MySQL Enterprise Backup User's Guide (Version 3.5.4)

MySQL Enterprise Backup User's Guide (Version 3.5.4)

Abstract

This is the User's Guide for the MySQL Enterprise Backup product, the successor to the InnoDB Hot Backup product.

Document generated on: 2014-10-13 (revision: 5163)

Table of Contents

Preface and Legal Notices

This is the User Manual for the MySQL Enterprise Backup product.

For license information, see the Legal Notices. This product may contain third-party code. For license information on third-party code, see Appendix C, *Licenses for Third-Party Components*.

Legal Notices

below. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

For more information on the terms of this license, or for details on how the MySQL documentation is built and produced, please visit MySQL Contact & Questions.

For additional licensing information, including licenses for third-party libraries used by MySQL products, see Appendix C, *Licenses for Third-Party Components*.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists where you can discuss your issues with other MySQL users.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Chapter 1 Introduction to MySQL Enterprise Backup

Table of Contents

The MySQL Enterprise Backup product performs hot backup operations for MySQL databases. The product is architected for efficient and reliable backups of tables created by the InnoDB storage engine. For completeness, it can also back up tables from MyISAM and other storage engines.

1.1 Hot Backups

Hot backups are performed while the database is running. This type of backup does not block normal database operations, and it captures even changes that occur while the backup is happening. For these reasons, hot backups are desirable when your database “grows up” -- when the data is large enough that the backup takes significant time, and when your data is important enough to your business so that you must capture every last change, without taking your application, web site, or web service offline.

MySQL Enterprise Backup does a hot backup of all tables that use the InnoDB storage engine. For tables using MyISAM or other non-InnoDB storage engines, it does a “warm” backup, where the database continues to run, but those tables cannot be modified while being backed up. For efficient backup operations, you can designate InnoDB as the default storage engine for new tables, or convert existing tables to use the InnoDB storage engine.

1.2 Backup Commands

When using the MySQL Enterprise Backup product, you primarily work with the `ibbackup` command and the `mysqlbackup` command, depending on the kinds of tables you are backing up. `ibbackup` is the simple and efficient way to back up InnoDB tables only, with many command-line options so that you can build your own backup script around it. `mysqlbackup` is the more flexible way to back up both InnoDB tables and tables from other storage engines at the same time; it also does other things that you would otherwise code into your own script, such as creating a timestamped subdirectory for each backup.

1.3 Performance and Space Considerations

In planning your backup strategy, you choose the balance between the overhead of performing the backup (CPU cycles, storage space, and network traffic), and the time needed to restore the data during planned maintenance or when disaster strikes. Using features such as incremental backups and compressed backups saves on storage space, and network traffic if you keep the backup data on a different server. Compression adds some CPU overhead to the backup process; incremental backup requires additional processing to make the backup ready to restore. For disaster recovery, when speed to restore the data is critical, you might prefer to have the backup data already prepared and uncompressed, so that the restore operation involves as few steps as possible.

When evaluating the performance of different backup techniques, put more emphasis on the speed of the restore operation than the speed of the initial backup. It is during a disaster recovery that speed is most critical. For example, although a logical backup performed with the `mysqldump` command might take about the same time as a physical backup with the MySQL Enterprise Backup product (at least for a small

database), the MySQL Enterprise Backup restore operation is typically faster. Copying the actual data files back to the data directory skips the overhead of inserting rows and updating indexes that comes from replaying the SQL statements from `mysqldump` output.

To minimize any impact on server performance on Linux and Unix systems, MySQL Enterprise Backup writes the backup data without storing it in the operating system's disk cache, by using the `posix_fadvise()` system call. This technique minimizes any slowdown following the backup operation, by preserving the data blocks in the disk cache rather than filling up the cache with the output from the backup.

1.4 Files that Are Backed Up

The primary InnoDB-related data files that are backed up include the ibdata* files that represent the system tablespace and possibly the data for some user tables; any .ibd files, containing data from user tables created with the file-per-table setting enabled; data extracted from the ib_logfile* files (the redo log information representing changes that occur while the backup is running), which is stored in a new backup file ibbackup_logfile.

If you use the compressed backup feature, the `.ibd` files are renamed in their compressed form to .ibz files.

The files, as they are originally copied, form a raw backup that requires further processing to be ready to be restored. You then run the apply step, which updates the backup files based on the changes recorded in the `ibbackup_logfile` file, producing a prepared backup. At this point, the backup data corresponds to a single point in time. The files are now ready to be restored to their original location, or for some other use, such as testing or deployment as a replication slave.

To restore InnoDB tables to their original state, you must also have the corresponding .frm files along with the backup data. Otherwise, the table definitions could be missing or outdated, if someone has run `ALTER TABLE` or `DROP TABLE` statements since the backup. The `mysqlbackup` command automatically copies the `.frm` files back and forth during backup and restore operations. If you rely on the lower-level `ibbackup` command for the backup, you must copy the `.frm` files yourself.

The `mysqlbackup` command can also back up the .MYD files, .MYI files, and associated `.FRM` files for MyISAM tables. The same applies to files with other extensions, as shown in this list [27]. MyISAM tables and these other types of files cannot be backed up in the same non-blocking way as InnoDB tables can; changes to these tables are prevented while they are being backed up, possibly making the database unresponsive for a time.

1.5 Comparison of MySQL Enterprise Backup and InnoDB Hot Backup

In terms of features, the MySQL Enterprise Backup product is a superset of the InnoDB Hot Backup product that it supersedes:

- The `ibbackup` command is available on all platforms, with the same options as before plus some new ones. In particular, the `--incremental` option enables incremental backups of InnoDB tables.
- The `mysqlbackup` command, a cross-platform replacement for the `innobackup` command, is now available on Windows. Windows users can back up MyISAM tables and tables from other storage engines besides InnoDB, without writing their own wrapper script for the `ibbackup` command.
- The `mysqlbackup` command is a C program connecting to the server through the MySQL API, rather than a Perl script that runs the `mysql` command. Because it does not run the actual `mysql` command,

it does not support the `--mysql-extra-args` option of `innobackup`, but otherwise the syntax is compatible.

This documentation refers to the `mysqlbackup` command exclusively.

If this implementation change presents any issues for former users of the InnoDB Hot Backup product (for example, if you customized the `innobackup` script or relied on specific `mysqld` options passed through the `--mysql-extra-args` option), please submit requirements against the new `mysqlbackup` command.

- Currently, the old `innobackup` Perl script is still supplied on Linux and Unix systems, for troubleshooting in case of upgrade issues as you transition to the `mysqlbackup` command.
- Backups produced by the InnoDB Hot Backup product can be restored by the MySQL Enterprise Backup product.
- The incremental backup feature is new to MySQL Enterprise Backup.
- Support for the Barracuda file format is new to MySQL Enterprise Backup.
- The MySQL Enterprise Backup product includes some new performance optimizations, such as the `posix_fadvise()` system call.
- A new logging capability records the progress of running backup jobs, and historical details for completed backup jobs. For details, see Section 9.3, “Using the MySQL Enterprise Backup Backup Logs”.
- The `mysqlbackup` command has extra flexibility for specifying the MySQL connection information. It can read the user, password, port and socket options from the `[client]` group of your default or user-specified configuration file. If you supply the `--password` option without an argument, you are prompted to enter the password interactively.

Chapter 2 Installing MySQL Enterprise Backup

The MySQL Enterprise Backup product is packaged as either an archive file (`.tgz`, archived with `tar` and compressed with `gzip`), or as a platform-specific installer that is more automated and convenient than with the former InnoDB Hot Backup product.

Installing on Unix and Linux Systems

For all Linux and Unix systems, the product is available as a `.tgz` file. Unpack this file as follows:

```
tar xvzf package.tgz
```

The `ibbackup` and `mysqlbackup` commands are unpacked into a subdirectory. You can either copy them into a system directory (preserving their execute permission bits), or add to your `$PATH` setting the directory where you unpacked them.

For certain Linux distributions, the product is also available as an RPM archive. When you install the RPM, using the command `sudo rpm -i package_name.rpm`, the `ibbackup` and `mysqlbackup` commands are installed in the directory `/opt/mysql/meb-3.5`. You must add this directory to your `$PATH` setting.

Installing on Windows Systems

Specify the installation location, preferably relative to the directory where the MySQL Server product is installed. (You can also install the MySQL Enterprise Backup product on a computer that is used for storing backup data, and does not have the MySQL Server product.)

Choose the option to add this directory to the windows `%PATH%` setting, so that you can run the `ibbackup` command from a command prompt.

Verify the installation by selecting the menu item Start > Programs > MySQL Enterprise Backup 3.5 > MySQL Enterprise Backup Command Line. This menu item opens a command prompt and runs the `ibbackup` command to display its help message showing the option syntax.

Chapter 3 ibbackup Command Reference

Table of Contents

The ibbackup of MySQL Enterprise Backup backs up InnoDB tables from a running MySQL database without disturbing normal database processing. You get a consistent copy of your database, as if the copy were taken at a precise point in time, even though the backup job could take minutes or hours. MySQL Enterprise Backup is also the ideal method of setting up new slaves if you use MySQL replication on InnoDB tables.

The basic command to take a backup is ibbackup my.cnf my2.cnf. This command reads from the my.cnf file the information where the ibdata and ib_logfiles are, and makes a backup of them to the locations specified in my2.cnf file.

3.1 Options Files

You specify two filenames on the ibbackup command line. These .cnf files represent MySQL options files and use similar syntax. These options file have the following requirements:

- Each options file must contain the following **parameter values**:

```
datadir=...
innodb_data_home_dir=...
innodb_data_file_path=...
innodb_log_group_home_dir=...
innodb_log_files_in_group=...
innodb_log_file_size=...
```

- The **directory paths** must be absolute. Because backup is such a critical procedure, to avoid the possibility of backing up the wrong files, ibbackup does not assume any defaults for file locations.
- The number of **data files** and their sizes must match in my.cnf and my2.cnf. For example, if the last data file is specified as auto-extending in my.cnf, it must be specified as auto-extending also in my2.cnf.
- The number of **log files** and their size must be explicitly specified, but their number and size can be different between my.cnf and my2.cnf.
- Any other options in these files are ignored. You can prepare new, minimal files with just these options, or use existing configuration files that have the appropriate options.

For example, suppose your my.cnf contains the following:

```
[mysqld]
datadir = /home/heikki/data
innodb_data_home_dir = /home/heikki/data
innodb_data_file_path = ibdata1:10M:autoextend
innodb_log_group_home_dir = /home/heikki/data
innodb_log_files_in_group=2
innodb_log_file_size=20M
```

To back up your database to directory /home/heikki/backup, create my2.cnf like the following:

```
datadir = /home/heikki/backup
innodb_data_home_dir = /home/heikki/backup
innodb_data_file_path = ibdata1:10M:autoextend
innodb_log_group_home_dir = /home/heikki/backup
innodb_log_files_in_group=2
innodb_log_file_size=20M
```

Because `ibbackup` **does not overwrite any files** during the initial backup step, the backup directory must not contain any old backup files. `ibbackup` stops when asked to create a file that already exists, to avoid harming an existing backup.

