**Databases in SQL Server**

There are basically the following two types of databases in SQL Server:

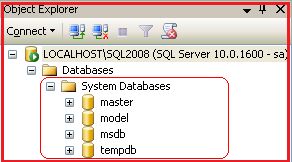
1. System Databases.
2. User Databases.

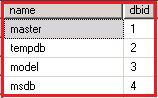
**System Databases in SQL Server**

System databases are databases that are created when SQL Server is installed. These databases are used for various operational and management activities for SQL Server.

**Types of System Databases**

There are basically four system databases in SQL Server, **master**, **msdb**, **tempdb** and **model** that we can see. Apart from that, there is one more system database **resource database** that is **hidden and read-only**. Let's move ahead with each system database.





## Master Database in SQL Server

* All the system-level information for a SQL Server record by the master database.
* The dbid (database id ) of master is 1.
* The master database has SIMPLE RECOVERY MODEL.
* It is a very important database and we must have the backup.
* Without the master database the server can't be started.

### Assume the master database files are missing or inaccessible, will SQL Server start or up?

**Answer:** No, SQL Server will not start because the master database is the important database and all the configuration and information needed to start the SQL Server is stored in the master database itself hence without the master database SQL server will not start.

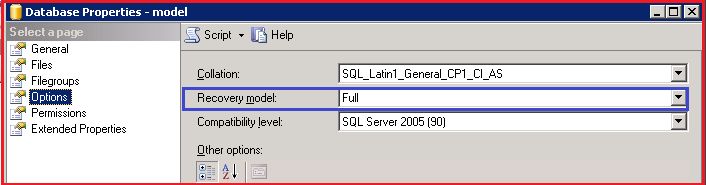
* Master database contains information about the server configuration. We can see the server configuration with the following query:  
  select \* from sys.sysconfigures;
* Master database contains information about all other databases and their location on SQL Server. We can see the information with executing the following query:  
  select \* from sys.sysdatabases; or sp\_helpdb
* Master database contains information about logins in SQL Server. The following is the query by which we can see it:  
  select \* from sys.syslogins;
* Master database also contains information about users on SQL Server. The following is the query to see user details:  
  select \* from sys.sysusers;
* Master and mastlog are the logical file names of master database.  
  master.mdf ( data file ) and mastlog.ldf are the physical files of master database.

Query to see the physical file location of the master database:

1. SELECT name, physical\_name FROM sys.database\_files;

Master Database in SQL Server

## Model Database in SQL Server

* The Model database acts as a template database used in creation of new databases.
* The dbid of the model database is 3.
* By default the model database has FULL RECOVERY MODEL.  
    
  ****  
  **Figure 4:** Recovery Model
* We can take a backup of the model database.
* Modeldev and modellog are the logical file names of the model database.
* Model.mdf ( data file ) and modellog.ldf are the physical files of the model database.
* The same query can be use to see the physical file location of the model database:

1. SELECT name, physical\_name FROM sys.database\_files;



A user has created a new database. What will be the recovery model of that database?

**Answer:** Because the model database acts as a template database, when a user creates a new database it will inherit the property of the model database and as we know by default the recovery model of the model database is FULL (until or unless the user changes it), hence the new database is created by the user with FULL RECOVERY MODEL.

## MSDB Database in SQL Server

* A MSDB database stores information related to backups, SQL Server Agent information, SQL Server Jobs, alerts and so on.
* The Dbid of the msdb database is 4.
* The recovery model of a msdb database is SIMPLE.
* We can take backup of a msdb database.
* MSDBData and MSDBLog are the logical file names of a msdb database.
* MSDBData.mdf (data file) and MSDBLog.ldf are the physical files of a msdb database.
* The same query can be used to see the physical file location of a msdb database:

1. SELECT name, physical\_name FROM sys.database\_files;

MSDB database Table

## TempDB in SQL Server

* It stores temporary objects, like temporary tables, temporary Stored Procedures and temporary tables to store sorting and so on.
* The dbid of a temp database is 2.
* The recovery model of a temp database is SIMPLE.
* We can't take a backup of a tempdb.
* tempdev and templog are the logical file names of tempdb.
* tempdb.mdf (data file) and templog.ldf are the physical files of a tempdb.
* Same query can be use to see the physical file location of tempdb:

1. SELECT name, physical\_name FROM sys.database\_files;

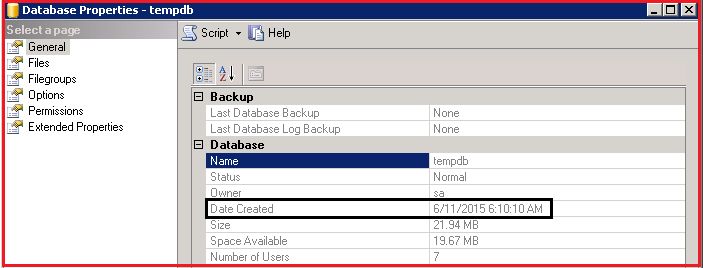
TempDB Table in SQL Server

### Why we can't take a backup of a temp database?

**Answer:** Temp databases, as the name says, are used to do temporary operations, such as tables, Stored Procedures and cursors. Once the operation is over it will be cleared and is minimally logged. A TempDB is recreated everytime SQL is started, so it is always has a clean copy of the database hence backup and restore operations are not allowed for a TempDB.

