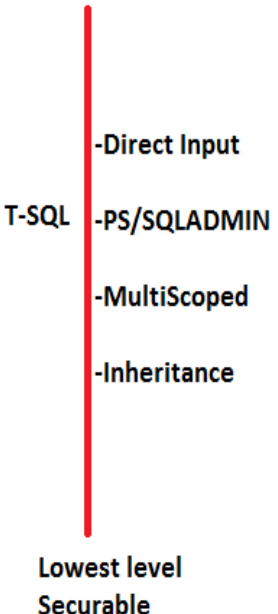
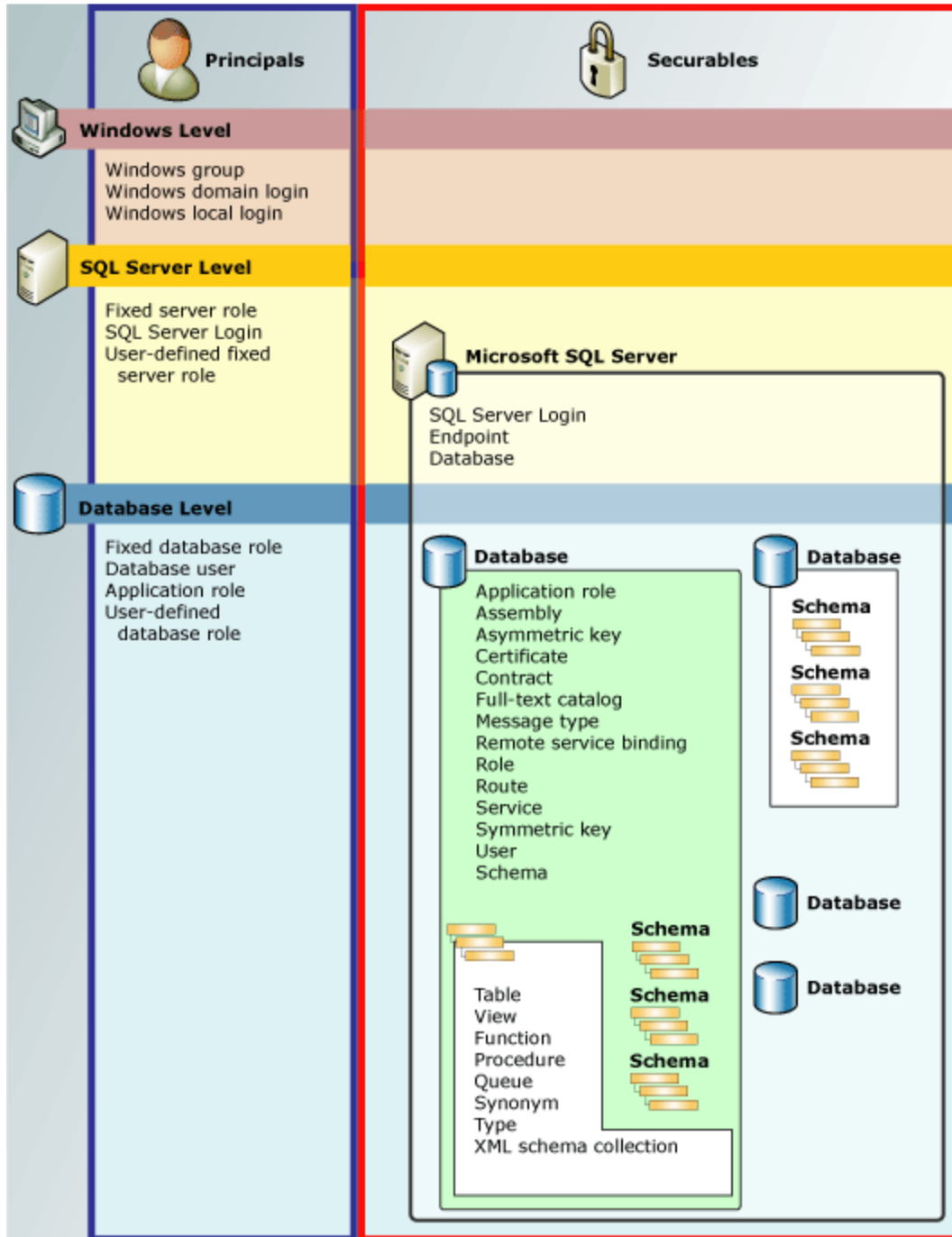