3.2 ibbackup Command-Line Options

The command `ibbackup --help` displays usage information for the command-line options:

```
$ ibbackup --help
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Usage:
ibbackup [--incremental lsn]
         [--sleep ms] [--suspend-at-end] [--compress [level]]
         [--include regexp] my.cnf backup-my.cnf
or
ibbackup --apply-log
         [--use-memory mb] [--uncompress]
         backup-my.cnf
or
ibbackup --apply-log --incremental
         [--use-memory mb] [--uncompress]
         incremental-backup-my.cnf full-backup-my.cnf

The first command line call above reads the data file and log file
information from my.cnf, and stores the backup data files and a backup
log file (named 'ibbackup_logfile') in the directories specified in the
backup-my.cnf file.

If --incremental is specified in the first command above, it instructs
ibbackup to make an INCREMENTAL backup that only contains data pages
whose latest modification has a log sequence number (lsn) greater than
lsn. Make sure that you have earlier taken a full
backup!

The .cnf files must contain explicit values of (ibbackup is not aware of defaults):
datadir=...
innodb_data_home_dir=...
innodb_data_file_path=...
innodb_log_group_home_dir=...
set-variable = innodb_log_files_in_group=...
set-variable = innodb_log_file_size=...

If --apply-log is specified, then the program prepares a backup for
starting mysql server on the backup. It applies the log records in
'ibbackup_logfile' to the data files, and creates new log files as
specified in backup-my.cnf. If you run with the --apply-log option,
then ibbackup does not check the hostname of the computer. You are allowed
to use ibbackup --apply-log in any computer without a need for a
MySQL Enterprise Backup license for that computer.

If --incremental is specified after --apply-log, then ibbackup
applies the incremental backup at incremental-backup-my.cnf to the FULL
backup at full-backup-my.cnf.

If --include regexp is specified, only those per-table data files which
```

```
match the given regular expression are included in the backup.
For each table with per-table data file a string of the form
db_name.table_name is checked against the regular expression.
If the regular expression matches the complete string db_name.table_name,
the table is included in the backup. The regular expression should be of
the POSIX 1003.2 "extended" form. Example: expression 'sales\.den.*'
matches all tables starting with "den" in database "sales".
Note that on Unix (not on Windows) the regular expression should be placed
in single quotes to prevent interpretation by the shell. This feature
is implemented with Henry Spencer's regular expression library.

--restore is an obsolete synonym for --apply-log. The use of --restore
is deprecated, because it may be dropped in the future.

--sleep ms instructs the program to sleep ms milliseconds after each 1 MB
of copied data. You can use this parameter to tune the additional disk i/o
load the backup program causes on the computer. ms must be < 1000000.
The default for ms is 0.

--suspend-at-end makes ibbackup to behave like this: when the backup
procedure is close to ending, ibbackup creates a file called
'ibbackup_suspended' to the log group home dir specified in backup-my.cnf
and waits until the user deletes that file 'ibbackup_suspended'.
You can use this option if you want to write a script which locks
and backs up your MyISAM tables at the end of ibbackup. In that way you
get a consistent snapshot of both InnoDB and MyISAM tables.

--use-memory mb is relevant only when --apply-log is specified.
It tells ibbackup that it can use mb megabytes of memory in recovery.
The default is 100 MB.

--compress instructs the program to compress the backup copies of data
files. Compressed data files are named by adding suffix '.ibz' to the
file name. Compression level can be specified as an optional argument
following --compress option. Compression level is an integer
between 0 and 9: 1 gives fastest compression, 9 gives best compression,
and 0 means no compression. If compression level is not given,
the default level 1 (fastest compression) is used.
Note that data files of per-table tablespaces for compressed tables do not
benefit from a second level of compression, so they are never compressed
in a backup.

--uncompress is relevant only when --apply-log is specified.
It tells ibbackup to recover data files from compressed copies.
Compressed data files are named with suffix '.ibz'.

The backup program does NOT make a backup of the .frm files of the tables,
and it does not make backups of MyISAM tables. To back up these items, either:

- Use the mysqlbackup program.

- Make backups of the .frm files with the Unix 'tar' or the Windows WinZip
or an equivalent tool both BEFORE and AFTER ibbackup finishes its work,
and also store the MySQL binlog segment that is generated between the moment
you copy the .frm files to a backup and the moment ibbackup finishes its work.
For extra safety, also use:

mysqldump -l -d yourdatabasename

to dump the table CREATE statements in a human-readable form before
ibbackup finishes its work.

From the binlog segment you see if any of the .frm files changed between the
moment you took a .frm files backup and the moment ibbackup finished
its work.

Please send bug reports to
```

```
http://bugs.mysql.com.
```

Chapter 4 Making a Backup

Table of Contents

`ibbackup` makes a backup of:

- The InnoDB system tablespace, which by default contains all the InnoDB tables.
- Any separate data files produced under the InnoDB file-per-table setting.
- Indexes associated with InnoDB tables.
- Data stored in both the original Antelope and new Barracuda file formats. Barracuda is a relatively recent file format that was first introduced with the InnoDB plugin. In particular, it includes support for compression within table data. Barracuda file format support is new in MySQL Enterprise Backup Version 3.5. It is automatic, without any option required for `ibbackup`.

`ibbackup` does *not*, however, copy `.frm` files, MyISAM tables, or MyISAM indexes to the backup. See Section 4.4, “Making a Point-in-Time Backup of InnoDB, MyISAM, and Other Tables” for details on how to make a complete backup.

4.1 Example: Making an Uncompressed Backup of InnoDB Tables

In this example, the options file `/home/pekka/.backup-my.cnf` defines the location of the backup (as described in the previous section). The options file `/home/pekka/.my.cnf` defines the MySQL installation to back up. Running `ibbackup` performs the first phase of the process:

```
$ ibbackup /home/pekka/.my.cnf /home/pekka/.backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.my.cnf:
innodb_data_home_dir got value /sqldata/simple
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata/simple
innodb_log_group_home_dir got value /sqldata/simple
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /home/pekka/.backup-my.cnf:
innodb_data_home_dir got value /sqldata-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata-backup
innodb_log_group_home_dir got value /sqldata-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

ibbackup: System tablespace file format is Antelope.
ibbackup: Found checkpoint at lsn 32164666892.
ibbackup: Starting log scan from lsn 32164666880.
```

```
101208 15:32:32  ibbackup: Copying log...
101208 15:32:32  ibbackup: Log copied, lsn 32164666892.
ibbackup: We wait 1 second before starting copying the data files...
101208 15:32:33  ibbackup: Copying /sqldata/simple/ibdata1 (Antelope file format).
101208 15:32:34  ibbackup: Copying /sqldata/simple/ibdata2 (Antelope file format).
101208 15:32:36  ibbackup: Copying /sqldata/simple/ibdata3 (Antelope file format).
ibbackup: Progress in MB: 200 400
ibbackup: A copied database page was modified at 32164665879.
ibbackup: Scanned log up to lsn 32164666892.
ibbackup: Was able to parse the log up to lsn 32164666892.
ibbackup: Maximum page number for a log record 0
101208 15:33:11  ibbackup: Full backup completed!
```

The backup directory now contains a backup log file and copies of InnoDB data files:

```
$ ls -l /sqldata-backup
total 499728
-rw-r--r-- 1 pekka pekka       158 2010-12-08 15:33 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka      1024 2010-12-08 15:33 ibbackup_logfile
-rw-r----- 1 pekka pekka  10485760 2010-12-08 15:32 ibdata1
-rw-r----- 1 pekka pekka  20971520 2010-12-08 15:32 ibdata2
-rw-r----- 1 pekka pekka 480247808 2010-12-08 15:33 ibdata3
```

Next Steps:

- Make a note of the LSN value in the message at the end of both full and incremental backups, for example, `ibbackup: Was able to parse the log up to lsn LSN_number`. You specify this value when performing incremental backups of changes that occur after this full backup.
- Apply the log to the uncompressed backup files, so that the full backup is ready to be restored at any time. You can move the backup data to a different server first, to avoid the CPU and I/O overhead of performing this operation on the database server.
- After applying the log, periodically take incremental backups, which are much faster and smaller than a full backup like this.

4.2 Example: Making a Compressed Backup of InnoDB Tables

To save disk space, you can compress backup data files by using the `--compress` option of `ibbackup`. Compression lets you keep more sets of backup data on hand, and saves on transmission time when sending the backup data to another server. The downside is extra CPU overhead during the backup itself, and extra time needed during the restore process as the data is uncompressed.

When tablespace files are compressed during backup, they receive the extension `.ibz` rather than the usual `.ibd` extension. To avoid wasting CPU cycles without saving additional disk space, `--compress` does not attempt to compress already-compressed tables that use the Barracuda file format; such tablespace files keep the usual `.ibd` extension.

You can use the `--compress` option for full backups, or for incremental backups in combination with the `--incremental` option.

```
$ ibbackup --compress /home/pekka/.my.cnf /home/pekka/.backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.my.cnf:
```

```
innodb_data_home_dir got value /sqldata/simple
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata/simple
innodb_log_group_home_dir got value /sqldata/simple
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /home/pekka/.backup-my.cnf:
innodb_data_home_dir got value /sqldata-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata-backup
innodb_log_group_home_dir got value /sqldata-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

ibbackup: System tablespace file format is Antelope.
ibbackup: Found checkpoint at lsn 32164666892.
ibbackup: Starting log scan from lsn 32164666880.
101208 15:47:39  ibbackup: Copying log...
101208 15:47:39  ibbackup: Log copied, lsn 32164666892.
ibbackup: We wait 1 second before starting copying the data files...
101208 15:47:40  ibbackup: Copying /sqldata/simple/ibdata1 (Antelope file format).
101208 15:47:41  ibbackup: Copying /sqldata/simple/ibdata2 (Antelope file format).
101208 15:47:42  ibbackup: Copying /sqldata/simple/ibdata3 (Antelope file format).
ibbackup: Progress in MB: 200 400
ibbackup: A copied database page was modified at 32164665879.
ibbackup: Scanned log up to lsn 32164666892.
ibbackup: Was able to parse the log up to lsn 32164666892.
ibbackup: Maximum page number for a log record 0

ibbackup: Compressed 488 MB of data files to 53 MB (compression 89%).

101208 15:48:09  ibbackup: Full backup completed!
```

The backup directory is shown below. Compressed data files have the suffix `.ibz`. Typically, compression ratios of more than 70% are achieved:

```
$ ls -l /sqldata-backup
total 54676
-rw-r--r-- 1 pekka pekka      158 2010-12-08 15:48 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka     1024 2010-12-08 15:48 ibbackup_logfile
-rw-r----- 1 pekka pekka  1095854 2010-12-08 15:47 ibdata1.ibz
-rw-r----- 1 pekka pekka   811625 2010-12-08 15:47 ibdata2.ibz
-rw-r----- 1 pekka pekka 54058462 2010-12-08 15:48 ibdata3.ibz
```

Next steps:

- Make a note of the LSN value in the message at the end of both full and incremental backups, for example, `ibbackup: Was able to parse the log up to lsn` *`LSN_number`*. You specify this value when performing incremental backups of changes that occur after this full backup.
- Apply the log to the compressed backup files, so that the full backup is ready to be restored at any time. You can move the backup data to a different server first, to avoid the CPU and I/O overhead of performing this operation on the database server.
- After applying the log, periodically take incremental backups, which are much faster and smaller than a full backup like this.

4.3 Example: Making an Incremental Backup of InnoDB Tables

An incremental backup only backs up data that changed since the previous backup. This technique provides additional flexibility in designing a backup strategy and reduces required storage for backups.

Because an incremental backup always adds to an existing set of backup files, make a full uncompressed or compressed backup before doing any incremental backups.

Incremental backups detect changes at the level of pages in the data file, as opposed to table rows; each page that has changed is backed up. Thus, the space and time savings are not exactly proportional to the percentage of changed rows or columns.

Incremental backup is enabled through the `--incremental` option of the `ibbackup` command. You must also indicate the point in time of the previous full or incremental backup, through the `--lsn` option, where you specify the highest log sequence number from a previous full or incremental backup.

When running the “apply log” step for an incremental backup, you specify the option sequence `--apply-log --incremental`, and the paths to 2 MySQL configuration files, first the `.cnf` file pointing to the full backup that you are updating, then the .cnf file pointing to the incremental backup data files. If you have taken several incremental backups since the last full backup, you might run several such “apply log” steps, one after the other, to bring the full backup entirely up to date.

If you use the `mysqlbackup` command to back up tables from MyISAM and other storage engines, you can perform an incremental backup with that command also, using slightly different command-line syntax.

This example shows an incremental backup. The last full backup we ran reported that the highest LSN was 2638548215:

```
ibbackup: Was able to parse the log up to lsn 2638548215
```

We specify that number again in the command here; the incremental backup includes all changes that came *after* the specified LSN.

```
$ ibbackup --incremental 2638548215 /home/pekka/.my.cnf /home/pekka/.incr-backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.my.cnf:
innodb_data_home_dir got value /sqldata/mts
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata/mts
innodb_log_group_home_dir got value /sqldata/mts
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /home/pekka/.incr-backup-my.cnf:
innodb_data_home_dir got value /incr-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /incr-backup
innodb_log_group_home_dir got value /incr-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

ibbackup: System tablespace file format is Barracuda.
ibbackup: Found checkpoint at lsn 2654252454.
ibbackup: Starting log scan from lsn 2654252032.
101208 17:12:16  ibbackup: Copying log...
101208 17:12:16  ibbackup: Log copied, lsn 2654252454.
ibbackup: We wait 1 second before starting copying the data files...
101208 17:12:17  ibbackup: Copying /sqldata/mts/ibdata1 (Barracuda file format).
101208 17:12:17  ibbackup: Copying /sqldata/mts/ibdata2 (Barracuda file format).
101208 17:12:18  ibbackup: Copying /sqldata/mts/ibdata3 (Barracuda file format).
```

```
101208 17:12:21  ibbackup: Copying /sqldata/mts/test/alex1.ibd (Antelope file format).
101208 17:12:22  ibbackup: Copying /sqldata/mts/test/alex2.ibd (Antelope file format).
101208 17:12:22  ibbackup: Copying /sqldata/mts/test/alex3.ibd (Antelope file format).
101208 17:12:22  ibbackup: Copying /sqldata/mts/test/blobt3.ibd (Antelope file format).
101208 17:12:23  ibbackup: Copying /sqldata/mts/test/ibstest0.ibd (Antelope file format).
101208 17:12:24  ibbackup: Copying /sqldata/mts/test/ibtest09.ibd (Antelope file format).
ibbackup: A copied database page was modified at 2654252454.
ibbackup: Scanned log up to lsn 2654252454.
ibbackup: Was able to parse the log up to lsn 2654252454.
ibbackup: Maximum page number for a log record 0
ibbackup: Backup contains changes from lsn 2638548216 to lsn 2654252454
101208 17:12:24  ibbackup: Incremental backup completed!
```

