



MySQL Online Backup

Version 1.0

(7/28/2006)

by

Robin Schumacher

Table of Contents

1. INTRODUCTION..... 3

2. FUNCTIONAL DESCRIPTION 3

 2.1 BACKUP FUNCTION 4

 2.1.1 Core Feature Set..... 4

 2.1.2 Restrictions in Backup Function..... 4

 2.2 RESTORE FUNCTION 5

 2.2.1 Core Feature Set..... 5

 2.2.2 Restrictions in Restore Function..... 5

 2.3 BACKUP/RESTORE SYNTAX 6

 2.3.1 Backup Syntax..... 6

 2.3.2 Restore Syntax 6

 2.4 BACKUP/RESTORE SECURITY PRIVILEGES 7

 2.5 INFORMATION SCHEMA ENHANCEMENTS..... 7

 2.6 MISCELLANEOUS NOTES REGARDING BACKUP AND RESTORE FUNCTIONS 7

 2.7 BACKUP/RESTORE EXAMPLES..... 8

 2.7.1 Backup Examples..... 8

 2.7.2 Restore Examples 9

 2.8 BACKUP/RESTORE FAQ..... 11

 2.9 OPEN ISSUES 12

3. HISTORY 12

1. Introduction

This document outlines the requirements for an online backup function (and accompanying restore function) for the MySQL database server. The document's intent is to introduce this new functionality to the marketing and engineering staff of MySQL AB, as well as selected customers and community members to obtain feedback.

The basic features and functions of the proposed online backup will be outlined below.

2. Functional Description

Currently, data contained within a MySQL server can be backed up in a variety of ways:

- **Cold Backup** – the MySQL server is stopped, the underlying data and configuration files are copied to another location, and the MySQL server is restarted.
- **MySQL Hot Copy** – the MySQL server continues to operate while the underlying data and configuration files are copied to another location. Note that the “hot” aspect of this backup is that the MySQL server is not stopped, however blocking does occur with respect to client operations while the datafiles are copied.
- **mysqldump** – the mysqldump utility creates a backup file containing SQL statements that are used to recreate one or more MySQL databases. Note the blocking does occur for objects that are the target of the backup operation.
- **SELECT INTO OUTFILE** – for single table backups, a SELECT statement can be run that writes a text file containing the data from a MySQL table.
- **Replication** – used for a standby or backup server scenario, replication can be configured to maintain an exact copy of one or more MySQL databases on a different MySQL instance (same or different machine).
- **Innodb Hot Backup** – used for the InnoDB storage engine, and purchased separately from Innobase/Oracle, the Innobase hot backup allows a true online backup (no disruption of user activity) of InnoDB tables to occur.

While each solution above offers various benefits, no one backup operation provides all these features:

- **Cross-engine backup** – while the Innobase hot backup does a real online backup, it does not backup all MySQL storage engines.
- **Non-blocking backup for DML operations** – no solution does a backup that supports not blocking DML operations for all storage engines.
- **Bundled free with the server** – the Innobase solution is a cost-based product.
- **DDL Command driven** – most solutions require either operating system commands to be issued or configuration changes to occur (replication).

The goal of the proposed online backup function is to provide a bullet-proof backup and recovery paradigm that provides coverage for all MySQL storage engines, does not block any DML activity so that the vast majority of database work can continue uninterrupted, is offered free as part of the MySQL server, and is very straightforward and easy to use.

2.1 Backup Function

This section contains a global overview of the new MySQL Backup function.

2.1.1 Core Feature Set

The first version of the Backup function will provide the following features and benefits:

- The Backup function will support all existing MySQL storage engines. As new internal storage engines are introduced, a requirement for their release will be that they support the Backup function. Third party vendors should also conform and support the MySQL Backup function. As of this writing, the following internal engines will be covered by the Backup function:
 - MyISAM
 - InnoDB
 - Falcon
 - Memory (only the structure definition will be backed up)
 - Archive
 - CSV
 - NDB
 - Merge (only the structure definition will be backed up)
 - Federated (only the structure definition will be backed up)
 - Blackhole (only the structure definition will be backed up)
- The Backup function will offer online capabilities in that DML operations are not blocked during the backup operation.
- The Backup function will be DDL-like command driven from any interface that connects into MySQL (mysql client program, GUI clients, etc.)
- The Backup function will support the writing to local disk drives, network disk drives/devices, and tape drives.
- The Backup function will be able to back up an entire MySQL instance into a single backup file. Selected schemas/databases or only a single schema/database may also be backed up into a single file.
- The consistency point of a backup – the time it is deemed consistent and proper – will be communicated to the person running the backup in two ways:
 - When a DDL Backup function ends, it will write back to the client a message stating the date/timestamp of the backup file's consistency point.
 - The above mentioned message will also be written to the MySQL error log.
- Consistency implies that all referential integrity between a set of tables (within a particular storage engine) is kept intact within a backup. MySQL's Backup function will meet this requirement.
- Non-transactional storage engines will only contain committed data from the point the backup command started.

2.1.2 Restrictions in Backup Function

For version one of the MySQL Backup function, the following restrictions will exist. Each will be addressed in future versions of the MySQL server roadmap.

- All DDL operations will be blocked during the backup operation. Note that DDL blockage is instance-wide in the MySQL server and includes DDL statements issued against schemas/databases that are not targets of a backup operation.
- Multiple backup locations will not be supported (the simultaneous writing of two or more backup streams for the same backup).
- Current engines that are in development at the time of this writing will not be supported in the first release of the Backup function.
- Support for the SolidDB engine is questionable at the time of this writing.
- A backup operation that fails to finish is not re-startable – the operation must be reissued.
- Differential or incremental backups will not be supported in version one.
- Table level backups will not be supported in version one.

2.2 Restore Function

This section contains a global overview of the new MySQL Restore function.

2.2.1 Core Feature Set

The first version of the Restore function will provide the following features and benefits:

- The Restore function will support the reading from local disk drives, network disk drives, and tape drives.
- The Restore function will support the restoration of an entire MySQL instance or selected schemas/databases from a backup file that contains either an entire MySQL instance or selected schemas/databases.
- Point-in-time recovery will be possible when a restore operation is performed along with a restore of the MySQL binary logs.

2.2.2 Restrictions in Restore Function

For version one of the MySQL Restore function, the following restrictions will exist. Each will be addressed in future versions of the MySQL server roadmap.

- A restore operation blocks all activity against the schema(s) being restored. Note that all activity can continue as normal against schemas/databases not being restored.
- A restore operation blocks all DDL activity on the entire instance until the restore operation is completed.
- A restore operation that fails to finish is not re-startable – the operation must be reissued.
- A restore operation will not overwrite existing MySQL files. This means that certain files will need to be removed prior to performing a restore operation. These files include:
 - All .FRM files
 - All non-InnoDB, Falcon, and NDB datafiles
 - All index files

Note that the InnoDB and NDB tablespace files, InnoDB and Falcon log files, and Falcon datafile(s) do **not** need to be removed to perform a restore. The files for the 'mysql' database should also **not** be removed.

- Table level restores will not be supported in version one.

2.3 Backup/Restore Syntax

This section covers the proposed Backup and Restore function syntax.

2.3.1 Backup Syntax

The following syntax is proposed for the Backup function:

```
BACKUP { [schema name] | [schema1, schema2, ...] | [ALL_SCHEMAS] }  
TO { DISK | TAPE } = { 'physical device name' }  
[OVERWRITE]  
[PASSWORD = { 'password' } ]  
[MEDIANAME = { 'name' } ]
```

Notes concerning the above syntax include:

- At least one schema name or ALL_SCHEMAS need to be supplied to the Backup function.
- DISK is the default media for the Backup function – it may be omitted in the command.
- OVERWRITE will overwrite any existing file specified in the TO clause. The default is to not overwrite any existing file.
- PASSWORD is used to assign a password to the file for security purposes.
- MEDIANAME is used to supply a name for the backup file when it is written to tape.