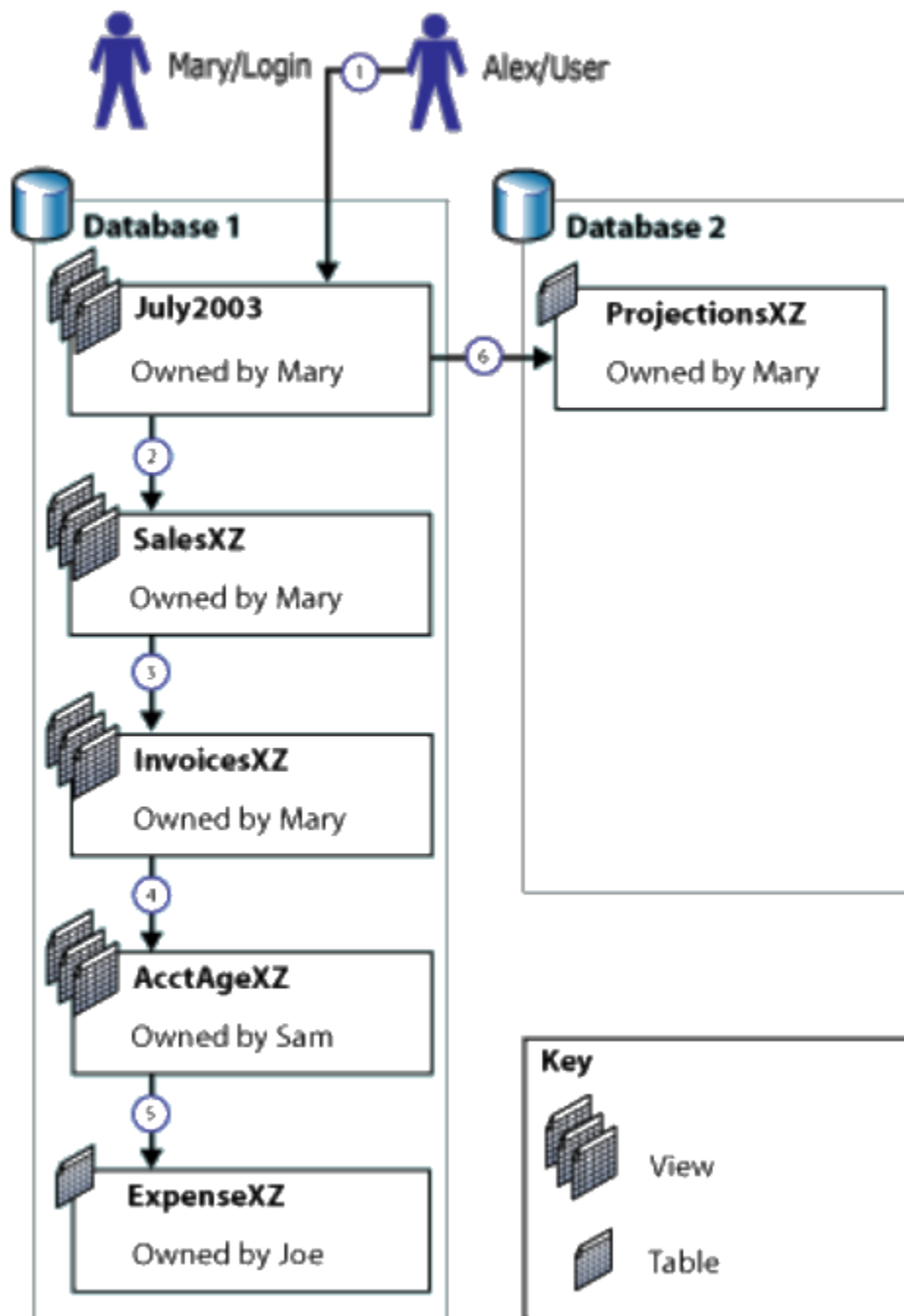
Securable's - <http://technet.microsoft.com/en-us/library/ms190401.aspx>