This is the initial state of the backup files from the full backup:

```
$ ls -hlR /full-backup/
/full-backup/:
total 105M
-rw-r--r-- 1 pekka pekka  155 2010-12-08 17:11 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka 1.0K 2010-12-08 17:11 ibbackup_logfile
-rw-r----- 1 pekka pekka  10M 2010-12-08 17:11 ibdata1
-rw-r----- 1 pekka pekka  20M 2010-12-08 17:11 ibdata2
-rw-r----- 1 pekka pekka  74M 2010-12-08 17:11 ibdata3
drwxr-x--- 2 pekka pekka   48 2010-12-08 17:11 test

/full-backup/test:
total 69M
-rw-r----- 1 pekka pekka  23M 2010-12-08 17:11 alex1.ibd
-rw-r----- 1 pekka pekka 240K 2010-12-08 17:11 alex2.ibd
-rw-r----- 1 pekka pekka 240K 2010-12-08 17:11 alex3.ibd
-rw-r----- 1 pekka pekka  19M 2010-12-08 17:11 blobt3.ibd
-rw-r----- 1 pekka pekka  25M 2010-12-08 17:11 ibstest0.ibd
-rw-r----- 1 pekka pekka 912K 2010-12-08 17:11 ibtest09.ibd
```

These are the (much smaller) files created by the incremental backup:

```
$ ls -hlR /incr-backup/
/incr-backup/:
total 2.7M
-rw-r--r-- 1 pekka pekka  155 2010-12-08 17:12 ibbackup_export_variables.txt
-rw-r--r-- 1 pekka pekka  118 2010-12-08 17:12 ibbackup_ibd_files
-rw-r----- 1 pekka pekka 1.0K 2010-12-08 17:12 ibbackup_logfile
-rw-r----- 1 pekka pekka 2.2M 2010-12-08 17:12 ibdata1
-rw-r----- 1 pekka pekka 481K 2010-12-08 17:12 ibdata2
drwxr-x--- 2 pekka pekka   24 2010-12-08 17:12 test

/incr-backup/test:
total 14M
-rw-r----- 1 pekka pekka 14M 2010-12-08 17:12 blobt3.ibd
```

Next steps:

- Make a note of the LSN value in the message at the end of the backup, for example, `ibbackup: Was able to parse the log up to lsn` *`LSN_number`*. You specify this value when performing incremental backups of changes that occur after this incremental backup.
- Apply the incremental backup to the backup files, so that the backup is ready to be restored at any time. You can move the backup data to a different server first, to avoid the CPU and I/O overhead of this operation on the database server itself.
- On a regular schedule, determined by date or amount of database activity, take further take incremental backups.

- Optionally, periodically start the cycle over again by taking a full uncompressed or compressed backup. Typically, this milestone happens when you can archive and clear out your oldest backup data.

4.4 Making a Point-in-Time Backup of InnoDB, MyISAM, and Other Tables

> **Note**
>
> This section describes low-level backup techniques that are intended for expert users. You can use the `mysqlbackup` command to automate the procedure presented in this section, as described in Chapter 7, *`mysqlbackup` Command Reference*.

You can use the `ibbackup` option `--suspend-at-end` to make a point-in-time backup of tables from MyISAM and other storage engines, in addition to InnoDB tables. This option pauses the backup process when an `ibbackup` run is close to ending, so that a script written by you can copy the MyISAM tables and other non-InnoDB files to a backup. Since the backup taken by `ibbackup` corresponds to the time at the end of the run, and your script blocks modifications to the MyISAM tables at that point, the backup contains a snapshot of MyISAM tables at the same point in time. Your script must follow this outline:

- Issue the command `ibbackup --suspend-at-end` ...;
- Periodically check for the existence of the file `ibbackup_suspended`. When this file appears, your script must:
  1. Issue the statement `FLUSH TABLES WITH READ LOCK;`
  2. Manually copy all `.frm` files, MyISAM tables, and other non-InnoDB files from database directories to a backup location.
  3. Resume the `ibbackup` run by deleting the file `ibbackup_suspended`.
  4. Wait until the `ibbackup` command finishes.
  5. Issue the statement `UNLOCK TABLES;`.

Chapter 5 Preparing the Backup to be Restored

Table of Contents

The initial backup files might not be in a consistent state, because data could be changed while the backup is running. These initial files are known as the raw backup. The next step is to update the backup files so that they reflect the state of the database corresponding to a specific InnoDB log sequence number. (The same kind of operation as crash recovery. When this step is complete, these final files are known as the prepared backup.

During the backup, `ibbackup` copies the accumulated InnoDB log to a file called `ibbackup_logfile`. This log file is used to “roll forward” the backed-up data files, so that every page in the data files corresponds to the same log sequence number of the InnoDB log.

The option for applying the log is `--apply-log`. In the prior InnoDB Hot Backup Product, this option was called `--restore`, but that option was renamed because it prepares the data to be restored, rather than actually restoring it.

This phase also creates new `ib_logfiles` that correspond to the data files.

5.1 Example: Applying the Log to a Backup

This example continues from Section 4.1, “Example: Making an Uncompressed Backup of InnoDB Tables”.

The backup directory looks like this before applying the log to the backup:

```
$ ls -l /sqldata-backup
total 499728
-rw-r--r-- 1 pekka pekka       158 2010-12-08 15:33 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka      1024 2010-12-08 15:33 ibbackup_logfile
-rw-r----- 1 pekka pekka  10485760 2010-12-08 15:32 ibdata1
-rw-r----- 1 pekka pekka  20971520 2010-12-08 15:32 ibdata2
-rw-r----- 1 pekka pekka 480247808 2010-12-08 15:33 ibdata3
```

We run `ibbackup` to roll forward the data files so that they correspond to the same log sequence number:

```
$ ibbackup --apply-log /home/pekka/.backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.backup-my.cnf:
innodb_data_home_dir got value /sqldata-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata-backup
innodb_log_group_home_dir got value /sqldata-backup
innodb_log_files_in_group got value 3
```

```
innodb_log_file_size got value 10485760

101208 15:47:17  ibbackup: ibbackup_logfile's creation parameters:
ibbackup: start lsn 32164666880, end lsn 32164666892,
ibbackup: start checkpoint 32164666892.
InnoDB: Doing recovery: scanned up to log sequence number 32164666892
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percents: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Setting log file size to 0 10485760
Setting log file size to 0 10485760
ibbackup: We were able to parse ibbackup_logfile up to
ibbackup: lsn 32164666892.
ibbackup: Last MySQL binlog file position 0 534, file name ./MySQL-bin.000008
ibbackup: The first data file is '/sqldata-backup/ibdata1'
ibbackup: and the new created log files are at '/sqldata-backup/'
ibbackup: System tablespace file format is Antelope.
101208 15:47:18  ibbackup: Full backup prepared for recovery successfully!
```

The contents of the backup directory after applying the log. `ibbackup` created InnoDB log files (`ib_logfile*`) and applied log records to the InnoDB data files (`ibdata*`):

```
$ ls -l /sqldata-backup
total 530448
-rw-r--r-- 1 pekka pekka       158 2010-12-08 15:33 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka      1024 2010-12-08 15:33 ibbackup_logfile
-rw-r----- 1 pekka pekka  10485760 2010-12-08 15:32 ibdata1
-rw-r----- 1 pekka pekka  20971520 2010-12-08 15:32 ibdata2
-rw-r----- 1 pekka pekka 480247808 2010-12-08 15:33 ibdata3
-rw-r----- 1 pekka pekka  10485760 2010-12-08 15:47 ib_logfile0
-rw-r----- 1 pekka pekka  10485760 2010-12-08 15:47 ib_logfile1
-rw-r----- 1 pekka pekka  10485760 2010-12-08 15:47 ib_logfile2
```

5.2 Example: Applying the Log to a Compressed Backup

This example continues from Section 4.2, “Example: Making a Compressed Backup of InnoDB Tables”.

If the backup is compressed, specify the `--uncompress` option to `ibbackup` when applying the log to the backup:

```
$ ibbackup --apply-log --uncompress /home/pekka/.backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.backup-my.cnf:
innodb_data_home_dir got value /sqldata-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata-backup
innodb_log_group_home_dir got value /sqldata-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

ibbackup: Uncompressing data file '/sqldata-backup/ibdata1.ibz'
ibbackup: Uncompressing data file '/sqldata-backup/ibdata2.ibz'
ibbackup: Uncompressing data file '/sqldata-backup/ibdata3.ibz'
 400
101208 16:06:16  ibbackup: ibbackup_logfile's creation parameters:
ibbackup: start lsn 32164666880, end lsn 32164666892,
ibbackup: start checkpoint 32164666892.
InnoDB: Doing recovery: scanned up to log sequence number 32164666892
InnoDB: Starting an apply batch of log records to the database...
```

```
InnoDB: Progress in percents: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Setting log file size to 0 10485760
Setting log file size to 0 10485760
ibbackup: We were able to parse ibbackup_logfile up to
ibbackup: lsn 32164666892.
ibbackup: Last MySQL binlog file position 0 534, file name ./MySQL-bin.000008
ibbackup: The first data file is '/sqldata-backup/ibdata1'
ibbackup: and the new created log files are at '/sqldata-backup/'
ibbackup: System tablespace file format is Antelope.
101208 16:06:21  ibbackup: Full backup prepared for recovery successfully!
```

The contents of the backup directory after successfully applying the log to the backup are shown below. The `ibbackup` output above shows that data files were rolled forward to log sequence number (lsn) 32164666892. Since `ibbackup` does not modify any of the original files in the backup (compressed data files and ibbackup log file), nothing is lost if the apply-log operation fails for some reason (for example, insufficient disk space). After fixing the problem, you can safely try the apply-log operation again.

```
$ ls -l /sqldata-backup
total 585120
-rw-r--r-- 1 pekka pekka       158 2010-12-08 15:48 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka      1024 2010-12-08 15:48 ibbackup_logfile
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:05 ibdata1
-rw-r----- 1 pekka pekka   1095854 2010-12-08 15:47 ibdata1.ibz
-rw-r----- 1 pekka pekka  20971520 2010-12-08 16:05 ibdata2
-rw-r----- 1 pekka pekka    811625 2010-12-08 15:47 ibdata2.ibz
-rw-r----- 1 pekka pekka 480247808 2010-12-08 16:06 ibdata3
-rw-r----- 1 pekka pekka  54058462 2010-12-08 15:48 ibdata3.ibz
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:06 ib_logfile0
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:06 ib_logfile1
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:06 ib_logfile2
```

5.3 Example: Applying an Incremental Backup to a Full Backup

This example continues from Section 4.3, “Example: Making an Incremental Backup of InnoDB Tables”.

After you take an incremental backup, the changes reflected in those backup files must be applied to a full backup to bring the full backup up-to-date, in the same way that you apply changes from the binary log.

