**Understanding and dealing with orphaned users in a SQL Server database**

[**http://www.mssqltips.com/tip.asp?tip=1590**](http://www.mssqltips.com/tip.asp?tip=1590)

Written By: [Atif Shehzad](http://www.mssqltips.com/author.asp?authorid=23) -- 9/24/2008

Parte inferior do formulário

**Problem**  
Attaching and restoring databases from one server instance to another are common tasks executed by a DBA. After attaching or restoring of a database, previously created and configured logins in that database do not provide access. The most common symptoms of this problem are that the application may face login failed errors or you may get a message like the user already exists in the current database when you try to add the login to the database.  This is a common scenario when performing an attach or a restore, so how do you resolve this?

**Solution**When a database is moved from one server to another server the login ids stored in the master database do not align with the login ids stored in each user database.  As mentioned above some of the errors you may see after attaching or restoring a database include:

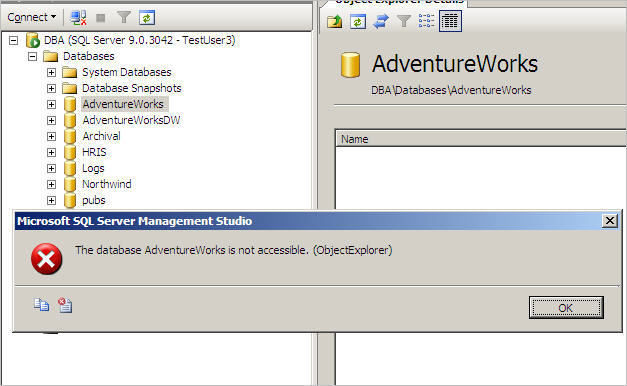
*Msg 229, Level 14, State 1  
%s permission denied on object %.\*s, database %.\*s, owner %.\*s*

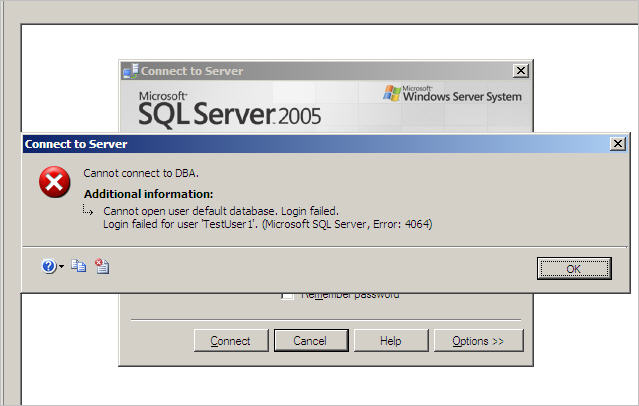
or

*Microsoft SQL-DMO (ODBC SQLState: 42000) Error 15023: User or role '%s' already exists in the current database.*

Without proper understanding and appropriate planning you may face this problem.  You could delete and recreate the user, but you will loose all of the configured permissions. So a proper linking mechanism is required, so that permissions are retained.

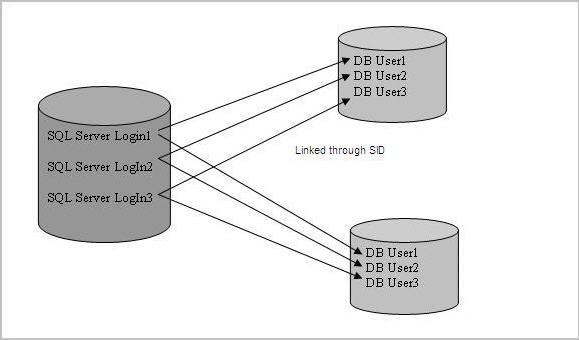
Some more possible error messages that you may see include





Before getting to the solution for this problem, it would be better to have a glance at the backend issue. SQL Server logins, which are stored in the master database are mapped against individual databases. SQL Server logins access individual databases using a database user that is mapped to the appropriate SQL Server login. There are two exceptions, the guest account and Microsoft Windows group memberships. The SQL Server 2005 logins on a server instance are visible in the***sys.server\_principals*** system catalog view and the ***sys.syslogins*** view. For SQL Server 2000 you may get SQL Server login information in ***sysxlogins*** table.

On the other hand the mapping information to a database user is stored within the database in the system table ***sysusers***. It includes the name of the database user and the SID of the corresponding SQL Server login. The permissions of this database user are used for authorization purposes in the database.