Think like an Admin.... (Goal – Exam: 70-462)	
<ol style="list-style-type: none"> <li>1. What is the task to perform or Question to ask?</li> <li>2. What Tools will I use? (PS/SSMS – other – ch3-p85)(T-SQL)</li> <li>3. Is it about-: (Permission Scope 3 levels – ch20-p694) <ol style="list-style-type: none"> <li>a. The server “Engine”?</li> <li>b. Schemas? (Permissions &amp; Access p-897) <ol style="list-style-type: none"> <li>i. Users/Groups/Roles (Ownership Chaining p-698)</li> </ol> </li> <li>c. The Database Itself?</li> </ol> </li> <li>4. How will I get “Results”? <ol style="list-style-type: none"> <li>a. Statement Permissions (695) <ol style="list-style-type: none"> <li>i. Alter server, database or schema “Securables”</li> <li>ii. Ex: GRANT CREATE Table to User01</li> </ol> </li> <li>b. Object Permissions (696) <ol style="list-style-type: none"> <li>i. Ability to access data within the structure. 12 of them.</li> <li>ii. Ex: GRANT SELECT ON Marketing WITH GRANT OPTION</li> </ol> </li> </ol> </li> </ol>	<p><b>Principal wants a Securable</b></p>  <p><b>Lowest level Securable</b></p>
<ol style="list-style-type: none"> <li>1. What T-SQL statements can I use in a “Permission Scope”? (694)</li> <li>2. Is there any inheritance?</li> <li>3. What is the "Least Privileged" path?</li> </ol>	

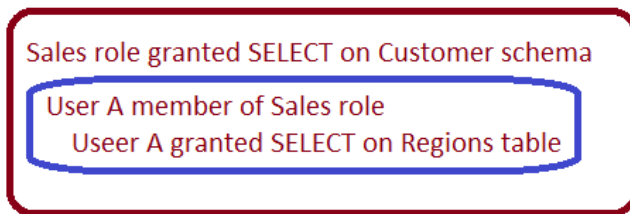
**Data Control Language (DCL)** - It is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it. These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE.

Configuring Permissions on Database Objects - <http://msdn.microsoft.com/en-us/library/f964b66a-ec32-44c2-a185-6a0f173bfa22>

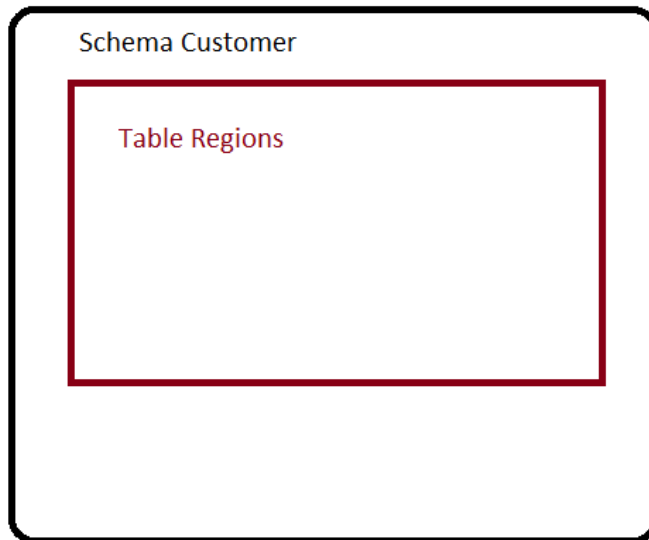


**Security**

## Sale Role



## SalesDB



You develop three Microsoft SQL Server 2012 databases named Database1, Database2, and Database3. You have permissions on both Database1 and Database2. You plan to write and deploy a stored procedure named `dbo.usp_InsertEvent` in Database3. `dbo.usp_InsertEvent` must execute other stored procedures in the other databases. You need to ensure that callers that do not have permissions on Database1 or Database2 can execute the stored procedure. Which Transact-SQL statement should you use?

- ☐ A. USE Database1
- ☐ B. EXECUTE AS CALLER
- ☒ C. EXECUTE AS OWNER
- ☐ D. USE Database2

You administer a SQL Server 2012 server that contains a database named **SalesDb**. SalesDb contains a schema named **Customers** that has a table named **Regions**. A user named **UserA** is a member of a role named **Sales**. UserA is granted the Select permission on the Regions table. The Sales role is granted the Select permission on the Customers schema. You need to ensure that the Sales role, including UserA, is **disallowed** to select from any of the tables in the Customers schema. Which Transact-SQL statement should you use? (why is User A also disallowed )