2.3.2 Restore Syntax

The following syntax is proposed for the Restore function:

```
RESTORE { [ALL_SCHEMAS] | [schema name] | [schema1, schema2, ...] }  
FROM { DISK | TAPE } = { 'physical device name' }  
[PASSWORD = { 'password' } ]  
[MEDIANAME = { 'name' } ]
```

Notes concerning the above syntax include:

- ALL_SCHEMAS is the default option for RESTORE
- DISK is the default media for the Restore function – it may be omitted in the command.

- PASSWORD is used to verify an optional password given to a file created by the BACKUP function.
- MEDIANAME is used to supply a name for the backup file when it has been written to tape.

2.4 Backup/Restore Security Privileges

The ability to run the Backup and Restore functions will require new special privileges. There will be a couple of new privileges to choose from:

- BACKUP_ADMIN – can backup and restore all schemas/databases on a MySQL instance. This privilege is available with the WITH GRANT option.
- BACKUP – granted at the per schema/database level, it permits a user to backup the schema/database for which they have been given access. This privilege is available with the WITH GRANT option.
- RESTORE – granted at the per schema/database level, it permits a user to restore the schema/database for which they have been given access. This privilege is available with the WITH GRANT option.

Note that the following privileges are implicitly granted in the following circumstances:

- The ‘root’ user will have the BACKUP_ADMIN privilege WITH GRANT option.
- Any user granted “ALL on *.*” will have the BACKUP_ADMIN privilege WITH GRANT option.
- Any user granted ALL on a particular schema will have the BACKUP and RESTORE privileges.

2.5 Information Schema Enhancements

At this time, the only structural change that needs to occur in the INFORMATION_SCHEMA is that the ENGINES table should be enhanced to include a new column that designates whether an engine supports the native MySQL Backup function:

```
mysql> desc engines;
```

Field	Type	Null	Key	Default	Extra
ENGINE	varchar(64)	NO			
SUPPORT	varchar(8)	NO			
COMMENT	varchar(80)	NO			
TRANSACTIONS	varchar(3)	NO			
XA	varchar(3)	NO			
SAVEPOINTS	varchar(3)	NO			
SUPP_BACKUP	char(1)	NO			

2.6 Miscellaneous Notes regarding Backup and Restore Functions

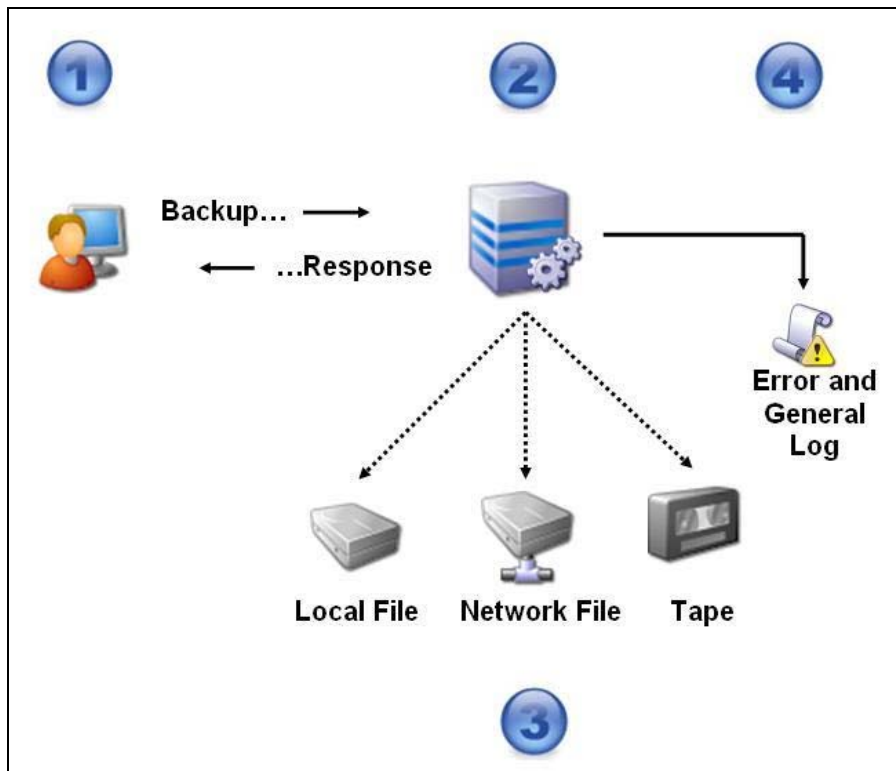
A few final notes regarding the Backup and Restore functions include the following:

- The Backup and Restore commands will be captured by the general query log if it is enabled. They should not be captured by the slow query log.
- The Backup and Restore commands should be supported syntax within the MySQL Event scheduler.