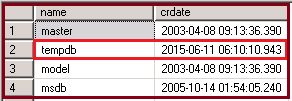
### How you will check to determine if the SQL Server is restarted?

**Answer:** Check the creation date of the tempdb, if it is new it means SQL Server is started.



We can execute the following query also to check the tempdb creation date:

1. select name, crdate from sys.sysdatabases;



## Resource Database in SQL Server

* It is a read-only database hidden from the user. It contains all the system objects that are included with the SQL Server.
* The Dbid of the resource database is 32767.
* The Resource database helps when we do a SQL Server upgrade.
* We can't see the resource database in SQL Server Management Studio but we can see its database file at the OS level by the name **mssqlsystemresource.mdf** and **mssqlsystemresource.ldf in the Binn folder of Program Files.**



We can see the location of the resource database file using the following query also:

1. Use master GO
2. SELECT
3. 'ResourceDB' AS 'Database Name',
4. NAME AS [Database File],
5. FILENAME AS [Database File Location]
6. FROM
7. sys.sysaltfiles
8. WHERE
9. DBID = 32767 GO

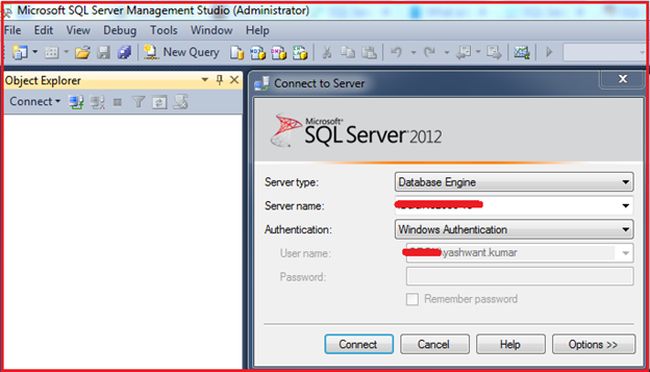
**What User Databases are:** User databases are databases created by the user themself.



**SQL Server Authentication Modes**

**What is Authentication ?**

Authentication is a process in which we need credentials, in other words username and word, to access the SQL Server.



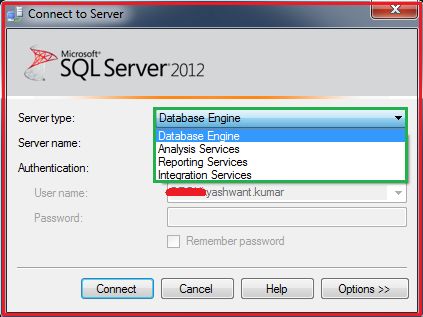
When you open SQL Server Management Studio (SSMS) for very first time you will get the following three things:

* Server Type
* Server Name
* Authentication

**1. Server Type**

There are the following four types of servers:

1. Database Engine: Used for storing, processing and securing data.
2. Analysis Services: Used for Online Analytical Processing and data mining functionality.
3. Reporting Services: Used for creating interactive, tabular, graphical, or free-form reports from relational, multidimensional, or XML-based data sources.
4. Integration Services: Used to do a broad range of data migration tasks. It is a platform for data integration and workflow applications.

****

**2. Server Name**

It can be any name of server by which a server can be identified.

**3. Authentication**

As we already discussed, it is a process in which we need credentials, in other words username and word, to access the SQL Server that is clearly visible in Figure 3.



**Types of Authentication in SQL Server**

There are basically the following two types of authentication in SQL Server:

1. Windows authentication
2. Mixed mode Authentication/SQL Server Authentication

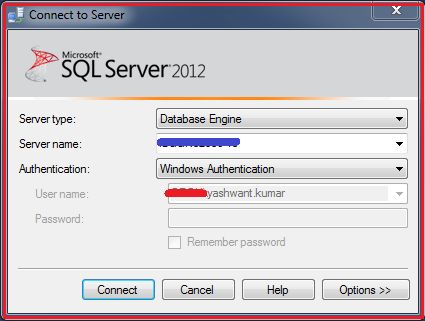
### **a. Windows Authentication**

Requires a valid Windows username and word to access the SQL Server.