- A. REVOKE SELECT ON Schema::Customers FROM UserA
- B. DENY SELECT ON Object::Regions FROM UserA
- C. EXEC sp\_addrolemember 'Sales', 'UserA'

- D. DENY SELECT ON Object::Regions FROM Sales
- E. REVOKE SELECT ON Object::Regions FROM UserA
- F. DENY SELECT ON Schema::Customers FROM Sales
- G. DENY SELECT ON Schema::Customers FROM UserA
- H. EXEC sp\_droprolemember 'Sales', 'UserA'
- I. REVOKE SELECT ON Object::Regions FROM Sales
- J. REVOKE SELECT ON Schema::Customers FROM Sales

Correct Answer: F

Question	Action	Permission	Operator	Securable	Action	Securable
44	DENY	SELECT	ON	Schema::Customer	FROM	Sales
45	DENY	SELECT	ON	Schema::Customer	FROM	UserA
46	REVOKE	SELECT	ON	Object::Regions	FROM	UserA
47	DENY	SELECT	ON	Object::Regions	FROM	Sales

Granting a user access to a database involves three steps. First, you create a login. The login lets the user connect to the SQL Server Database Engine. Then you configure the login as a user in the specified database. And finally, you grant that user permission to database objects.

#### Grant a Permission to a Principal

The entity that receives permission to a securable is called a principal. The most common principals are logins and database users. Access to Securables is controlled by granting or denying permissions, or by adding logins and user to roles which have access. For information about controlling permissions, see [GRANT \(Transact-SQL\)](#), [REVOKE \(Transact-SQL\)](#), [DENY \(Transact-SQL\)](#), [sp\\_addrolemember \(Transact-SQL\)](#), and [sp\\_droprolemember \(Transact-SQL\)](#).

- Grant permission to roles, instead of individual logins or users. When one individual is replaced by another, remove the departing individual from the role and add the new individual to the role.
- Configure similar Securables (tables, views, and procedures) to be owned by a schema, then grant permissions to the schema. For example, the payroll schema might own several tables, views, and stored procedures. By granting access to the schema, all the necessary permissions to perform the payroll function can be granted at the same time.
- A credential is a record that contains the authentication information (credentials) required to connect to a resource outside SQL Server.

#### Permission Hierarchy

Permissions have a parent/child hierarchy. That is, if you grant **SELECT** permission on a database, it 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; that is, if you grant **SELECT** permission on a database, it includes **SELECT** permission on all (child) schemas, and all (grandchild) tables, and all views.

Permissions also have covering permissions. The **CONTROL** permission on an object, normally gives you all other permissions on the object.

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 table (**Region**), in a schema (**Customers**), in a database (**SalesDB**).

- **CONTROL** permission on table **Region** includes all the other permissions on the table **Region**, including **ALTER**, **SELECT**, **INSERT**, **UPDATE**, **DELETE**, and some other permissions.
- **SELECT** on the **Customers** schema that owns the **Region** table includes the **SELECT** permission on the **Region** table.

So **SELECT** permission on the **Region** table can be achieved through any of these three statements:

- **GRANT SELECT ON OBJECT::Region TO Ted**
- **GRANT CONTROL ON OBJECT::Region TO Ted**
- **GRANT SELECT ON SCHEMA::Customers TO Ted**
- **GRANT CONTROL ON SCHEMA::Customers TO Ted**
- **GRANT SELECT ON DATABASE::SalesDB TO Ted**
- **GRANT CONTROL ON DATABASE::SalesDB TO Ted**

### Grant the Least Permissions

The first permission listed above (**GRANT SELECT ON OBJECT::Region TO Ted**) is the most granular, that is, that statement is the least permission possible that grants the **SELECT**. No permissions to subordinate objects come with it. Always grant the least permission possible, but grant at higher levels in order to simplify the granting system. So if Ted needs permissions to the entire schema, grant **SELECT** once at the schema level, instead of granting **SELECT** at the table or view level many times. The design of the database has a great deal of impact on how successful this strategy can be. This strategy will work best when your database is designed so that objects needing identical permissions are included in a single schema.

### List of Permissions

SQL Server 2008 R2 has 195 permissions. SQL Server Code-named 'Denali' has 214 permissions. The following graphic shows the permissions and their relationships to each other. Some of the higher level permissions (such as **CONTROL SERVER**) are listed many times.

