How to Copy User Databases from One Server to Another Server

# Create Stub User Databases

1. On the source server, execute the T-SQL script ‘**get sql server 2005 databases.sql**’ to produce a T-SQL script that will create stub (empty) databases that will have the same names, options and file structure as those on the source server
   1. The script has a single parameter, @databaseName, which can be used specify a single database or all online user databases; system databases ([master], [model], [msdb] and [tempdb]) are ignored if no value is specified
2. Review the output produced to ensure the correctness of the following:
   1. Databases to reproduce
   2. Paths for the data, log and full-text files; these should be appropriate to the destination server
   3. Database options, owner, and compatibility level; again, these should be appropriate to the destination server
3. On the destination server, execute the new script
   1. Save this script, its output/results and the results of (1) as artifacts

# Create User Logins

1. On the source server, execute the T-SQL script ‘**get sql server 2005 logins.sql**’ to produce a T-SQL script that will create server principals that have the same logon names, SIDs, passwords and default databases as those on the source server
   1. The script has several parameters that can be used to filter and control its output:
      1. @login\_name: use this to have the output reproduce all logins or a single login
      2. @include\_db: use this to have the output specify each login’s default database or use the destination server’s default (usually [master])
      3. @include\_role: use this to have the output reproduce each login’s system roles
      4. @include\_sqlAuth: use this to have the output reproduce logins that are SQL-Authenticated
      5. @include\_winAuth: use this to have the output reproduce logins that are Windows-Authenticated; server principals that are Windows Groups are also included when this is selected
      6. @debug: use this to have the script also output intermediate datasets in addition to the T-SQL script output
2. Review the output produced to ensure the correctness of the following:
   1. Server Principals to reproduce
   2. Server Roles to reapply
   3. If copying SQL-Authenticated server principals across SQL Server versions (e.g. 2005 to 2014), be aware that Microsoft have discontinued use of some older password hashing algorithms, so there is a chance that logins’ passwords are hashed using an algorithm that is no longer supported, which will cause SQL Server to throw an error when running the script on the destination server
3. On the destination server, execute the new script
   1. Save this script, its output/results and the results of (1) as artifacts
4. Apply any server-level rights on the destination server as required and appropriate

# Back Up User Databases

1. On the source server, execute the T-SQL script ‘**get script to back up all online user databases.sql**’ to produce a T-SQL script that will take full database backups of user databases on the source server
   1. The script has several parameters that can be used to filter and control its output:
      1. @exclude\_system: use this to avoid taking backups of the system databases ([master], [model], [msdb]); [tempdb] is always ignored
      2. @tailLog: use this to have the output perform a tail log back up (of each database’s transaction log), if set, or a full database back up, if not set; see the section about Tail Log backups for further information
      3. @copyOnly: use this to have the output perform copy-only backups (so as to not disrupt each database’s Log Sequence Chain)
      4. @debug: use this to have the script also include intermediate datasets in addition to the T-SQL script output
      5. @databaseFilter: use this table variable to have the script only back up specific databases
   2. The script uses the backup location configured on the source server to store the backup files produced by the output script
   3. The backup files’ path and names take the form “{backup\_path}{database\_name}\_backup\_{backupTime}.{extension}”, where:
      1. {backup\_path} is the server’s configured backup location
      2. {database\_name} is the name of the database being backed up
      3. {backupTime} is the local date and time that the (generator) script is being run on the source server, in the format {yyyyMMddHHmm}
      4. {extension} is “bak”, for full database backups, or “trn” for tail log backups
2. Review the output produced to ensure the correctness of the following:
   1. Databases to back up
   2. Type of backup to take
   3. Paths for the backup files
3. On the source server, execute the new script
   1. Save this script, its output/results and the results of (1) as artifacts
   2. The database files result set produced in (1) will be used along with the backup files’ path and names to construct the script to restore the backups to the destination