### **b. Mixed mode Authentication**

A user can login either via SQL Server authentication or Windows authentication mode to connect to SQL Server.

Example of Windows Authentication Mode:



The following describes how to check that we are logged in with Windows Authentication Mode or Mixed Mode.

For this simply execute the following query in SSMS:

Use Master

GO

SELECT

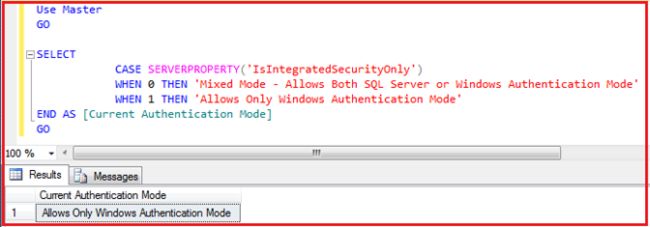
CASE SERVERPROPERTY('IsIntegratedSecurityOnly')

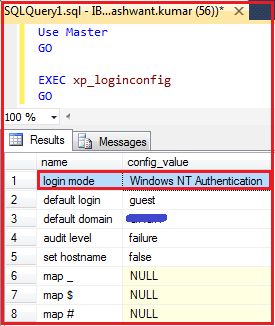
WHEN 0 THEN 'Mixed Mode - Allows Both SQL Server or Windows Authentication Mode'

WHEN 1 THEN 'Allows Only Windows Authentication Mode'

END AS [Current Authentication Mode]

GO

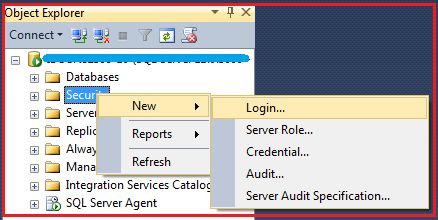


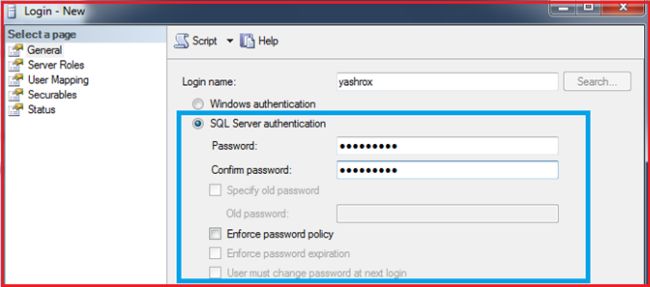


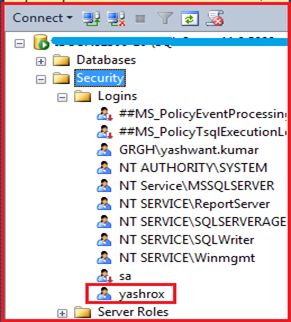
Remember that when only Windows Authentication is enabled, you can't login with SQL Server Authentication or Mixed Mode.

Let's make it clear with an example.

I have created a new login “yashrox” with SQL Authentication.

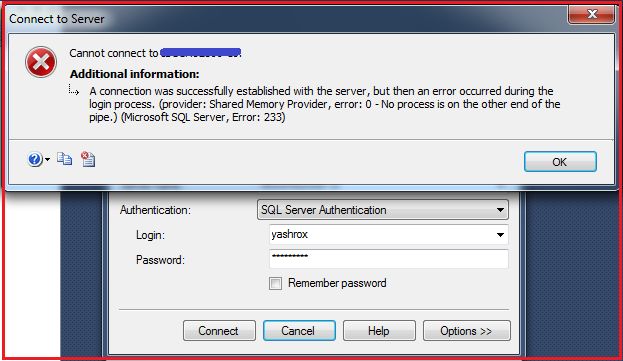




s

Now to check Mixed Mode / SQL Server Authentication we will log into SQL Server with the login "yashrox" that was created with SQL Server / Mixed Mode Authentication.





An error occurred when we tried to login with SQL Server Authentication / Mixed Mode Authentication. Why this happened is because Mixed Mode / SQL Server Authentication is not **enabled** by default as we saw in Figure 5 and Figure 6 also.

## Resolution

The resolution for this problem is to enable the Mixed Mode / SQL Server Authentication, so let's move ahead to enable the Mixed Mode/SQL Server Authentication.

## Enabling Mixed Mode/SQL Server Authentication

There are two ways to **enable** Mixed Mode/SQL Server Authentication mode.