[5710.Permissions\\_Poster\\_2008\\_R2\\_Wiki.pdf](#)

### Permissions vs. Fixed Server and Fixed Database Roles

The permissions of the fixed server roles and fixed database roles are similar but not exactly the same as the granular permissions. For example, members of the **sysadmin** fixed server role have all permissions on the instance of SQL Server, as do logins with the **CONTROL SERVER** permission. But granting the **CONTROL SERVER** permission does not make a login a member of the **sysadmin** fixed server role, and making adding a login to the **sysadmin** fixed server role does not explicitly grant the login the **CONTROL SERVER** permission. Sometimes a stored procedure will check permissions by checking the fixed role and not checking the granular permission. For example detaching a database requires membership in the **db\_owner** fixed database role. The equivalent **CONTROL DATABASE** permission is not enough. These two systems operate in parallel but rarely interact with each other. Microsoft recommends using the newer, granular permission system instead of the fixed roles whenever possible.

### Monitoring permissions

The following views return security information.

- The logins and user-defined server roles (available in SQL Server Code-named 'Denali') on a server can be examined by using the sys.server\_principals view.
- The users and user-defined roles in a database can be examined by using the sys.database\_principals view.
- The permissions granted to logins and user-defined fixed server roles can be examined by using the sys.server\_permissions view.
- The permissions granted to user and user-defined fixed database roles can be examined by using the sys.database\_permissions view.
- Database role membership can be examined by using the sys.database\_role\_members view.
- Server role membership can be examined by using the sys.server\_role\_members view.
- For additional security related views, see [Security Catalog Views \(Transact-SQL\)](#).

The following statements return useful information about permissions.

To return the explicit permissions granted or denied in a database, execute the following statement in the database.

```
SELECT
perms.state_desc AS State,
permission_name AS [Permission],
obj.name AS [on Object],
dPrinc.name AS [to User Name],
sPrinc.name AS [who is Login Name]
FROM sys.database_permissions AS perms
JOIN sys.database_principals AS dPrinc
ON perms.grantee_principal_id = dPrinc.principal_id
JOIN sys.objects AS obj
ON perms.major_id = obj.object_id
LEFT OUTER JOIN sys.server_principals AS sPrinc
ON dPrinc.sid = sPrinc.sid
```

To return the members of the server roles, execute the following statement.

```
SELECT sRole.name AS [Server Role Name] , sPrinc.name AS [Members]
FROM sys.server_role_members AS sRo
JOIN sys.server_principals AS sPrinc
ON sRo.member_principal_id = sPrinc.principal_id
JOIN sys.server_principals AS sRole
ON sRo.role_principal_id = sRole.principal_id;
```

To return the members of the database roles, execute the following statement in the database.

```
SELECT dRole.name AS [Database Role Name], dPrinc.name AS [Members]
FROM sys.database_role_members AS dRo
JOIN sys.database_principals AS dPrinc
ON dRo.member_principal_id = dPrinc.principal_id
```



```
JOIN sys.database_principals AS dRole
ON dRo.role_principal_id = dRole.principal_id;
```

<http://blogs.msdn.com/b/sqlsecurity/archive/2011/08/25/database-engine-permission-basics.aspx>

Database Engine permissions are managed at the server level through logins and fixed server roles, and at the database level through database users and user-defined database roles.

### Return the complete list of grantable permissions

```
SELECT * FROM fn_builtin_permissions(default);

GO
```

Backup Restore Models - <http://technet.microsoft.com/en-us/library/ms187510.aspx>

1. What is the Governance plan for "Point in Time" Recovery?
2. What is the Transaction Log requirements?

**Recovery model** - A database property that controls transaction log maintenance on a database. Three recovery models exist: simple, full, and bulk-logged. (D=Daily – S/W=Sun/Wed – 24hr clock)

Dbase	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Sales	Full: 24hr- S/W	Diff: D- 2hrs T-	Diff: D- 2hrs T-	Full: 24hr- S/W	Diff: D- 2hrs T-	Diff: D- 2hrs T-	Diff: D- 2hrs T-
	Diff: D- 2hrs	Log:D- 0.15hrs	Log:D- 0.15hrs	Diff: D- 2hrs	Log:D- 0.15hrs	Log:D- 0.15hrs	Log:D- 0.15hrs
	T- Log:D- 0.15hrs			T- Log:D- 0.15hrs			
HR							
Marketing							