To bring the data files from the full backup up to date, first we run the apply log step so that the data files include any changes that occurred while the full backup was running:

```
$ ibbackup --apply-log /home/pekka/.full-backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.full-backup-my.cnf:
innodb_data_home_dir got value /full-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /full-backup
innodb_log_group_home_dir got value /full-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

101208 17:13:47  ibbackup: ibbackup_logfile's creation parameters:
ibbackup: start lsn 2638547968, end lsn 2638548215,
ibbackup: start checkpoint 2638548215.
InnoDB: Doing recovery: scanned up to log sequence number 2638548215
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percents: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

```
Setting log file size to 0 10485760
Setting log file size to 0 10485760
ibbackup: We were able to parse ibbackup_logfile up to
ibbackup: lsn 2638548215.
ibbackup: Last MySQL binlog file position 0 538, file name ./MySQL-bin.000046
ibbackup: The first data file is '/full-backup/ibdata1'
ibbackup: and the new created log files are at '/full-backup/'
ibbackup: System tablespace file format is Barracuda.
101208 17:13:48  ibbackup: Full backup prepared for recovery successfully!
```

Then we apply the changes from the incremental backup, to the data files produced by the full backup:

```
$ ibbackup --apply-log --incremental /home/pekka/.incr-backup-my.cnf /home/pekka/.full-backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.incr-backup-my.cnf:
innodb_data_home_dir got value /incr-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /incr-backup
innodb_log_group_home_dir got value /incr-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /home/pekka/.full-backup-my.cnf:
innodb_data_home_dir got value /full-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /full-backup
innodb_log_group_home_dir got value /full-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

101208 17:13:49  ibbackup: ibbackup_logfile's creation parameters:
ibbackup: start lsn 2654252032, end lsn 2654252454,
ibbackup: start checkpoint 2654252454.
InnoDB: Doing recovery: scanned up to log sequence number 2654252454
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percents: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Setting log file size to 0 10485760
Setting log file size to 0 10485760
ibbackup: We were able to parse ibbackup_logfile up to
ibbackup: lsn 2654252454.
ibbackup: Last MySQL binlog file position 0 538, file name ./MySQL-bin.000046
ibbackup: The first data file is '/full-backup/ibdata1'
ibbackup: and the new created log files are at '/full-backup/'
ibbackup: System tablespace file format is Barracuda.
101208 17:13:51  ibbackup: Incremental backup applied successfully!
```

Now the data files that are labelled as the full backup are fully up-to-date, as of the time of the incremental backup:

```
$ ls -hlR /full-backup/
/full-backup/:
total 135M
-rw-r--r-- 1 pekka pekka  155 2010-12-08 17:11 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka 1.0K 2010-12-08 17:13 ibbackup_logfile
-rw-r----- 1 pekka pekka  10M 2010-12-08 17:13 ibdata1
-rw-r----- 1 pekka pekka  20M 2010-12-08 17:13 ibdata2
-rw-r----- 1 pekka pekka  74M 2010-12-08 17:11 ibdata3
-rw-r----- 1 pekka pekka  10M 2010-12-08 17:13 ib_logfile0
-rw-r----- 1 pekka pekka  10M 2010-12-08 17:13 ib_logfile1
-rw-r----- 1 pekka pekka  10M 2010-12-08 17:13 ib_logfile2
```

```
drwxr-x--- 2 pekka pekka   48 2010-12-08 17:11 test

/full-backup/test:
total 73M
-rw-r----- 1 pekka pekka  23M 2010-12-08 17:11 alex1.ibd
-rw-r----- 1 pekka pekka 240K 2010-12-08 17:11 alex2.ibd
-rw-r----- 1 pekka pekka 240K 2010-12-08 17:11 alex3.ibd
-rw-r----- 1 pekka pekka  23M 2010-12-08 17:13 blobt3.ibd
-rw-r----- 1 pekka pekka  25M 2010-12-08 17:11 ibstest0.ibd
-rw-r----- 1 pekka pekka 912K 2010-12-08 17:11 ibtest09.ibd
```

5.4 Starting mysqld on a Restored Backup

After applying the log and creating the new ib_logfiles, we are ready to start mysqld on our backup database.

The backup database directory looks like this before we start mysqld. Note that the backup directory contains database subdirectories mysql, test, test115. These are not created by ibbackup, but they are created either manually by the user or automatically by the mysqlbackup command (see Chapter 7, *mysqlbackup Command Reference*).

```
$ ls -lh /sqldata-backups
total 872M
-rw-r-----    1 pekka    dev           41M Jan 22 15:33 ibbackup_logfile
-rw-r-----    1 pekka    dev          100M Jan 22 15:38 ibdata1
-rw-r-----    1 pekka    dev          200M Jan 22 15:38 ibdata2
-rw-r-----    1 pekka    dev          500M Jan 22 15:32 ibdata3
-rw-r-----    1 pekka    dev           10M Jan 22 15:40 ib_logfile0
-rw-r-----    1 pekka    dev           10M Jan 22 15:38 ib_logfile1
-rw-r-----    1 pekka    dev           10M Jan 22 15:38 ib_logfile2
drwxr-xr-x    2 pekka    dev          4.0k Jan 22 15:33 mysql
drwxr-xr-x    2 pekka    dev          4.0k Jan 22 15:33 test
drwxr-xr-x    2 pekka    dev          4.0k Jan 22 15:33 test115
```

We start mysqld on the backup database:

```
$ mysqld --defaults-file=/home/pekka/.backup-my.cnf
040122 15:41:57  InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
InnoDB: Reading tablespace information from the .ibd files...
InnoDB: Restoring possible half-written data pages from the doublewrite
InnoDB: buffer...
040122 15:41:57  InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 646218252.
InnoDB: Doing recovery: scanned up to log sequence number 0 646218252
InnoDB: 2 transaction(s) which must be rolled back or cleaned up
InnoDB: in total 18 row operations to undo
InnoDB: Trx id counter is 0 239872
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx with id 0 239533, 16 rows to undo
InnoDB: Rolling back of trx id 0 239533 completed
InnoDB: Rolling back trx with id 0 239532, 2 rows to undo
InnoDB: Rolling back of trx id 0 239532 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Last MySQL binlog file position 0 27183537, file name ./binlog.000005
040122 15:41:57  InnoDB: Flushing modified pages from the buffer pool...
040122 15:41:59  InnoDB: Started; log sequence number 0 646218252
040122 15:41:59  mysql.user table is not updated to new password format;
Disabling new password usage until mysql_fix_privilege_tables is run
mysqld: ready for connections.
```

Now mysqld is running on the restored backup database and ready to serve clients.

If you originally took the backup from a MySQL replication slave, then mysqld also prints the **master** database binlog position when you start mysqld on the backup.

Chapter 6 Recovering or Restoring a Database

Table of Contents

6.1 Point-in-Time Recovery from a Hot Backup

InnoDB only stores the binlog position information to its tablespace at a transaction commit. **To make InnoDB aware of the current binlog position, you must run at least one transaction while binlogging is enabled.** When you run `ibbackup --apply-log` on your backup, `ibbackup` versions >= 1.03 print the latest MySQL binlog position the backup knows of. Also, `mysqld` prints it when you start it on the backup after the `--apply-log`:

```
$ mysqld --defaults-file=/home/pekka/.backup-my.cnf
040122 15:41:57  InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
…
InnoDB: Last MySQL binlog file position 0 27183537, file name ./binlog.000005
…
mysqld: ready for connections.
```

The MySQL version must be >= 5.1.

The printed position is the MySQL binlog byte position from the moment when MySQL Enterprise Backup finished the copying of your data files. Then you can apply the binlog file(s) starting from that position to the restored database:

```
$ mysqlbinlog --start-position=27183537 /sqldata/binlog.000005 | mysql
```

To recover the database to a specific point in time, direct the output of `mysqlbinlog` to an output file, instead of piping it directly to `mysql`. This output file contains timestamps for all SQL statements in the binlog. In an editor, remove all statements after the specified point in time. Process the modified file with `mysql`, like this:

```
$ mysql < modified_output_file
```

6.2 Setting Up a New Slave from a Hot Backup in Replication

If you use *MySQL replication* to replicate InnoDB tables, MySQL Enterprise Backup allows you to set up a slave database without stopping the master because you can make a restored hot backup a new slave database.

1. Take the backup, use `ibbackup --apply-log` to restore it, and put the restored backup and the log files to the right places for the new slave.

2. Edit the `my.cnf` file of the new slave so that you put `skip-slave-start` to the `[mysqld]` section.

3. Start the new slave `mysqld` (version >= 5.1). It prints the latest MySQL binlog position the backup knows of.

```
…
InnoDB: Last MySQL binlog file position 0 128760128, file name ./hundin-bin.006
…
```

   Note that InnoDB only stores the binlog position information to its tablespace at a transaction commit. **To make InnoDB aware of the current binlog position, you must run at least one transaction while binlogging is enabled.**

4. Use the `CHANGE MASTER` SQL command on the slave to initialize it properly. For example:

```
CHANGE MASTER TO
MASTER_LOG_FILE='hundin-bin.006',
MASTER_LOG_POS=128760128;
```

5. Start replication in the new slave with the `SLAVE START` SQL command.

6. Remove the line `skip-slave-start` from the `my.cnf` file of the slave.

6.3 Restoring a Master Database in Replication

Let us assume a master database gets corrupt.

1. Use the backup of the master database, do `ibbackup --apply-log yourbackupmy.cnf` and put the `ibdata` and `ib_logfile` files to the right places. Then put the `.frm` files to the right place. Let us assume you have a tar file of the `.frm` files: `cd mysqldatadir; tar xvf yourtarfile`

2. Edit the master `my.cnf` file so that you comment out `log-bin` in it so that the slaves do not receive twice the binlog needed to recover the master.

3. Replication in the slaves must be stopped temporarily when you do the piping of the binlog to the master. In the slaves, do

```
mysql> STOP SLAVE;
```

4. Start the master `mysqld` on the restored backup:

```
$ mysqld
…
InnoDB: Doing recovery: scanned up to log sequence number 0 64300044
InnoDB: Last MySQL binlog file position 0 5585832, file name
./omnibook-bin.002
…
```

   InnoDB printed the binlog file and position it was able to recover to.

5. Next you should pipe the remaining binlog files to the restored backup:

```
$ mysqlbinlog --start-position=5585832mysqldatadir/omnibook-bin.002 | mysql
$ mysqlbinlog /mysqldatadir/omnibook-bin.003 | mysql
```

6. The master database is now recovered. Shut down the master and edit `my.cnf` to uncomment `log-bin`.

7. Start the master again.

8. Start replication in the slaves again:

```
mysql> START SLAVE;
```

Chapter 7 `mysqlbackup` Command Reference

Table of Contents

The `mysqlbackup` command is an easy-to-use tool for making a more complete backup than by using just the `ibbackup` command. The `mysqlbackup` command backs up:

- All the same InnoDB tables, indexes, and so on as `ibbackup`. (`mysqlbackup` runs `ibbackup` to perform this part of the backup.)
- MyISAM tables.
- By default, all files in the data directory are included in the backup. If the `--only-known-file-types` option is specified, the backup includes additional files with only these file extensions:
  - .ARM: Archive storage engine metadata
  - .ARZ: Archive storage engine data
  - .FRM: table definitions
  - .MRG: Merge storage engine references to other tables
  - .MYD: MyISAM data
  - .MYI: MyISAM indexes
  - .OPT: database configuration information
  - .PAR: partition definitions
  - .TRG: trigger parameters
  - .TRN: trigger namespace information

In addition to creating backups, `mysqlbackup` can prepare a backup for starting a MySQL server on the backup, and it can copy data, index, and log files from backup directory back to their original locations.

`mysqlbackup` is a command that you can use to take an online backup of your InnoDB tables, and a snapshot of your MyISAM tables which correspond to the same binlog position as the backup of InnoDB tables. It also backs up the `.frm` files of the tables.

A sample command line to start `mysqlbackup` is:

```
$ mysqlbackup --user=dba --password=xyz --compress /etc/my.cnf /backups
```

The `--user` and the `--password` you specify are used to connect to the MySQL server. This MySQL user must have enough **rights** in the MySQL server to execute `FLUSH TABLES WITH READ LOCK` and to create a dummy marker table `ibbackup_binlog_marker` in the `mysql` system database in the server (see Section 7.6, “Example: Setting MySQL Privileges for mysqlbackup” for details on the required MySQL privileges). In a backup run, `mysqlbackup` places the backup in a subdirectory it creates under the directory `/backups` you specified above. The name of the backup subdirectory is formed from the date and the clock time of the backup run.

Make sure that the user or the cron job running `mysqlbackup` has the **rights** to copy files from the MySQL database directories to the backup directory.

Make sure that your **connection timeouts are long enough** so that the command can keep the connection to the server open for the duration of the backup run. `mysqlbackup` pings the server after copying each database to keep the connection alive.

When `mysqlbackup` is run, it first tests a connection to the MySQL server. Then, it calls `ibbackup` and takes an online backup of InnoDB tables. (This phase does not disturb normal database processing.) When the `ibbackup` run has almost completed, `mysqlbackup` executes the SQL command `FLUSH TABLES WITH READ LOCK` and then copies the MyISAM tables and `.frm` files to the backup directory. If you do not run long `SELECT` or other queries in the database at this time, and your MyISAM tables are small, the locked phase only lasts a couple of seconds. Otherwise, the whole database, including InnoDB type tables, can be locked for quite a while. After this, `mysqlbackup` lets `ibbackup` run to completion and `UNLOCK`s the tables.

> **IMPORTANT:**
>
> - Although the `mysqlbackup` command backs up InnoDB tables (ideally, the bulk of your data) without interrupting database use, the final stage that copies non-InnoDB files does temporarily put the database into a read-only state. For best backup performance and minimal impact on database processing:
>
>   1. Do not run long `SELECT` or other SQL queries at the time of the backup run.
>   2. Keep your MyISAM tables relatively small.
>
>   Then the locked phase at the end of an `mysqlbackup` run is short (maybe a few seconds), and does not disturb the normal processing of `mysqld` much. If the above two conditions are not met in your database application, use the plain `ibbackup` binary to take the backups.
> - For a large database, a backup run may take a long time. Always check that `mysqlbackup` has completed successfully, either by verifying that the `mysqlbackup` command returned exit code 0, or by observing that `mysqlbackup` has printed the text “mysqlbackup completed OK!”.
> - The `mysqlbackup` command is not the same as the former “MySQL Backup” open source project from the MySQL 6.0 source tree. The MySQL Enterprise Backup product supersedes the MySQL Backup initiative.
> - Schedule backups during periods when no DDL operations involving tables are running. See Section A.1, “Limitations of `mysqlbackup` and `ibbackup` Commands” for restrictions on backups at the same time as DDL operations.

7.1 mysqlbackup Command-Line Options