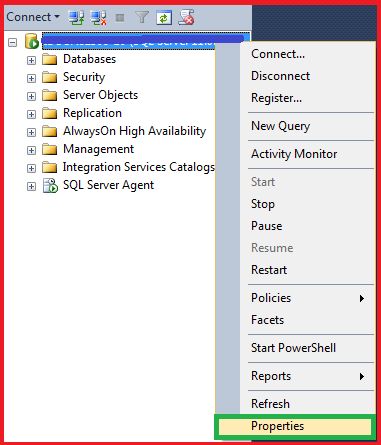
**First Method**

Step 1

Log into SQL Server with Windows authentication mode with the login name “XYZ\yashwant.kumar"

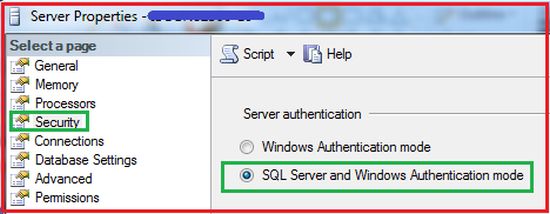
Step 2

Right-click on the Server and then click on properties as in the following:



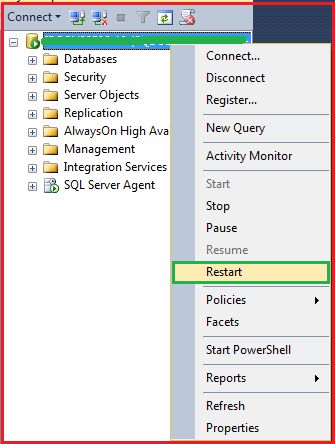
**Step 3**

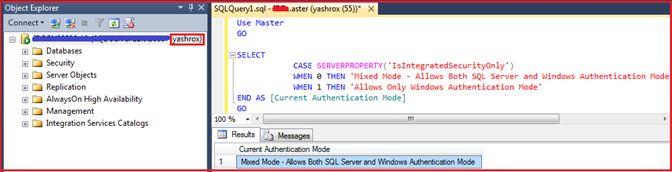
Click on security in the left pane and select SQL Server and Windows Authentication Mode and click OK to save.

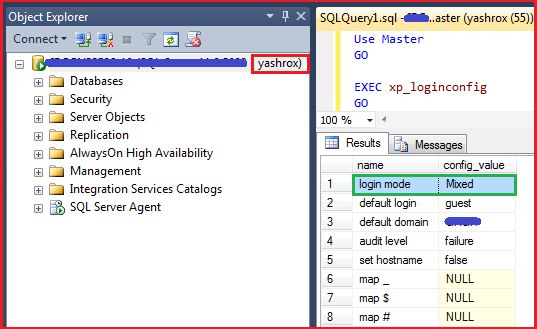


**Step 4**

Restart SQL Server and try to login with SQL Server Authentication.







**Figure 16:** Checking Authentication mode with other query

Wow! Now this time we are able to login with SQL Server Authentication / Mixed Mode and with both queries we are getting the login mode as Mixed Mode / SQL Server Authentication.

### **Second Method**

Enabling Mixed from Regedit/Registry.

**Step 1**

Press the Windows key + R to open the Run box. Type regedit and press Enter.

**Step 2**

Navigate to the registry location HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Microsoft SQL

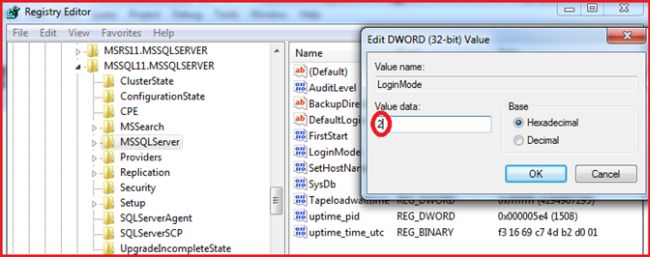
Server\MSSQL11.MSSQLSERVER\MSSQLServer.

In the right panel, change the LoginMode from 1 to 2.

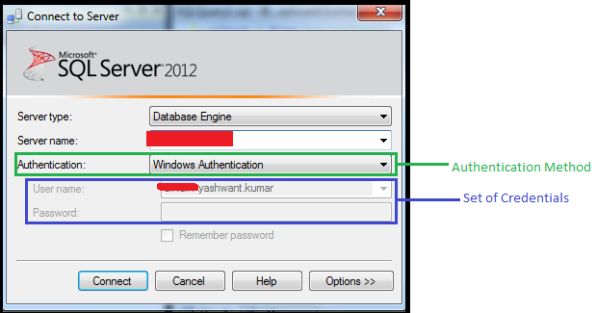
1. Windows authentication Only.
2. Mixed mode.