# Restore User Databases

1. Copy the backups from the source server to the destination server
2. Open the file ‘**convert backup paths into sql to list database name and backup path.xlsx**’ to construct the inputs for the script ‘**get script to restore databases' backups.sql**’
   1. Copy the path and names of the backup files into the ‘BackupFileName’ column in the sheet ‘back up files’, specifying whether or not to ‘KeepReplication’ (this should only be set to TRUE if there are replication objects in the source database that are desired in the destination database)
      1. The SQL output column in this sheet is the input to @backup\_files
   2. Copy the ‘database\_name’, ‘file\_id’, ‘logical\_name’, ‘type\_desc’, and ‘physical\_name’ data for the databases to be restored (from ‘**get script to back up all online user databases.sql**’) into the corresponding columns in the sheet ‘database files’
      1. The SQL output column in this sheet is the input to @database\_files
3. On the destination server, execute the script ‘**get script to restore databases' backups.sql**’ to produce a T-SQL script that will restore the user databases on the destination server
   1. The script has several parameters that can be used to control its outp ut:
      1. @backup\_files: use this to specify the databases, whether or not to keep replication objects, the type of backup file, and the path and name of the backup file for each database to be restored
      2. @database\_files: use this to specify the database files to be expected in the backups for each database to be restored
      3. @ownerLogin: use this to specify the login of the server principal that should be set as each database’s owner
      4. @withRecovery: use this to have the output recover the databases to be restored, if set, or leave each database in recovering state after the restore is complete, if not set; see the section on Tail Log Backups for further information
      5. @quiesce: use this to have the output set each database to single user mode (with immediate rollback) before beginning its restore; this is useful for ensuring that no other user is active in the databases to be restored
   2. The script examines the server’s configuration to determine the default location for data and log files
   3. Review the output produced to the ensure the correctness of the following:
      1. Databases to restore
      2. Paths for the data and log files
      3. Type of backup to restore
      4. Database owner’s principal name
      5. Recovery status
      6. Replication objects retention
4. On the destination server, disable maintenance plans or SQL Agent jobs that take backups or replicate data
5. On the destination server, execute the new script
   1. Save this script, its output/results and the results of (2) and (3) as artifacts
6. If copying databases across SQL Server versions (e.g. 2005 to 2014), on the destination server, review the databases’ compatibility levels and adjust as needed
7. Re-enable any maintenance plans or SQL Agent jobs disabled in (4)

# Tail Log Backups

1. To just copy a snapshot of a database, or databases, from one server to another, a single, simple, full backup of the database is sufficient.
   1. When generating the script to back up the source databases, set the following:
      1. @tailLog = 0
      2. @copyOnly = 1
   2. When generating the script to restore the database backups at the destination, set the following:
      1. @withRecovery = 1
2. However, when “final” copying databases during a migration, the tail of each database’s transaction log is needed to ensure that the most up-to-date version of the database is migrated; and so some adjustments to the process must be made:
   1. The steps described in the preceding sections to produce snapshots of the desired databases on the destination server can be executed in advance to facilitate testing, staging, etc.
   2. When ready for the final migration, perform the following:
      1. Re-copy the desired user databases from the source to the destination by repeating the steps to back up the user databases on the source server with the following exceptions
         1. When producing the database back up script, set @copyOnly = 0; this will begin a new Log Sequence Chain, of which the tail log backups taken later will be a part
         2. When producing the database restore script, set @withRecovery = 0; this will leave the destination databases in a restoring state onto which the tail log backups taken later can be restored
         3. Hold any post-restore adjustments until after the tail log backups taken later are restored
         4. This re-copy can be done prior to the migration window; with the caveat that the destination is essentially unusable until the migration is complete
      2. On the source server, execute the T-SQL script ‘**get scripts to alter database user mode to restricted-user and multi-user.sql**’ to produce a T-SQL script that will quiesce the source user databases
         1. The script has the following parameters to filter and control its output
            1. @mode: set this to N’restricted\_user’ to specify that the user databases should only be available to administrative users; single-user mode doesn’t work here b/c the scripts will be executed in multiple sessions on the destination server and ‘single\_user’ doesn’t allow this
            2. @databaseFilter: use this to specify a single database or all online user databases; system databases ([master], [model], [msdb], [tempdb]) are always ignored
            3. @debug: use this to have the script also include intermediate datasets in addition to the T-SQL script output
         2. Review the output produced to ensure the correctness of the following:
            1. Databases to be altered
            2. Mode into which databases are to be placed
         3. This script can be prepared ahead of time, as it doesn’t require any special conditions or timing
      3. On the source server, execute the new script produced in (ii)
         1. The databases should now not be accessible to their normal users on the source server
         2. Save this script, its output/results and the results of (ii) as artifacts
      4. Execute the steps in the sections ‘Back Up User Databases’ and ‘Restore User Databases’ with the following exceptions
         1. When producing the database back up script, set @tailLog = 1; this will produce tail log backups of each of the desired databases
         2. When producing the database restore script, set @withRecovery = 1; this will leave each of the new databases in a recovered state; the destination databases should still only be available to restricted-use users, since the tail log backups were taken after that went into effect
         3. The databases should now be fully copied from the source server to the destination
      5. Perform any post-copy adjustments (e.g. database compatibility level) on the destination server and databases
      6. On the destination server, execute the T-SQL script ‘**get scripts to alter database user mode to restricted-user and multi-user.sql**’
         1. This time, set @mode = N’multi\_user’
         2. Review the output produced to ensure the correctness of the following:
            1. Databases to be altered
            2. Mode into which databases are to be placed
         3. This script can be prepared ahead of time, as it doesn’t require any special conditions or timing
      7. On the destination server, execute the new script produced in (vi)
         1. The databases should now be accessible to all appropriate users and applications
         2. Save this script, its output/results and the results of (vi) as artifacts
      8. Adjust any connection strings, configurations or aliases as needed