So we can say that every time we create a SQL Server login, it will be seen in the ***sys.server\_principals*** system catalog view or ***sys.syslogins*** views on SQL Server 2005 or in the ***sysxlogins*** table for SQL Server 2000. The entries inthe ***sysusers*** table of a database are linked to SQL Server logins shown in the above picture. This link is created through a column named ***SID.***

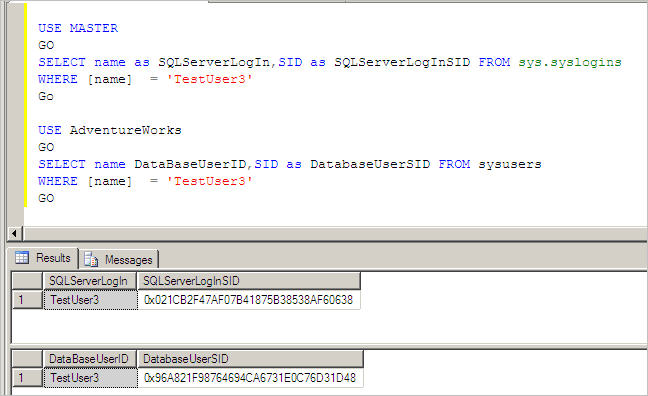
If we move our database to another SQL Server instance through any process, the new server might or might not have the same logins and the SIDs of these logins would probably be different from the SIDs of these logins in the original server. What this means is that, the **sysusers** table in the moved database has SIDs that are not matched with the login info in the master database on this new server. Therefore we get orphaned users.

As an example, I have created and configured four users with permissions in the [AdventureWorks](http://www.codeplex.com/MSFTDBProdSamples/Release/ProjectReleases.aspx?ReleaseId=4004) database. The users are TestUser1, TestUser2, TestUser3 and TestUser4. When I restored the backup of this database onto another SQL Server 2005 instance, although the users were present in the AdventureWorks database and the logins existed on the new server, but none of these logins had access to the newly restored database.

So keeping this scenario in mind, let us run some queries to see the difference between the SQL Server login SIDs (if SQL Server login is present) and database user SID for TestUser3.

|  |
| --- |
| **Script to view difference in SID** |
| USE MASTER GO  SELECT name as SQLServerLogIn,SID as SQLServerSID FROM sys.syslogins WHERE [name] = 'TestUser3' GO  USE AdventureWorks GO  SELECT name DataBaseID,SID as DatabaseSID FROM sysusers WHERE [name] = 'TestUser3' GO |

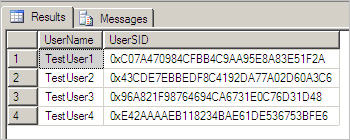
The result below shows the SID of the SQL Server login and that of the user database id differ and this is what causes the problem.



Now that we have a better understanding of the problem, it is time to get to some useful commands for analysis and a solution.

I have restored the AdventureWorks database from one instance to another with the four above users. Now to analyze how many orphaned users there are in my restored database, I will run the following T-SQL command which produces a listing of all the orphaned users and in our case all four users are orphaned.

|  |
| --- |
| Command to generate list of orphaned users |
| USE adventureWorks GO  sp\_change\_users\_login @Action='Report' GO |



Now that we have the list of the orphaned users we can begin to fix the problem.  To overcome this problem, you need to link the SIDs of the users (from sysusers) to the SIDs of the valid logins in the master database. The following command remaps the server login account specified by TestUser1 with the database user specified by TestUser1.

|  |
| --- |
| Command to map an orphaned user |
| USE AdventureWorks GO  sp\_change\_users\_login @Action='update\_one',  @UserNamePattern='TestUser1',  @LoginName='TestUser1' GO |

Or if you are sure that SQL Server login name is the same as the orphaned user of database being mapped, then you may use a shorter command such as this for TestUser2..

|  |
| --- |
| Command to map an orphaned user |
| EXEC sp\_change\_users\_login 'Auto\_Fix', 'TestUser2' GO |

Both commands will map the users to the logins and they will be not orphaned any longer.

If a login name does not exists, you would have to create it first before doing the mapping.  A quick way to do this is to use the following command which will create the login and then map the login to the user.

|  |
| --- |
| Command to map an orphaned user to a login that is not present but will be created |
| EXEC sp\_change\_users\_login 'Auto\_Fix', 'TestUser3', null,'pwd' GO |

**Summarizing the T-SQL Used**In the above process, the stored procedure **sp\_change\_users\_login** is used. The variable **[ @Action ]** specifies the exact use of this stored procedure. It accepts a parameter as varchar(10) and can have one of the following values:

* If parameter is **Auto\_Fix,** database user is mapped with same named SQL Server login. It may also create login, if not present.
* If parameter is **Report**, it lists the orphaned users and their security identifiers (SID).
* If parameter is **Update\_One,** it links the specified database user to an existing SQL Server login.

**Some considerations**

* **sp\_change\_users\_login** requires membership in the db\_owner fixed database role. Only members of the sysadmin fixed server role can specify the **Auto\_Fix** option.
* When the orphaned user is mapped, the SID in the master database is allotted to orphaned user, so every time a DB is attached or restored the SID will differ between SQL Server login and database user.
* If you have different Server login names mapped to a database user then do not use the command with **Auto\_Fix** for linking
* A user may become orphaned also if the corresponding SQL Server login is dropped
* Although obvious, but it is good to mention, after re-linking the password of SQL Server login can be used by the database user.

**Next Steps**

* Now that you have a better understanding of this issue and how to resolve it using **sp\_change\_users\_login**, it would be a good practice to prepare a script for detection and linkage of orphaned users for all of your databases.