**Step 3**

Restart your SQL Server instance and you can then connect to the server using SQL Server Authentication.

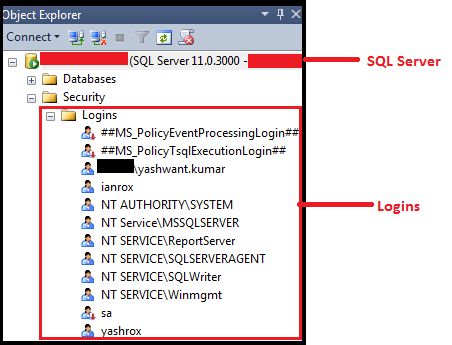


**Login**  
A login is simply a set of credentials to gain access to SQL Server that requires proper authentication.  
  
Users need login to connect to SQL Server or we can say logins are associated to a user and the association is done by a Security Identifier (SID). We can create a login based on a Windows Authentication (like a domain user or a Windows domain group) or we can create a login with SQL Server Authentication.



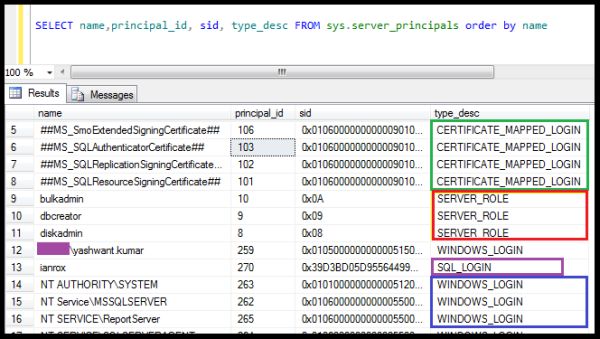
You can refer to Abhishek Yadav's article on “[Logins and Users in SQL Server](http://www.c-sharpcorner.com/UploadFile/ff0d0f/logins-and-users-in-sql-server/)“ for more information in which he explained very well about SQL Server logins and users.  
  
**Properties of logins**

**1)** It is a server level entity.



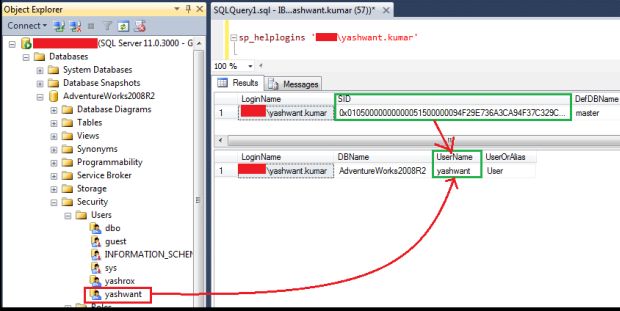
2) It is a set of credentials, in other words, username and password are required.

3) Login information is stored in the sys.syslogins / sys.server\_principals table in the master database, the output in my case is of 27 rows, I have taken only those rows that covered all the types of logins, for example Windows, SQL, certificate mapped and server role, so don't be confused here.

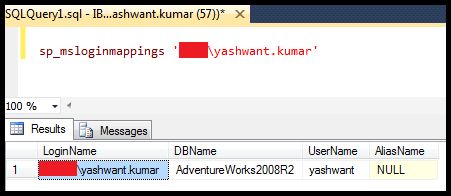


**sys.syslogins:** It is a compatibility view to support 2000 databases and hence not recommended for use from SQL Server 2005 onwards.  
  
This includes logins that are Windows, certificate mapped, or SQL authentication based.  
**sys.server\_principals:** It is recommended to use this view from SQL Server 2005 onwards.  
Apart from Windows, certificate mapped or SQL authentication based logins it also includes server role information.

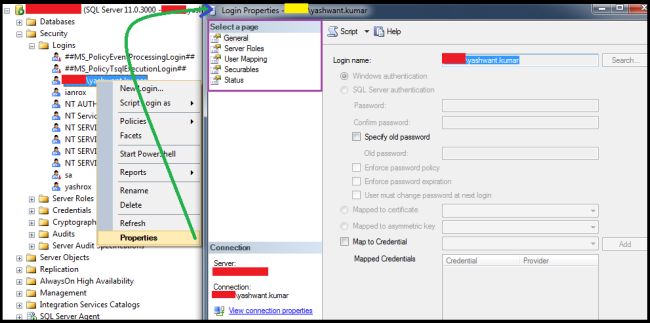
4) Logins are associated with users by a Security Identifier (SID), in other words we can say if a database user exists but there is no login associated, then in this case the user will not able to log into SQL Server. We can check it by Stored Procedure *sp\_helplogins or sp\_msloginmappings.*



"0x01050000000000051500000094F29E736A3CA94F37C329CD7B540000" using sp\_helplogins in a query window and in the Object Explorer also.  
  