```
$ mysqlbackup --help
```

```
Usage:
mysqlbackup [--sleep=MS] [--compress[=LEVEL]] [--include=REGEXP]
            [--user=WORD][--password=WORD] [--port=PORT]
            [--socket=SOCKET] [--no-timestamp]
            [--ibbackup=IBBACKUP-BINARY] [--slave-info]
            [--backup-and-apply-log] [--databases=LIST]
            [--exec-when-locked="utility arg1 arg2 ..."]
            [--incremental --lsn=LSN]
            [--only-known-file-types]
            MY.CNF BACKUP-ROOT-DIR

mysqlbackup --apply-log [--use-memory=MB] [--uncompress]
            [--ibbackup=IBBACKUP-BINARY]  MY.CNF BACKUP-DIR

mysqlbackup --apply-log --incremental [--use-memory=MB] [--uncompress]
            [--ibbackup=IBBACKUP-BINARY]
            INCREMENTAL-BACKUP-MY.CNF FULL-BACKUP-MY.CNF

mysqlbackup --copy-back MY.CNF BACKUP-DIR

The first command line above makes a hot backup of a MySQL database. By
default it creates a backup directory (named by the current date and time)
in the given backup root directory.  With the --no-timestamp option it
does not create a time-stamped backup directory, but it puts the backup in
the given directory (which must not exist). This command makes a complete
backup of all MyISAM and InnoDB tables and indexes in all databases or in
all of the databases specified with the --databases option. The created
backup contains by default all InnoDB data and log files, .frm files, and
all files in the subdirectories of the database directory. Please notice
that backups of other than MyISAM and InnoDB engines may not be valid,
even if mysqlbackup copies all files associated with them, because the
engine might not flush all data to disk when mysqlbackup locks all
databases.

If --incremental is specified, it instructs ibbackup to make an INCREMENTAL
backup that only contains data pages whose latest modification has a log
sequence number (lsn) greater than LSN. This LSN must be specified via the
--lsn switch. Make sure that you have taken a full backup before taking an
incremental backup or you will not be able to restore your database.

The MY.CNF options file defines the location
of the database. This command connects to the MySQL server using
mysql client API, and runs ibbackup (InnoDB Hot Backup program) as
a child process.

The command with --apply-log option prepares a backup for starting a MySQL
server on the backup. This command expands InnoDB data files as specified
in BACKUP-DIR/backup-my.cnf using BACKUP-DIR/ibbackup_logfile,
and creates new InnoDB log files as specified in BACKUP-DIR/backup-my.cnf.
The BACKUP-DIR should be a path name of a backup directory created by
mysqlbackup. This command runs ibbackup as a child process, but it does not
connect to the database server.

If --incremental is specified after --apply-log, then ibbackup applies the
incremental backup at INCREMENTAL-BACKUP-MY.CNF to the FULL backup at
FULL-BACKUP-MY.CNF.

The command with --copy-back option copies data, index, and log files
from backup directory back to their original locations.
The MY.CNF options file defines the original location of the database.
The BACKUP-DIR is a path name of a backup directory created by mysqlbackup.

mysqlbackup reads [client] group only from the configuration files.
The configuration file MY.CNF or INCREMENTAL-BACKUP.CNF is read after all
the default configuration files are read. mysqlbackup acts only on user,
password, port and socket options from [client] group and ignores the rest
of the options.
```

```
On success the exit code of mysqlbackup process is 0. A non-zero exit code
indicates an error.

Options:
    --help      Display this helpscreen and exit.

    --version   Print version information and exit.

    --apply-log Prepare a backup for starting mysql server on the backup.
                Expand InnoDB data files as specified in
                backup-dir/backup-my.cnf, using backup-dir/ibbackup_logfile,
                and create new log files as specified in
                backup-dir/backup-my.cnf.

    --backup-and-apply-log
                Make a backup and prepare it for starting mysql server on it
                as if --apply-log option had been given. The --incremental
                option is not allowed with this option. The --compress and
                --uncompress options are ignored with this option.

    --copy-back Copy data and index files from backup directory back to
                their original locations.

    --use-memory=MB
                This option is passed to the ibbackup child process.
                It tells ibbackup that it can use MB megabytes of
                memory in restoration.
                Try 'ibbackup --help' for more details on this option.

    --sleep=MS  This option is passed to the ibbackup child process.
                You can use this parameter to tune the additional
                MS milliseconds after each 1 MB of copied data.
                disk i/o load the ibbackup program causes on the computer.
                Try 'ibbackup --help' for more details on this option.

    --compress[=LEVEL]
                This option is passed to the ibbackup child process.
                It instructs ibbackup to compress the backup copies of
                InnoDB data files. Compression level can be
                specified as an optional argument. Compression level is
                an integer between 0 and 9: 1 gives fastest compression,
                9 gives best compression, and 0 means no compression.
                If compression level is not given, the default level 1 is
                used. Try 'ibbackup --help' for more details on this option.

    --include=REGEXP
               This option is passed to the ibbackup child process.
               It tells ibbackup to backup only those per-table data
               files which match the given regular expression.  For
               each table with a per-table data file a string of the
               form db_name.table_name is checked against the regular
               expression.  If the regular expression matches the
               complete string db_name.table_name, the table is
               included in the backup. The regular expression should
               be of the POSIX 1003.2 "extended" form.
               Try 'ibbackup --help' for more details on this option.

    --databases=LIST
                This option is used to specify the list of databases that
                mysqlbackup should backup. The list is of the form
                "db_name[.table_name] db_name1[.table_name1] ...".
                If this option is not specified all databases containing
                MyISAM and InnoDB tables will be backed up.
                Please make sure that --databases contains all of the
                Innodb databases & tables so that all of the innodb .frm
                files are also backed up. In Unix variant systems, if the LIST
                argument begins with a slash, mysqlbackup assumes that it is
```

```
            a pathname of file and tries to read the list of databases
            from that file. In Windows systems, if the list argument
            begins with a "<drive_letter>:\", mysqlbackup assumes
            that it is a pathname of file and tries to read the list
            of databases from that file.

--uncompress
            This option is passed to the ibbackup child process.
            It tells ibbackup to uncompress compressed InnoDB data files.
            Try 'ibbackup --help' for more details on this option.

--user=NAME It defines the user for database login if not current user.

--password=WORD
            It defines the password to use when connecting to database.
            If WORD is not given, a prompt for the same is issued on the
            tty.

--port=PORT It defines the port to use when connecting to local database
            server with TCP/IP.

--slave-info
            This option is useful when backing up a replication
            slave server. It prints the binary log position and
            name of the binary log file of the master server.
            It also writes this information to the 'ibbackup_slave_info'
            file as a 'CHANGE MASTER' command. A new slave for this
            master can be set up by starting a slave server on this
            backup and issuing a 'CHANGE MASTER' command with the
            binary log position saved in the 'ibbackup_slave_info' file.

--socket=SOCKET
            It defines the socket to use when connecting to local
            database server with UNIX domain socket.

--no-timestamp
            This option prevents the creation of a new backup
            directory (named by the current date and time) under
            the backup root directory. Instead, the backup is put
            in the directory given on the command-line (in the
            place of BACKUP-ROOT-DIR argument). The directory must not
            exist, because mysqlbackup creates it while making a backup.

--ibbackup=IBBACKUP-BINARY
            Use this option to specify which ibbackup (InnoDB Hot
            Backup) binary should be used. IBBACKUP-BINARY
            should be the command used to run ibbackup. This can
            be useful if ibbackup is not in your search path or
            working directory. If this option is not given,
            ibbackup is run with command "./ibbackup" in Unix
            like systems and "ibbackup.exe" in Windows systems.

--exec-when-locked="utility arg1 arg2 ..."
            The specified utility is executed when all tables are
            locked near the end of the execution. This can be used
            to mysqldump memory tables (engine=memory) and get a
            consistent backup.
            BACKUP_DIR will be set in the environment of the utility
            to the directory that contains the backup (a subdirectory
            of the specified BACKUP-ROOT-DIR).
            For example, in Unix like systems:
            --exec-when-locked='mysqldump mydb t1 > $BACKUP_DIR/t1.sql'
            Note the single quotes that prevent your shell from
            interpreting $BACKUP_DIR before starting mysqlbackup.
            In Windows systems:
            --exec-when-locked="mysqldump mydb t1 > %BACKUP_DIR%/t1.sql"
            If the utility cannot be executed or if it returns
```

```
                failure (non-zero exit status), then the whole backup
                process will be aborted.

    --only-known-file-types
                Backup only files used in MySQL 5.1 by InnoDB, MyISAM, and
                ARCHIVE engines. With this option the backup contains
                InnoDB data and log files, table definitions (.frm files),
                merge tables (.MRG), MyISAM tables and indexes (.MYD and
                .MYI), triggers (.TRG and .TRN), database characteristics
                (.opt), partitions (.par) and ARCHIVE tables (.ARM and .ARZ).
```

7.2 Example: Backing Up InnoDB and MyISAM Tables

In this example, `mysqlbackup` takes two arguments: the options file of the MySQL installation to be backed up, and a backup root directory. The last command line argument `/backups` is the root directory in which `mysqlbackup` creates the backup directory.

```
$ mysqlbackup /home/pekka/.my.cnf /backups
mysqlbackup: Starting mysqlbackup with following arguments:
mysqlbackup /home/pekka/.my.cnf /backups
mysqlbackup: The unique backup id generated for the current backup operation is 12918179022011620

mysqlbackup: IMPORTANT: Please check that backup run completes successfully.
            At the end of a successful 'backup' run mysqlbackup
            prints "mysqlbackup completed OK!".

mysqlbackup: Created backup directory '/backups/2010-12-08_16-18-22'
mysqlbackup: Using ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
mysqlbackup: Using MySQL client version: 5.1.47
mysqlbackup: Checking a connection to MySQL Server with parameters:
mysqlbackup: port=3306, socket=/home/pekka/mysql/MySQL.socket
mysqlbackup: Using MySQL server version: 5.1.37

101208 16:18:22 mysqlbackup: Starting ibbackup binary with args:

./ibbackup --suspend-at-end /home/pekka/.my.cnf /backups/2010-12-08_16-18-22/backup-my.cnf
mysqlbackup: Waiting for ibbackup process to suspend
mysqlbackup: Suspend file '/backups/2010-12-08_16-18-22/ibbackup_suspended'
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.my.cnf:
innodb_data_home_dir got value /sqldata/simple
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata/simple
innodb_log_group_home_dir got value /sqldata/simple
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /backups/2010-12-08_16-18-22/backup-my.cnf:
innodb_data_home_dir got value /backups/2010-12-08_16-18-22
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /backups/2010-12-08_16-18-22
innodb_log_group_home_dir got value /backups/2010-12-08_16-18-22
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

ibbackup: System tablespace file format is Antelope.
ibbackup: Found checkpoint at lsn 32164666892.
ibbackup: Starting log scan from lsn 32164666880.
101208 16:18:22  ibbackup: Copying log...
```

```
101208 16:18:22  ibbackup: Log copied, lsn 32164666892.
ibbackup: We wait 1 second before starting copying the data files...
101208 16:18:23  ibbackup: Copying /sqldata/simple/ibdata1 (Antelope file format).
101208 16:18:24  ibbackup: Copying /sqldata/simple/ibdata2 (Antelope file format).
101208 16:18:25  ibbackup: Copying /sqldata/simple/ibdata3 (Antelope file format).
ibbackup: Progress in MB: 200 400

ibbackup: You specified the option --suspend-at-end.

101208 16:19:05 mysqlbackup: Continuing after ibbackup has suspended

101208 16:19:05 mysqlbackup: Starting to lock all the tables....

101208 16:19:06 mysqlbackup: All tables are locked and flushed to disk
mysqlbackup: Opening backup source directory '/sqldata/simple'

101208 16:19:06 mysqlbackup: Starting to backup all files in subdirectories of '/sqldata/simple'
mysqlbackup: Backing up the database directory 'mysql'
mysqlbackup: Backing up the database directory 'test'
mysqlbackup: Resuming ibbackup

101208 16:19:05  ibbackup: Suspending the backup procedure to wait
ibbackup: until you delete the marker file /backups/2010-12-08_16-18-22/ibbackup_suspended.
101208 16:19:06  ibbackup: Suspension ends. Continuing the backup procedure.

101208 16:19:06  ibbackup: Copying of the last data file is close to ending...
ibbackup: We still once copy the latest flushed log to ibbackup_logfile.
ibbackup: A copied database page was modified at 32164665879.
ibbackup: Scanned log up to lsn 32164666892.
ibbackup: Was able to parse the log up to lsn 32164666892.
ibbackup: Maximum page number for a log record 0
101208 16:19:06  ibbackup: Full backup completed!