2.7 Backup/Restore Examples

This section outlines several examples for both backup and restore scenarios. Regardless of the type of backup desired, any backup operation consists of the following:

1. DBA issues request for backup to occur in either ad-hoc fashion or via a scheduled task on some machine.
2. The MySQL Server processes the backup request on the server that contains the schemas to be backed up
3. The MySQL Server writes the backup file to either a locally attached disk device, a network disk device, or a tape device.
4. Once completed, the MySQL Server sends a response back to the issuing program and writes a message to the MySQL error and (optionally) the backup command to the MySQL general log.



2.7.1 Backup Examples

The most simple backup operation involves the backup of a single MySQL schema to a resident physical disk drive:

```
BACKUP robin_schema TO '/usr/local/mysqlbackups/robinbackup.dmp'
```

The following would be used to backup up multiple schemas:

```
BACKUP robin_schema1, robin_schema2 TO  
'/usr/local/mysqlbackups/robinbackup.dmp'
```

The following would be used to backup up all schemas:

```
BACKUP all_schemas TO '/usr/local/mysqlbackups/robinbackup.dmp'
```

The following would be used to backup up all schemas with a password:

```
BACKUP all_schemas TO '/usr/local/mysqlbackups/robinbackup.dmp'  
PASSWORD='mypassword'
```

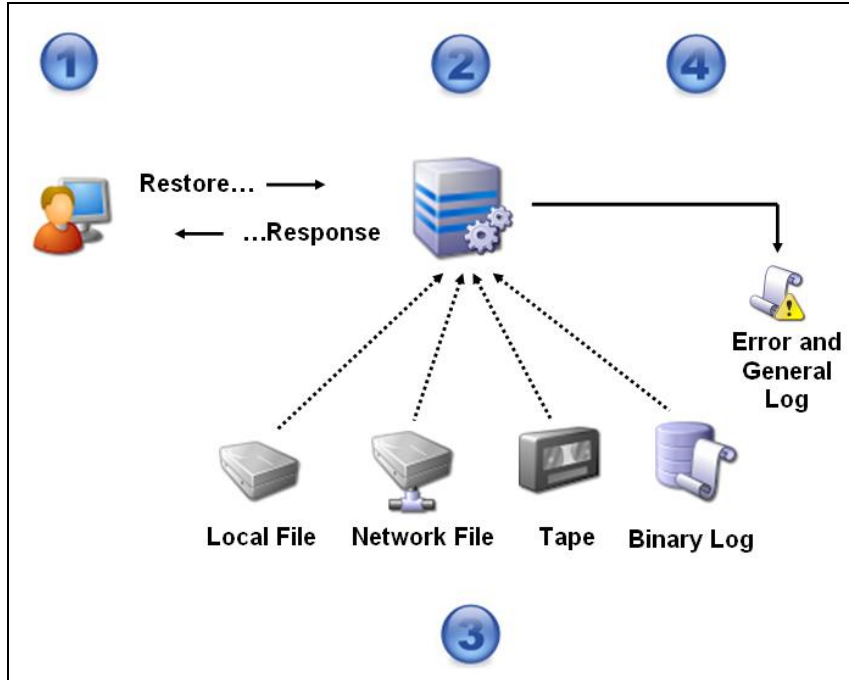
The following would be used to backup up all schemas with a password to a tape device:

```
BACKUP all_schemas TO tape='\\.\tape0' PASSWORD='mypassword'  
MEDIANAME='robinbak'
```

2.7.2 Restore Examples

Regardless of the type of restore desired, any restore operation consists of the following:

1. After preparing target server, DBA issues request for restore to occur in either ad-hoc fashion or via a scheduled task on some machine.
2. The MySQL Server processes the restore request on the server that contains the schemas to be restored
3. The MySQL Server reads the file(s) used for the restore from either a locally attached disk device, a network disk device, or a tape device.
4. Once completed, the MySQL Server sends a response back to the issuing program and writes a message to the MySQL error and (optionally) the restore command to the MySQL general log.