**or**



**Exploring More about Logins**

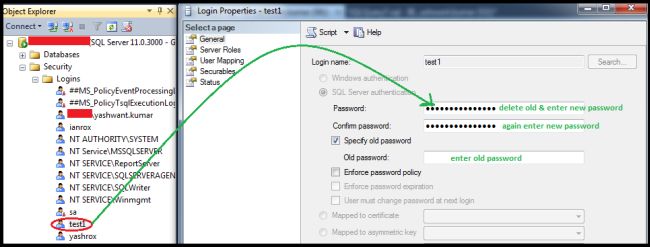


The Login Properties page is divided into the following five sections:

1. General
2. Server Roles
3. User Mapping
4. Securables
5. Status

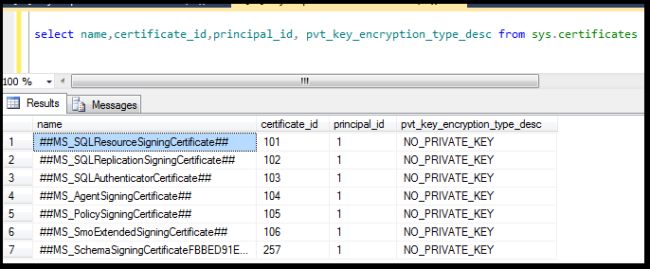
**1. General**  
When we open the General tab of the login properties page we can see the following information:

1. **Login Name:** Information about the name of the login including the authentication types information.
2. **Password:** It is a password for the login name.
3. **Specify Old Password:** If we want to change the password. The following is the procedure to change the password:
   1. Right-click on login test1 then click on properties.
   2. Delete the old password and enter a new password in the password box.
   3. Again enter the new password in the confirm password box.
   4. Enable the check box to specify the old password and input the old password.
   5. Click OK. It's done now. Login with new password and check that it's done.



* Enforce password policy: enable this checkbox, if you want to enforce the password policy.
* Enforce Password Expiration: enable this checkbox if you want a password expiration time for the login.
* Mapped to Certificate: Certificates are the way to encrypt with a digitally signed object.   
  The certificate provides database-level security control.  
    
  We can execute sys.certificates views to see the certificates, for example:

select name,certificate\_id,principal\_id, pvt\_key\_encryption\_type\_desc from sys.certificates;

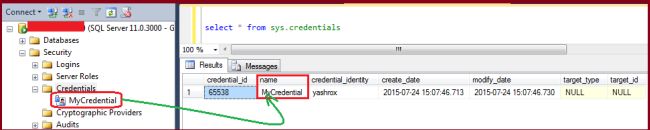


Mapped to Asymmetric Key: These are the keys in SQL Server for encrypting and decrypting data that is being transmitted from one place to another.  
  
We can see the asymmetric keys by querying the view for sys.asymmetric\_keys, for example:

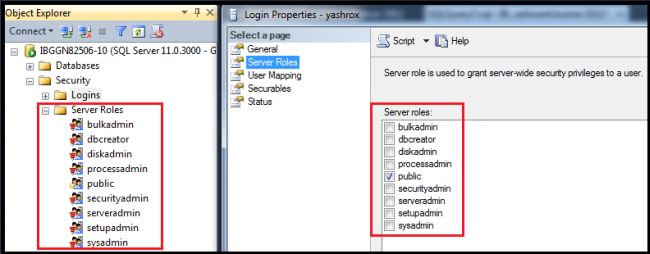
1. select name,principal\_id,pvt\_key\_encryption\_type\_desc,algorithm\_desc from sys.asymmetric\_keys



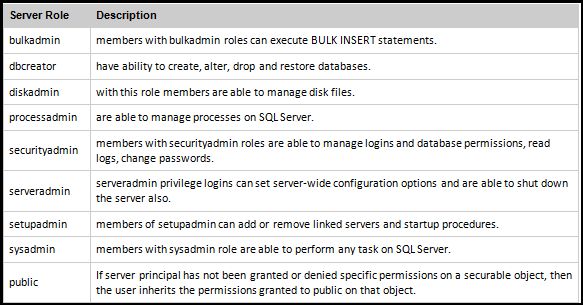
1. **Note:** SQL Server suports three algorithms for asymmetric key encryption: RSA\_512, RSA\_1024 & RSA\_2048.
   * RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977.
   * All three RSA\_512, RSA\_1024 & RSA\_2048 algorithms are all based on the RSA [cryptosystem](https://en.wikipedia.org/wiki/RSA_%28cryptosystem%29). The difference in these RSA is the key length: 512, 1024 or 2048 bits. The longer the key (the more bits it has) results in more security of the encrypted data is and more bits also means that more CPU resources will be used.
2. Map to Credential: A credential is a record that contains the authentication information required to connect to a resource outside SQL Server. ( Source: msdn.microsoft.com.)  
     