101208 16:19:06 mysqlbackup: All tables unlocked
mysqlbackup: All MySQL tables were locked for 0.887 seconds
mysqlbackup: Backup created in directory '/backups/2010-12-08_16-18-22'
mysqlbackup: start_lsn: 32164666880
mysqlbackup: incremental_base_lsn: 32164665879
mysqlbackup: end_lsn: 32164666892

101208 16:19:07 mysqlbackup: mysqlbackup completed OK!
```

Now we see the backup directory under the `BACKUP-ROOT-DIR` we specified. The directory name for each new backup is formed from the date and the clock time when the backup run was started, in the local time zone.

```
$ ls -l /backups
total 72
drwxr-xr-x 4 pekka pekka 4096 2010-03-18 15:34 2009-10-27_16-44-38
drwxr-xr-x 4 pekka pekka 4096 2009-12-14 18:56 2009-12-14_18-56-02
drwxr-xr-x 4 pekka pekka   88 2010-03-25 17:04 2010-03-25_17-03-34
drwxr-xr-x 4 pekka pekka 4096 2010-05-17 18:26 2010-05-17_18-23-52
drwxr-xr-x 4 pekka pekka 4096 2010-10-04 12:14 2010-05-17_18-35-41
drwx------ 4 pekka pekka   80 2010-09-15 13:17 2010-09-15_13-16-19
drwx------ 4 pekka pekka   80 2010-09-16 14:19 2010-09-16_14-17-59
drwx------ 4 pekka pekka   80 2010-09-20 13:37 2010-09-20_13-36-07
drwx------ 4 pekka pekka   80 2010-09-27 12:56 2010-09-27_12-54-57
drwx------ 4 pekka pekka   80 2010-09-30 15:25 2010-09-30_15-24-23
drwx------ 4 pekka pekka 4096 2010-12-08 16:18 2010-10-04_12-14-47
drwx------ 2 pekka pekka    1 2010-12-08 14:57 2010-12-08_14-57-34
drwx------ 4 pekka pekka   80 2010-12-08 15:17 2010-12-08_15-16-42
drwx------ 4 pekka pekka   80 2010-12-08 16:19 2010-12-08_16-18-22
```

The backup directory contains the backed-up `ibdata` files and `ibbackup_logfile`. Its subdirectories `mysql`, `test`, `test115` are copies of the database directories, and contain copies of `.frm`, `.MYD`, and `.MYI` files:

```
$ ls -l /backups/2010-12-08_16-18-22
total 499740
-rw-r--r-- 1 pekka pekka       347 2010-12-08 16:18 backup-my.cnf
-rw-r--r-- 1 pekka pekka        18 2010-12-08 16:19 ibbackup_binlog_info
-rw-r--r-- 1 pekka pekka       158 2010-12-08 16:19 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka      1024 2010-12-08 16:19 ibbackup_logfile
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:18 ibdata1
-rw-r----- 1 pekka pekka  20971520 2010-12-08 16:18 ibdata2
-rw-r----- 1 pekka pekka 480247808 2010-12-08 16:19 ibdata3
drwx------ 2 pekka pekka      4096 2010-12-08 16:19 mysql
drwx------ 2 pekka pekka        24 2010-12-08 16:19 test
```

7.3 Example: Verifying a Backup

This example continues from where we ended up at the end of the previous example. To verify the backup, we will run the MySQL daemon (`mysqld`) on the backup data. Then we can execute queries to verify the data.

We prepare the backup for starting the database server on it by applying the log records (in file `ibbackup_logfile`) to the InnoDB data files (`ibdata*`).

```
$ mysqlbackup --apply-log /home/pekka/.my.cnf /backups/2010-12-08_16-18-22
mysqlbackup: Starting mysqlbackup with following arguments:
mysqlbackup --apply-log /home/pekka/.my.cnf /backups/2010-12-08_16-18-22
mysqlbackup: IMPORTANT: Please check that apply-log run completes successfully.
            At the end of a successful 'apply-log' run mysqlbackup
            prints "mysqlbackup completed OK!".

mysqlbackup: Using ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
mysqlbackup: Starting ibbackup binary with args:

./ibbackup --apply-log /backups/2010-12-08_16-18-22/backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /backups/2010-12-08_16-18-22/backup-my.cnf:
innodb_data_home_dir got value /backups/2010-12-08_16-18-22
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /backups/2010-12-08_16-18-22
innodb_log_group_home_dir got value /backups/2010-12-08_16-18-22
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

101208 16:23:44  ibbackup: ibbackup_logfile's creation parameters:
ibbackup: start lsn 32164666880, end lsn 32164666892,
ibbackup: start checkpoint 32164666892.
InnoDB: Doing recovery: scanned up to log sequence number 32164666892
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percents: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Setting log file size to 0 10485760
Setting log file size to 0 10485760
ibbackup: We were able to parse ibbackup_logfile up to
ibbackup: lsn 32164666892.
ibbackup: Last MySQL binlog file position 0 534, file name ./MySQL-bin.000008
ibbackup: The first data file is '/backups/2010-12-08_16-18-22/ibdata1'
```

```
ibbackup: and the new created log files are at '/backups/2010-12-08_16-18-22/'
ibbackup: System tablespace file format is Antelope.
101208 16:23:45  ibbackup: Full backup prepared for recovery successfully!

101208 16:23:45 mysqlbackup: mysqlbackup completed OK!
```

Here is the backup directory after successfully applying the log to the backup. The `ibbackup` output above shows that data files were rolled forward to log sequence number (lsn) `928223244`.

```
$ ls -l /backups/2010-12-08_16-18-22
total 530464
-rw-r--r-- 1 pekka pekka       347 2010-12-08 16:18 backup-my.cnf
-rw-r--r-- 1 pekka pekka        18 2010-12-08 16:19 ibbackup_binlog_info
-rw-r--r-- 1 pekka pekka       158 2010-12-08 16:19 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka      1024 2010-12-08 16:19 ibbackup_logfile
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:18 ibdata1
-rw-r----- 1 pekka pekka  20971520 2010-12-08 16:18 ibdata2
-rw-r----- 1 pekka pekka 480247808 2010-12-08 16:19 ibdata3
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:23 ib_logfile0
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:23 ib_logfile1
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:23 ib_logfile2
drwx------ 2 pekka pekka      4096 2010-12-08 16:19 mysql
drwx------ 2 pekka pekka        24 2010-12-08 16:19 test
```

We are now almost ready to start a server on this backup. The final requirement is to prepare a valid `my.cnf` file for the backup. We cannot reuse the `backup-my.cnf` in the backup directory, because it contains only the small subset of parameters required by `ibbackup`. We must create a `my.cnf` file by copying these parameters from the `backup-my.cnf` file, and copying the other parameters from the original `my.cnf` file (which is `/home/pekka/.my.cnf` in this example). We make a copy of the original file and append the additional parameters in `backup-my.cnf`.

```
$ cat /home/pekka/.my.cnf  /backups/2010-12-08_16-18-22/backup-my.cnf > /backups/2010-12-08_16-18-22/my.cnf
```

Now we can start a server on the backup, specifying this new `my.cnf` as the configuration file:

```
$ mysqld --defaults-file=/backups/2010-12-08_16-18-22/my.cnf
101208 16:23:45 [Warning] Changed limits: max_open_files: 1024  max_connections: 200  table_cache: 407
101208 16:23:45 [Note] Plugin 'FEDERATED' is disabled.
InnoDB: The log file was created by ibbackup --apply-log at
InnoDB: ibbackup 101208 16:23:45
InnoDB: NOTE: the following crash recovery is part of a normal restore.
InnoDB: The log sequence number in ibdata files does not match
InnoDB: the log sequence number in the ib_logfiles!
101208 16:23:46  InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
InnoDB: Reading tablespace information from the .ibd files...
InnoDB: Restoring possible half-written data pages from the doublewrite
InnoDB: buffer...
InnoDB: Last MySQL binlog file position 0 534, file name ./MySQL-bin.000008
101208 16:23:46  InnoDB: Started; log sequence number 7 2099896332
101208 16:23:46 [Note] Event Scheduler: Loaded 0 events
101208 16:23:46 [Note] mysqld: ready for connections.
```

7.4 Example: Restoring a Database at its Original Location

The log files have been applied to the backup (in `/backups/2004-02-03_13-27-09`):

```
$ ls -l /backups/2010-12-08_16-18-22
total 530468
-rw-r--r-- 1 pekka pekka       347 2010-12-08 16:18 backup-my.cnf
-rw-r--r-- 1 pekka pekka        18 2010-12-08 16:19 ibbackup_binlog_info
-rw-r--r-- 1 pekka pekka       158 2010-12-08 16:19 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka      1024 2010-12-08 16:19 ibbackup_logfile
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:24 ibdata1
-rw-r----- 1 pekka pekka  20971520 2010-12-08 16:24 ibdata2
-rw-r----- 1 pekka pekka 480247808 2010-12-08 16:24 ibdata3
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:24 ib_logfile0
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:23 ib_logfile1
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:23 ib_logfile2
-rw-r--r-- 1 pekka pekka      3600 2010-12-08 16:23 my.cnf
drwx------ 2 pekka pekka      4096 2010-12-08 16:19 mysql
drwx------ 2 pekka pekka        24 2010-12-08 16:19 test
```

We copy InnoDB and MyISAM indexes, and `.frm` files back to their original locations (defined by `/home/pekka/.my.cnf` file):

```
$ mysqlbackup --copy-back /home/pekka/.my.cnf /backups/2010-12-08_16-18-22
mysqlbackup: Starting mysqlbackup with following arguments:
mysqlbackup --copy-back /home/pekka/.my.cnf /backups/2010-12-08_16-18-22
mysqlbackup: IMPORTANT: Please check that copy-back run completes successfully.
            At the end of a successful 'copy-back' run mysqlbackup
            prints "mysqlbackup completed OK!".

mysqlbackup: Starting to copy back files
mysqlbackup: in '/backups/2010-12-08_16-18-22' directory
mysqlbackup: back to original data directory '/sqldata/simple'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/ibbackup_binlog_info'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/my.cnf'
mysqlbackup: Copying back directory '/backups/2010-12-08_16-18-22/mysql'
mysqlbackup: Copying back directory '/backups/2010-12-08_16-18-22/test'
mysqlbackup: Starting to copy back InnoDB tables and indexes
mysqlbackup: in '/backups/2010-12-08_16-18-22'
mysqlbackup: back to original InnoDB datadirectory '/sqldata/simple'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/ibdata1'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/ibdata2'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/ibdata3'
mysqlbackup: Starting to copy back InnoDB log files
mysqlbackup: in '/backups/2010-12-08_16-18-22'
mysqlbackup: back to original InnoDB log directory '/sqldata/simple'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/ib_logfile0'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/ib_logfile1'
mysqlbackup: Copying back file '/backups/2010-12-08_16-18-22/ib_logfile2'
mysqlbackup: Finished copying backup files.