The most simple example is recovering all schemas written to a backup file:

```
RESTORE all_schemas FROM
`/usr/local/mysqlbackups/robinbackup.dmp`
```

The following would be used to restore a single schema:

```
RESTORE schema1 FROM `/usr/local/mysqlbackups/robinbackup.dmp`
```

The following would be used to restore selected schemas from a single backup file containing an entire MySQL instance:

```
RESTORE schema1, schema2 FROM
`/usr/local/mysqlbackups/robinbackup.dmp`
```

A point in time restore requires the use of a backup file along with binary log files that are needed to roll forward to a point in time. For example, to do a point-in-time recovery for an entire MySQL instance, you would first restore all schemas from the last backup file that is before the time a disaster or corruption occurred :

```
RESTORE all_schemas FROM
`/usr/local/mysqlbackups/robinbackup.dmp`
```

And then use the binary log files to roll forward. For example, the following would be entered on a Linux/Unix machine to roll forward up until 9:59:59 am on April 20, 2005 using a MySQL binary log named 'bin.123456':

```
mysqlbinlog --stop-date="2005-04-20 9:59:59" /var/log/mysql/bin.123456 |
mysql -u root -pmypwd --socket=/tmp/mysql_restore.sock
```

2.8 Backup/Restore FAQ

The following are some common questions and answers regarding the new Backup and Restore function that are not explicitly covered in the previous sections of this document:

Question: What will be the experience of a user who submits a DDL operation during a backup operation?

Answer: The user will receive an error back from the MySQL server stating that a backup operation is running and that they should retry their request after it has finished.

Question: What will be the experience of a user who submits a backup operation when a long running DDL command had previously been submitted?

Answer: The user will receive an error back from the MySQL server stating that a DDL command is currently running and that they should retry their backup request after it has finished.

Question: What will be the performance impact of a backup operation?

Answer: A degradation will occur, however the actual measurable impact is not known at this time. Once the backup operation enters the alpha stage, benchmarks will be used to gauge the performance impact.

Question: Will pre-defined backup locations be required?

Answer: No, a DBA can use any accessible location in an ad-hoc fashion in the Backup command.

Question: Will the Backup command create directories/filesystems that do not exist at the time the command is issued?

Answer: No, all backup locations must exist prior to the Backup command being issued or the operation will fail.

Question: What happens if the DBA uses a name for a backup file that already exists on the disk/tape drive?

Answer: They will receive an error unless they specifically state that the file should be overwritten in the Backup command. The default is to not overwrite an existing file.

Question: Will any log or configuration files be included in a backup operation?

Answer: No, only data files and structural information (server and object metadata) will be included in a backup operation. Any desire to save other files – such as binary logs – will require that they be backed up separately.

Question: Does the binary log need to be enabled to support backup/restore operations?

Answer: No, full backups and restores can be done without enabling the binary log. If point-in-time recovery is desired, then binary logging will need to be enabled.

Question: Other than the previously covered DDL syntax, are any other API's or functions being made available to third-party backup vendors?

Answer: Not at this time.

Question: Will backup file encryption or compression be included in the Backup function?

Answer: Not in version one.

Question: What impact will a restore operation have on a master server used for replication?

Answer: If the restored schemas are being replicated, then the slave server(s) will need to have the corresponding database files removed before the restore is performed on the master.

2.9 Open Issues

Below are minor issues yet to be decided.

- Can a predicted completion message – either in percent or some other form – be communicated to the DBA for both a backup and restore operation so the DBA has some understanding of the length of time and progress made thus far of either type of operation?
- Should a checksum validation option be granted so users can verify a backup file is good?

3. History

Author	Date	What done
Robin Schumacher	July 14, 2006	Initial spec written