   **Note:** A single credential can be mapped to multiple SQL Server logins. However, a SQL Server login can be mapped to only one credential.  
     
   We can see the credentials using sys.credentials view, for example:
   * select \* from sys.credentials

  
**Figure10:** Querying sys.credentials view

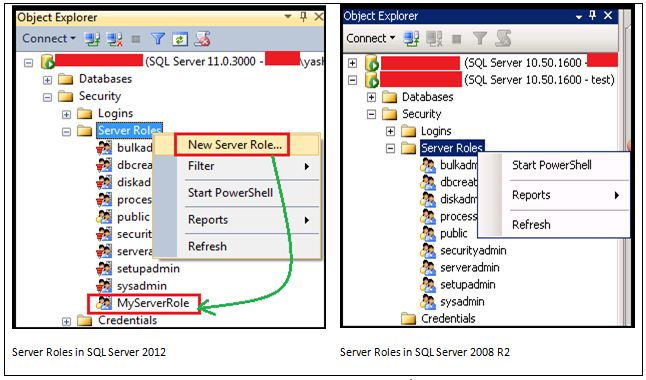
**2. Server Roles**Used to grant server-wide security privileges to a user. Figure 11 shows the various server roles available for various tasks. There are 9.



**Figure 11:** Illustration of Server roles in login properties dialogue



**Figure 12:** Types of server roles & their description



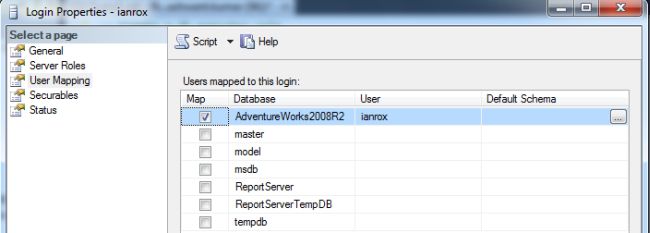
**Figure 13:** Difference between Server Roles in SQL Server 2012 & SQL Server 2008/2008 R2

* In SQL Server 2012 there is a new feature introduced in which we can create a server role that was not possible in SQL Server 2008/2008 R2. This is a difference also, that is clearly visible and to make this difference clear Microsoft put a Red-Pin with a fixed server role. These server roles can't be modified when the user-defined server role “MyServerRole” can be modified.
* When we create a user-defined server role we can add only server-level permissions to that user-defined server role. We can list server-level permissions using the following statement:
  1. SELECT \* FROM sys.fn\_builtin\_permissions('SERVER') ORDER BY permission\_name;

**3. User Mapping**  
  
There are the following two further options available:

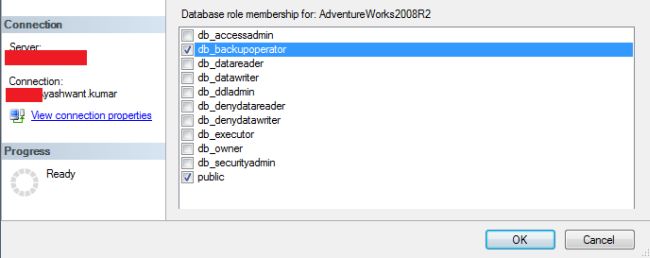
1. Users mapped to this login
2. Database role membership for "DatabaseXYZ"

**a. Users mapped to this login:** In this we can specify that the specific login can access which database.



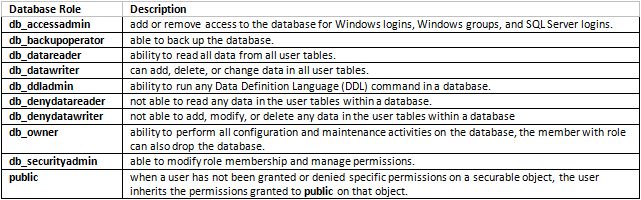
**Figure 14:** Illustration of user mapped to this login

In this example, it is clear that login "ianrox" will be able to access the Adventureworks2008R2 database. Now here the question is whether "ianrox" can perform the tasks on the AdventureWorks2008R2 database. For this database roles are relevant.  
  
**b. Database role membership for "DatabaseXYZ":** in this section, we can specify the activities a login can perform. In the following screenshot, we provided the database role "db\_backupoperator". It means that "ianrox" can perform backup activity for the Advetureworks2008R2 database.  
  