101208 16:48:13 mysqlbackup: mysqlbackup completed OK!
```

The original database directory is now restored from the backup:

```
$ ls -l /sqldata/simple
total 531292
-rw-r--r-- 1 pekka pekka       533 2010-12-08 15:47 backup-my.cnf
-rw-r--r-- 1 pekka pekka        18 2010-12-08 16:47 ibbackup_binlog_info
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:47 ibdata1
-rw-r----- 1 pekka pekka  20971520 2010-12-08 16:47 ibdata2
-rw-r----- 1 pekka pekka 480247808 2010-12-08 16:48 ibdata3
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:48 ib_logfile0
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:48 ib_logfile1
-rw-r----- 1 pekka pekka  10485760 2010-12-08 16:48 ib_logfile2
-rw-r--r-- 1 pekka pekka      3600 2010-12-08 16:47 my.cnf
```

```
drwx------ 2 pekka pekka      4096 2010-05-17 18:41 mysql
drwx------ 2 pekka pekka    827392 2010-03-22 16:13 test
```

and we can start a server on it:

```
$ mysqld
101208 16:48:14 [Warning] Changed limits: max_open_files: 1024  max_connections: 200  table_cache: 407
101208 16:48:14 [Warning] One can only use the --user switch if running as root

101208 16:48:14 [Note] Plugin 'FEDERATED' is disabled.
101208 16:48:15  InnoDB: Started; log sequence number 7 2099896332
101208 16:48:15 [Note] Event Scheduler: Loaded 0 events
101208 16:48:15 [Note] mysqld: ready for connections.
```

7.5 Specifying MySQL Connection Settings for mysqlbackup

When `mysqlbackup` creates a backup, it sends SQL commands to MySQL server. As part of the `mysqlbackup` invocation, specify the appropriate `--user`, `--password`, `--port`, and/or `--socket` options that are necessary to connect to the MySQL server. These connection options can be a part of [client] group in configuration files. `mysqlbackup` reads your default configuration files and then the `my.cnf` file specified on the command line. `mysqlbackup` reads only `--user`, `--password`, `--port`, and `--socket` options from the `[client]` group, and ignores any other options. If you do not provide a value for `--password`, the command prompts for one from the keyboard.

7.6 Example: Setting MySQL Privileges for mysqlbackup

The minimum privileges for the MySQL user that `mysqlbackup` connects are:

- `RELOAD` on all databases and tables.
- `CREATE`, `INSERT`, and `DROP` on the tables `mysql.ibbackup_binlog_marker`, `mysql.backup_progress`, and `mysql.backup_history`, and also `SELECT` on `mysql.backup_history`.
- `SUPER`, used to optimize locking and minimize disruption to database processing. This privilege is only needed to back up MySQL 5.5 and higher servers.
- `CREATE TEMPORARY TABLES` for the `mysql` database. This privilege is only needed to back up MySQL 5.5 and higher servers.
- `REPLICATION CLIENT`, to retrieve the binlog position, which is stored with the backup.

To set these privileges for a MySQL user (`dba` in this example) connecting from localhost, issue statements like the following from the `mysql` client program:

```
$ mysql -u root

mysql> GRANT RELOAD ON *.* TO 'dba'@'localhost';

mysql> GRANT CREATE, INSERT, DROP ON mysql.ibbackup_binlog_marker TO 'dba'@'localhost';

mysql> GRANT CREATE, INSERT, DROP ON mysql.backup_progress TO 'dba'@'localhost';

mysql> GRANT CREATE, INSERT, SELECT, DROP ON mysql.backup_history TO 'dba'@'localhost';

mysql> GRANT REPLICATION CLIENT ON *.* TO 'dba'@'localhost';
```

```
mysql> GRANT SUPER ON *.* TO 'dba'@'localhost';

mysql> GRANT CREATE TEMPORARY TABLES ON `mysql.*` TO 'dba'@'localhost';

mysql> FLUSH PRIVILEGES;
```

7.7 Example: Making an Incremental Backup of InnoDB and MyISAM tables

In this example, we use the `mysqlbackup` command to make an incremental backup of a database that includes both InnoDB tables and MyISAM tables. We specify `--lsn 2654255716` on the command line because the previous backup displayed this line near the end of the output:

```
ibbackup: Scanned log up to lsn 2654255716.
```

The `test/` subdirectory within the backup directory includes several `.frm` files:

```
$ mysqlbackup --incremental --lsn 2654255716 /home/pekka/.my.cnf /incr-backup
mysqlbackup: Starting mysqlbackup with following arguments:
mysqlbackup --incremental --lsn 2654255716 /home/pekka/.my.cnf /incr-backup
mysqlbackup: The unique backup id generated for the current backup operation is 12918212880165500

mysqlbackup: IMPORTANT: Please check that backup run completes successfully.
             At the end of a successful 'backup' run mysqlbackup
             prints "mysqlbackup completed OK!".

mysqlbackup: Created backup directory '/incr-backup/2010-12-08_17-14-48'
mysqlbackup: Using ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
mysqlbackup: Using MySQL client version: 5.1.47
mysqlbackup: Checking a connection to MySQL Server with parameters:
mysqlbackup: port=3308, socket=/home/pekka/mysql/MySQL.socket
mysqlbackup: Using MySQL server version: 5.1.37

101208 17:14:48 mysqlbackup: Starting ibbackup binary with args:

./ibbackup --suspend-at-end --incremental 2654255716 /home/pekka/.my.cnf /incr-backup/2010-12-08_17-14-48/back
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.my.cnf:
innodb_data_home_dir got value /sqldata/mts
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata/mts
innodb_log_group_home_dir got value /sqldata/mts
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /incr-backup/2010-12-08_17-14-48/backup-my.cnf:
innodb_data_home_dir got value /incr-backup/2010-12-08_17-14-48
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /incr-backup/2010-12-08_17-14-48
innodb_log_group_home_dir got value /incr-backup/2010-12-08_17-14-48
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

ibbackup: System tablespace file format is Barracuda.
ibbackup: Found checkpoint at lsn 2666733462.
ibbackup: Starting log scan from lsn 2666733056.
```

```
101208 17:14:48  ibbackup: Copying log...
mysqlbackup: Waiting for ibbackup process to suspend
mysqlbackup: Suspend file '/incr-backup/2010-12-08_17-14-48/ibbackup_suspended'
101208 17:14:48  ibbackup: Log copied, lsn 2666733462.
ibbackup: We wait 1 second before starting copying the data files...
101208 17:14:49  ibbackup: Copying /sqldata/mts/ibdata1 (Barracuda file format).
101208 17:14:49  ibbackup: Copying /sqldata/mts/ibdata2 (Barracuda file format).
101208 17:14:50  ibbackup: Copying /sqldata/mts/ibdata3 (Barracuda file format).
101208 17:14:53  ibbackup: Copying /sqldata/mts/test/alex1.ibd (Antelope file format).
101208 17:14:54  ibbackup: Copying /sqldata/mts/test/alex2.ibd (Antelope file format).
101208 17:14:54  ibbackup: Copying /sqldata/mts/test/alex3.ibd (Antelope file format).
101208 17:14:54  ibbackup: Copying /sqldata/mts/test/blobt3.ibd (Antelope file format).
101208 17:14:55  ibbackup: Copying /sqldata/mts/test/ibstest0.ibd (Antelope file format).
101208 17:14:56  ibbackup: Copying /sqldata/mts/test/ibtest09.ibd (Antelope file format).

ibbackup: You specified the option --suspend-at-end.

101208 17:14:56 mysqlbackup: Continuing after ibbackup has suspended

101208 17:14:56 mysqlbackup: Starting to lock all the tables....

101208 17:14:56 mysqlbackup: All tables are locked and flushed to disk
mysqlbackup: Opening backup source directory '/sqldata/mts'

101208 17:14:56 mysqlbackup: Starting to backup all files in subdirectories of '/sqldata/mts'
mysqlbackup: Backing up the database directory 'mysql'
mysqlbackup: Backing up the database directory 'test'
mysqlbackup: Resuming ibbackup

101208 17:14:56  ibbackup: Suspending the backup procedure to wait
ibbackup: until you delete the marker file /incr-backup/2010-12-08_17-14-48/ibbackup_suspended.
101208 17:14:57  ibbackup: Suspension ends. Continuing the backup procedure.

101208 17:14:57  ibbackup: Copying of the last data file is close to ending...
ibbackup: We still once copy the latest flushed log to ibbackup_logfile.
101208 17:14:57  ibbackup: Copying /sqldata/mts/mysql/ibbackup_binlog_marker.ibd (Antelope file format).
ibbackup: A copied database page was modified at 2666733462.
ibbackup: Scanned log up to lsn 2666736714.
ibbackup: Was able to parse the log up to lsn 2666736714.
ibbackup: Maximum page number for a log record 51
ibbackup: Backup contains changes from lsn 2654255717 to lsn 2666736714
101208 17:14:57  ibbackup: Incremental backup completed!

101208 17:14:57 mysqlbackup: All tables unlocked
mysqlbackup: All MySQL tables were locked for 1.023 seconds
mysqlbackup: Backup created in directory '/incr-backup/2010-12-08_17-14-48'
mysqlbackup: start_lsn: 2654255717
mysqlbackup: incremental_base_lsn: 2666733462
mysqlbackup: end_lsn: 2666736714

101208 17:14:58 mysqlbackup: mysqlbackup completed OK!
```

The incremental backup directory also contains these `.frm` files:

```
$ ls -l /full-backup/2010-12-08_17-14-11/{.,test}
/full-backup/2010-12-08_17-14-11/.:
total 106540
-rw-r--r-- 1 pekka pekka      359 2010-12-08 17:14 backup-my.cnf
-rw-r--r-- 1 pekka pekka       18 2010-12-08 17:14 ibbackup_binlog_info
-rw-r--r-- 1 pekka pekka      155 2010-12-08 17:14 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka     4608 2010-12-08 17:14 ibbackup_logfile
-rw-r----- 1 pekka pekka 10485760 2010-12-08 17:14 ibdata1
-rw-r----- 1 pekka pekka 20971520 2010-12-08 17:14 ibdata2
-rw-r----- 1 pekka pekka 77594624 2010-12-08 17:14 ibdata3
drwx------ 2 pekka pekka     4096 2010-12-08 17:14 mysql
```

```
drwxr-x--- 2 pekka pekka     4096 2010-12-08 17:14 test

/full-backup/2010-12-08_17-14-11/test:
total 74216
-rw-r----- 1 pekka pekka     9076 2010-12-08 17:14 alex1.frm
-rw-r----- 1 pekka pekka 24117248 2010-12-08 17:14 alex1.ibd
-rw-r----- 1 pekka pekka     9076 2010-12-08 17:14 alex2.frm
-rw-r----- 1 pekka pekka   245760 2010-12-08 17:14 alex2.ibd
-rw-r----- 1 pekka pekka     9076 2010-12-08 17:14 alex3.frm
-rw-r----- 1 pekka pekka   245760 2010-12-08 17:14 alex3.ibd
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:14 blobt3.frm
-rw-r----- 1 pekka pekka 24117248 2010-12-08 17:14 blobt3.ibd
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:14 ibstest0.frm
-rw-r----- 1 pekka pekka 26214400 2010-12-08 17:14 ibstest0.ibd
-rw-r----- 1 pekka pekka     8722 2010-12-08 17:14 ibtest09.frm
-rw-r----- 1 pekka pekka   933888 2010-12-08 17:14 ibtest09.ibd
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:14 ibtest11a.frm
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:14 ibtest11b.frm
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:14 ibtest11c.frm
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:14 ibtest11d.frm
```

Once again, we apply to the full backup any changes that occurred while the backup was running:

```
$ mysqlbackup --apply-log /home/pekka/.my.cnf /full-backup/2010-12-08_17-14-11
mysqlbackup: Starting mysqlbackup with following arguments:
mysqlbackup --apply-log /home/pekka/.my.cnf /full-backup/2010-12-08_17-14-11
mysqlbackup: IMPORTANT: Please check that apply-log run completes successfully.
            At the end of a successful 'apply-log' run mysqlbackup
            prints "mysqlbackup completed OK!".

mysqlbackup: Using ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
mysqlbackup: Starting ibbackup binary with args:

./ibbackup --apply-log /full-backup/2010-12-08_17-14-11/backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /full-backup/2010-12-08_17-14-11/backup-my.cnf:
innodb_data_home_dir got value /full-backup/2010-12-08_17-14-11
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /full-backup/2010-12-08_17-14-11
innodb_log_group_home_dir got value /full-backup/2010-12-08_17-14-11
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

101208 17:15:08  ibbackup: ibbackup_logfile's creation parameters:
ibbackup: start lsn 2654252032, end lsn 2654255716,
ibbackup: start checkpoint 2654252464.
InnoDB: Doing recovery: scanned up to log sequence number 2654255716
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percents: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Setting log file size to 0 10485760
Setting log file size to 0 10485760
ibbackup: We were able to parse ibbackup_logfile up to
ibbackup: lsn 2654255716.
ibbackup: Last MySQL binlog file position 0 538, file name ./MySQL-bin.000046
ibbackup: The first data file is '/full-backup/2010-12-08_17-14-11/ibdata1'
ibbackup: and the new created log files are at '/full-backup/2010-12-08_17-14-11/'
ibbackup: System tablespace file format is Barracuda.
101208 17:15:10  ibbackup: Full backup prepared for recovery successfully!

