from other principals to which it has been granted by this principal.

Fixed Server Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Every SQL Server login belongs to the public server role.</td>
</tr>
<tr>
<td>dbcreator</td>
<td>Can create, alter, drop, and restore any database.</td>
</tr>
<tr>
<td>diskadmin</td>
<td>Can manage disk files.</td>
</tr>
<tr>
<td>bulkadmin</td>
<td>Can run the BULK INSERT statement.</td>
</tr>
<tr>
<td>setupadmin</td>
<td>Can add and remove linked servers.</td>
</tr>
<tr>
<td>processadmin</td>
<td>Can end processes that are running in an instance of SQL Server.</td>
</tr>
<tr>
<td>securityadmin</td>
<td>Manages logins and their properties. Can also GRANT, DENY, and REVOKE database-level permissions.</td>
</tr>
<tr>
<td>serveradmin</td>
<td>Can change server-wide configuration options and shut down the server.</td>
</tr>
<tr>
<td>sysadmin</td>
<td>Can perform any activity in the server.</td>
</tr>
</tbody>
</table>
Fixed Database Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_owner</td>
<td>can perform all configuration and maintenance activities on the database.</td>
</tr>
<tr>
<td>db_securityadmin</td>
<td>can modify role membership and manage permissions.</td>
</tr>
<tr>
<td>db_accessadmin</td>
<td>can add or remove access to the database for Windows logins, Windows groups, and SQL Server logins.</td>
</tr>
<tr>
<td>db_backupoperator</td>
<td>can back up the database.</td>
</tr>
<tr>
<td>dbddladmin</td>
<td>can run any DDL command in a database.</td>
</tr>
<tr>
<td>db_datawriter</td>
<td>can add, delete, or change data in all user tables.</td>
</tr>
<tr>
<td>db_datareader</td>
<td>can read all data from all user tables.</td>
</tr>
<tr>
<td>db_denydatawriter</td>
<td>cannot add, modify, or delete any data in the user tables within a database.</td>
</tr>
<tr>
<td>db_denydatareader</td>
<td>cannot read any data in the user tables within a database.</td>
</tr>
</tbody>
</table>

Interesting Features

- Negative permissions
- User-Schema Separation
- Application Roles

Interesting Features

**NEGATIVE PERMISSIONS**

DENY (ALL [ PRIVILEGES ]) | permission (column [ ,...n ] ) [ ,...n ] [ ON [ class :: ] securable ] TO principal [ ,...n ] [
CASCADE] [ AS principal ]

"Denies a permission to a principal.
Prevents that principal from inheriting the permission through its group or role memberships."
### DENY – Salient features

- Denied permission takes precedence over the granted permission
- The caller of DENY must have the CONTROL permission on the securable
- What happens when a permission is denied on a base object or on a derived object?
- What happens in case of a role hierarchy?

### Example: DENY takes precedence

- Grant select on t1 to u1;
- Deny select on t1 to r1;
- Exec sp_addrolemember 'r1', 'u1';
- u1: Select * from t1;
- Result: **Access Denied !**

### Example: DENY on base object

- **Base Object**: table t1
- **Derived Object**: view v1 (select c1 from t1)
- Grant select on v1 to u1;
- Deny select on t1 to u1;
- u1: Select * from v1;
- Result: **Access Granted !**

### Example: DENY on derived object

- **Base Object**: table t1
- **Derived Object**: view v1 (select c1 from t1)
- Grant select on t1 to u1;
- Deny select on v1 to u1;
- u1: Select * from t1;
- Result: **Access Granted !**
Example

DENY in a role hierarchy

- Create role r1;
- Create role r2;
- exec sp_addrolemember 'r1', 'r2';
- exec sp_addrolemember 'r1', 'u1';
- Grant select on t1 to r1;
- Deny select on t1 to r2;
- u1: Select * from t1;
- Result: Access Allowed !

Example

DENY in a role hierarchy

- Create role r1;
- Create role r2;
- exec sp_addrolemember 'r1', 'r2';
- exec sp_addrolemember 'r1', 'u1';
- Grant select on t1 to r1;
- Deny select on t1 to r2;
- u1: Select * from t1;
- Result: Access Denied !

Example

DENY in a permission hierarchy

USE SimpleDB
GO

GRANT SELECT ON authors TO public
GO

DENY SELECT, INSERT, UPDATE, DELETE
ON authors TO Mary, John, Tom

USER-SCHEMA
SEPARATION
A schema is a container for database objects such as tables, views, stored procedures etc.
Every securable in a specific schema must have a unique name.
A schema also acts as namespace.

A schema can be owned by any database principal such as user, database role or application role.
The schema owner is the owner of all objects in the schema.
Every database user has a default schema. Multiple users can share a single default schema.

Ownership of schemas and schema-scoped securables is transferable (using the *Alter Authorization* command).
Objects can be moved between schemas (using the *Alter Schema* command).
A single schema can contain objects owned by multiple database users.
**Application roles**

- Database roles are the regular roles that we are aware of.
- An application role is a database principal that enables an application to run with its own, user-like permissions.

**Creation of roles**

```
CREATE ROLE role_name
[AUTHORIZATION owner_name]
```

Arguments:
- `Role_name`: is the name of the role to be created
- `AUTHORIZATION owner_name`: is the database user or role that is to own the role. If no user is specified, the role will be owned by the user that executes `CREATE ROLE`.

**Application roles usage**

The following steps make up the process by which an application role switches security contexts:
- A user executes a client application.
- The client application connects to an instance of SQL Server as the user.
- The application then executes the `sp_setapprole` stored procedure with a password known only to the application.
- If the application role name and password are valid, the application role is enabled.
- At this point the connection loses the permissions of the user and assumes the permissions of the application role.
- `sp_unsetapprole` stored procedure can be used to revert the session to its original context.