There are 10 database roles in SQL Server.



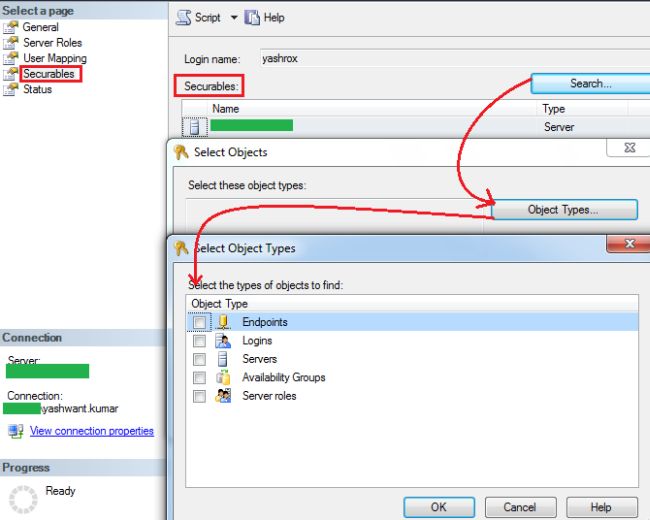
**Figure 15:** Showing database roles

Database roles & their description: The table in Figure 16 describes the database roles.



**Figure 16:** Showing database roles & their description

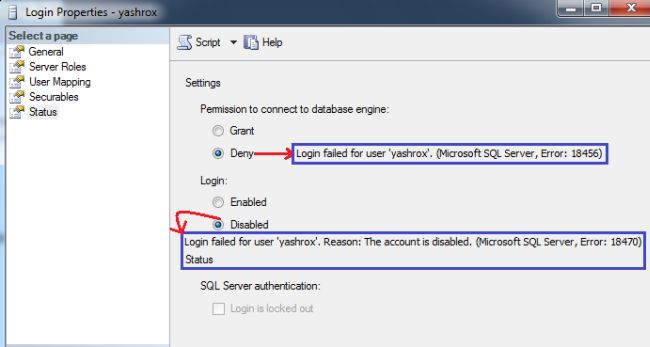
**Note:** a public database role cannot be dropped.  
 **4. Securable**Securables are the resources that we can assign permissions, either at the server level that includes resources like Endpoints, Logins, Server Roles and Databases or at database-level that includes resources like Users, Database Roles, Certificates and Schemas.  
  
SQL Server has securable at Schema level also that are called schema scope securable. Tables, Views, Procedures and so on are example of a schema scope securable.



**Figure 17:** Showing Securables from login properties page

**5. Status**In this page we can set the permission to connect to a database engine for login or we can enable/disable a login. But what is the difference between grant/deny and enable/disable login?  
  
To check the differences I created the following **four** cases:

1. Select DENY and ENABLED login
2. Select GRANT and DISABLED login
3. Select DENY and DISABLED login
4. Select GRANT and ENABLED login ( It is by default selected when we create new login )



**Figure 18:** Working on the Status page of Login Properties

**i. Select DENY and ENABLED login:** When we select this combination we will get the error message “Login failed for user 'yashrox'. (Microsoft SQL Server, Error: 18456)".  
  
**ii. Select GRANT and DISABLED login:** When we select this combination we will get the error message “Login failed for user 'yashrox'. Reason: The account is disabled. (Microsoft SQL Server, Error: 18470)".  
  
**iii. Select DENY and DISABLED login:** When we select this combination we will get the same error message as in the second combination, “Login failed for user 'yashrox'. Reason: The account is disabled. (Microsoft SQL Server, Error: 18470)".  
  
**iv. Select GRANT and ENABLED login:** It's a default method when a login is created, from this combination we will successfully connect to a database when the right login name & password are provided.  
  
**From all these cases I concluded the following things:**

* DENY CONNECT SQL will **not** block members of the **sysadmin** fixed server role from logging in because deny do **not** apply to sysadmins. You can check it provide **sysadmin** role to that login and try with first case in other words **deny and enabled**. You will be able to login in this case.
* Connecting to SQL Server is a two-step process as in the following:  
    
  First, the login must prove its identity with the correct login name & password. Second, after verifying the identity for the effective permission (grant/deny) will check. If login has connect SQL permission, in other words Grant, then they will be able to connect to SQL Server.  
    
  You can check it with the third case where we take deny & disable but we get the error “Login failed for user ‘yashrox’. Reason: The account is disabled". (Microsoft SQL Server, Error: 18470). It clearly shows that the first login is authenticated then connect SQL permission.