101208 17:15:10 mysqlbackup: mysqlbackup completed OK!
```

Then, we apply the changes from the incremental backup:

```
$ mysqlbackup --apply-log --incremental /incr-backup/2010-12-08_17-14-48/backup-my.cnf /full-backup/2010-12
mysqlbackup: Starting mysqlbackup with following arguments:
mysqlbackup --apply-log --incremental /incr-backup/2010-12-08_17-14-48/backup-my.cnf /full-backup/2010-12-0
mysqlbackup: IMPORTANT: Please check that apply-log run completes successfully.
            At the end of a successful 'apply-log' run mysqlbackup
            prints "mysqlbackup completed OK!".

mysqlbackup: Using ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2

101208 17:15:10 mysqlbackup: Copying all non-InnoDB files in subdirectories of
mysqlbackup: '/incr-backup/2010-12-08_17-14-48'
mysqlbackup: to the corresponding subdirectories in
mysqlbackup: '/full-backup/2010-12-08_17-14-11'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/backup_history.CSM'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/backup_history.CSV'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/backup_history.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/backup_progress.CSM'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/backup_progress.CSV'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/backup_progress.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/columns_priv.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/columns_priv.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/columns_priv.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/db.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/db.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/db.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/event.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/event.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/event.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/func.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/func.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/func.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/general_log.CSM'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/general_log.CSV'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/general_log.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_category.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_category.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_category.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_keyword.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_keyword.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_keyword.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_relation.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_relation.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_relation.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_topic.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_topic.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/help_topic.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/host.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/host.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/host.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/ibbackup_binlog_marker.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/ndb_binlog_index.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/ndb_binlog_index.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/ndb_binlog_index.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/plugin.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/plugin.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/plugin.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/proc.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/proc.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/proc.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/procs_priv.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/procs_priv.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/procs_priv.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/servers.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/servers.MYI'
```

```
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/servers.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/slow_log.CSM'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/slow_log.CSV'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/slow_log.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/tables_priv.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/tables_priv.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/tables_priv.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_leap_second.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_leap_second.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_leap_second.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_name.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_name.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_name.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_transition.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_transition.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_transition.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_transition_type.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_transition_type.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/time_zone_transition_type.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/user.MYD'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/user.MYI'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/mysql/user.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/alex1.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/alex2.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/alex3.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/blobt3.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/ibstest0.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/ibtest09.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/ibtest11a.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/ibtest11b.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/ibtest11c.frm'
mysqlbackup: Copying file '/incr-backup/2010-12-08_17-14-48/test/ibtest11d.frm'
mysqlbackup: Checking for deleted databases and non-InnoDB files in them

101208 17:15:10 mysqlbackup: Finished copying all non-InnoDB files from the
mysqlbackup: incremental backup to the full backup.

mysqlbackup: Starting ibbackup binary with args:

./ibbackup --apply-log --incremental /incr-backup/2010-12-08_17-14-48/backup-my.cnf /full-backup/2010-12-08_17
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /incr-backup/2010-12-08_17-14-48/backup-my.cnf:
innodb_data_home_dir got value /incr-backup/2010-12-08_17-14-48
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /incr-backup/2010-12-08_17-14-48
innodb_log_group_home_dir got value /incr-backup/2010-12-08_17-14-48
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /full-backup/2010-12-08_17-14-11/backup-my.cnf:
innodb_data_home_dir got value /full-backup/2010-12-08_17-14-11
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /full-backup/2010-12-08_17-14-11
innodb_log_group_home_dir got value /full-backup/2010-12-08_17-14-11
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

101208 17:15:10  ibbackup: ibbackup_logfile's creation parameters:
ibbackup: start lsn 2666733056, end lsn 2666736714,
```

```
ibbackup: start checkpoint 2666733462.
InnoDB: Doing recovery: scanned up to log sequence number 2666736714
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percents: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Setting log file size to 0 10485760
Setting log file size to 0 10485760
ibbackup: We were able to parse ibbackup_logfile up to
ibbackup: lsn 2666736714.
ibbackup: Last MySQL binlog file position 0 538, file name ./MySQL-bin.000046
ibbackup: The first data file is '/full-backup/2010-12-08_17-14-11/ibdata1'
ibbackup: and the new created log files are at '/full-backup/2010-12-08_17-14-11/'
ibbackup: System tablespace file format is Barracuda.
101208 17:15:12  ibbackup: Incremental backup applied successfully!
mysqlbackup: Copying file 'ibbackup_export_variables.txt'
mysqlbackup: Copying file 'ibbackup_binlog_info'

101208 17:15:12 mysqlbackup: mysqlbackup completed OK!
```

Now, the data files in the full backup directory are fully up-to-date, as of the time of the last incremental backup:

```
$ ls -l /full-backup/2010-12-08_17-14-11/{.,test}
/full-backup/2010-12-08_17-14-11/.:
total 137260
-rw-r--r-- 1 pekka pekka      359 2010-12-08 17:14 backup-my.cnf
-rw-r--r-- 1 pekka pekka       18 2010-12-08 17:15 ibbackup_binlog_info
-rw-r--r-- 1 pekka pekka      155 2010-12-08 17:15 ibbackup_export_variables.txt
-rw-r----- 1 pekka pekka     4608 2010-12-08 17:15 ibbackup_logfile
-rw-r----- 1 pekka pekka 10485760 2010-12-08 17:15 ibdata1
-rw-r----- 1 pekka pekka 20971520 2010-12-08 17:15 ibdata2
-rw-r----- 1 pekka pekka 77594624 2010-12-08 17:14 ibdata3
-rw-r----- 1 pekka pekka 10485760 2010-12-08 17:15 ib_logfile0
-rw-r----- 1 pekka pekka 10485760 2010-12-08 17:15 ib_logfile1
-rw-r----- 1 pekka pekka 10485760 2010-12-08 17:15 ib_logfile2
drwx------ 2 pekka pekka     4096 2010-12-08 17:14 mysql
drwxr-x--- 2 pekka pekka     4096 2010-12-08 17:14 test

/full-backup/2010-12-08_17-14-11/test:
total 74216
-rw-r----- 1 pekka pekka     9076 2010-12-08 17:15 alex1.frm
-rw-r----- 1 pekka pekka 24117248 2010-12-08 17:14 alex1.ibd
-rw-r----- 1 pekka pekka     9076 2010-12-08 17:15 alex2.frm
-rw-r----- 1 pekka pekka   245760 2010-12-08 17:14 alex2.ibd
-rw-r----- 1 pekka pekka     9076 2010-12-08 17:15 alex3.frm
-rw-r----- 1 pekka pekka   245760 2010-12-08 17:14 alex3.ibd
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:15 blobt3.frm
-rw-r----- 1 pekka pekka 24117248 2010-12-08 17:15 blobt3.ibd
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:15 ibstest0.frm
-rw-r----- 1 pekka pekka 26214400 2010-12-08 17:14 ibstest0.ibd
-rw-r----- 1 pekka pekka     8722 2010-12-08 17:15 ibtest09.frm
-rw-r----- 1 pekka pekka   933888 2010-12-08 17:14 ibtest09.ibd
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:15 ibtest11a.frm
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:15 ibtest11b.frm
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:15 ibtest11c.frm
-rw-r----- 1 pekka pekka     8626 2010-12-08 17:15 ibtest11d.frm
```

Chapter 8 Making a Partial Backup

Table of Contents

When the multiple tablespaces feature is enabled, it is possible to make a partial backup of the InnoDB tables. The multiple tablespaces feature allows storing each InnoDB table in a separate tablespace holding the data and indexes of one table only. A single-table tablespace consists of one autoextending datafile named `table_name.ibd` in the database directory of the table. Those InnoDB tables for which the multiple tablespaces feature is not enabled, are stored as usual in the system tablespace defined by the `innodb_data_file_path` and `innodb_data_home_dir` parameters in the `my.cnf` file.

With its `--include` option, `ibbackup` (v2.0 or higher) can be instructed to make a partial backup excluding some of the per-table datafiles. A partial backup contains the system tablespace and per-table datafiles of those tables that match the include pattern.

For each table with a per-table data file a string of the form `db_name.table_name` is checked against the regular expression specified with the `--include` option. If the regular expression matches the complete string `db_name.table_name`, the table is included in the backup. The regular expression uses the POSIX extended form. On Unix-like systems, quote the regular expression appropriately to prevent interpretation of shell meta-characters. This feature has been implemented with Henry Spencer's regular expression library.

IMPORTANT: Although the `mysqlbackup` command supports taking partial backups, be careful when restoring a database from a partial backup. `mysqlbackup` copies also the `.frm` files of those tables that are not included in the backup. If you use `mysqlbackup` with `--include` option, before restoring the database, delete from the backup data the `.frm` files for any tables that are not included in the backup.

IMPORTANT: If *no* tables match the regular expression pattern specified with the `--include` option, the backup currently includes *all* the file-per-table tables. This behavior might change; do not rely on it as part of your backup procedure.

8.1 Example: Making an Uncompressed Partial Backup

In this example, we have configured MySQL so that some InnoDB tables have their own tablespaces. We make a partial backup including only those InnoDB tables in `test` database whose name starts with `ib`. The contents of the database directory for `test` database are shown below. The directory contains a MySQL description file (`.frm` file) for each of the tables (`alex1`, `alex2`, `alex3`, `blobt3`, `ibstest0`, `ibstest09`, `ibtest11a`, `ibtest11b`, `ibtest11c`, and `ibtest11d`) in the database. Of these 10 tables six (`alex1`, `alex2`, `alex3`, `blobt3`, `ibstest0`, `ibstest09`) are stored in per-table datafiles (`.ibd` files).

```
$ ls /sqldata/mts/test
alex1.frm  alex2.ibd  blobt3.frm    ibstest0.ibd  ibtest11a.frm  ibtest11d.frm
alex1.ibd  alex3.frm  blobt3.ibd    ibtest09.frm  ibtest11b.frm
alex2.frm  alex3.ibd  ibstest0.frm  ibtest09.ibd  ibtest11c.frm
```

We run ibbackup with the `--include` option.

```
$ ibbackup --include 'test\.ib.*' /home/pekka/.my.cnf /home/pekka/.backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.my.cnf:
innodb_data_home_dir got value /sqldata/mts
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata/mts
innodb_log_group_home_dir got value /sqldata/mts
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /home/pekka/.backup-my.cnf:
innodb_data_home_dir got value /sqldata-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata-backup
innodb_log_group_home_dir got value /sqldata-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Value of include option: 'test\.ib.*'

ibbackup: System tablespace file format is Barracuda.
ibbackup: Found checkpoint at lsn 2666737471.
ibbackup: Starting log scan from lsn 2666737152.
101208 17:17:34  ibbackup: Copying log...
101208 17:17:34  ibbackup: Log copied, lsn 2666737471.
ibbackup: We wait 1 second before starting copying the data files...
101208 17:17:35  ibbackup: Copying /sqldata/mts/ibdata1 (Barracuda file format).
101208 17:17:36  ibbackup: Copying /sqldata/mts/ibdata2 (Barracuda file format).
101208 17:17:37  ibbackup: Copying /sqldata/mts/ibdata3 (Barracuda file format).
101208 17:17:43  ibbackup: Copying /sqldata/mts/test/ibstest0.ibd (Antelope file format).
101208 17:17:45  ibbackup: Copying /sqldata/mts/test/ibtest09.ibd (Antelope file format).
ibbackup: A copied database page was modified at 2666737471.
ibbackup: Scanned log up to lsn 2666737471.
ibbackup: Was able to parse the log up to lsn 2666737471.
ibbackup: Maximum page number for a log record 0
101208 17:17:45  ibbackup: Full backup completed!
```

The backup directory contains only backups of `ibstest` and `ibtest09` tables. Other InnoDB tables did not match the include pattern `test\.ib.*`. Notice, however, that the tables `ibtest11a`, `ibtest11b`, `ibtest11c`, `ibtest11d` are in the backup even though they are not visible in the directory shown below, because they are stored in the system tablespace (`ibdata1` file) which is always included in the backup.

```
$ ls /sqldata-backup/test
ibstest0.ibd   ibtest09.ibd
```

8.2 Example: Making a Compressed Partial Backup

We have configured MySQL so that every InnoDB table has its own tablespace. We make a partial backup including only those InnoDB tables whose name starts with `alex` or `blob`. The contents of the database directory for `test` database is shown below.

```
$ ls /sqldata/mts/test
alex1.frm  alex2.ibd  blobt3.frm    ibstest0.ibd  ibtest11a.frm  ibtest11d.frm
alex1.ibd  alex3.frm  blobt3.ibd    ibtest09.frm  ibtest11b.frm
```

```
alex2.frm  alex3.ibd  ibstest0.frm  ibtest09.ibd  ibtest11c.frm
```

We run ibbackup with the `--compress` and `--include` options:

```
$ ibbackup --compress --include '.*\.(alex|blob).*' /home/pekka/.my.cnf /home/pekka/.backup-my.cnf
ibbackup version 3.5.2 MySQL Enterprise Backup 3.5.2
Copyright (c) 2002, 2010, Oracle and/or its affiliates.
Run 'ibbackup --help' for help and 'ibbackup --version' for version info.

Note: Uses posix_fadvise() for performance optimization.

Contents of /home/pekka/.my.cnf:
innodb_data_home_dir got value /sqldata/mts
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata/mts
innodb_log_group_home_dir got value /sqldata/mts
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Contents of /home/pekka/.backup-my.cnf:
innodb_data_home_dir got value /sqldata-backup
innodb_data_file_path got value ibdata1:10M;ibdata2:20M;ibdata3:50M:autoextend
datadir got value /sqldata-backup
innodb_log_group_home_dir got value /sqldata-backup
innodb_log_files_in_group got value 3
innodb_log_file_size got value 10485760

Value of include option: '.*\.(alex|blob).*'

ibbackup: System tablespace file format is Barracuda.
ibbackup: Found checkpoint at lsn 2666737471.
ibbackup: Starting log scan from lsn 2666737152.
101208 17:17:57  ibbackup: Copying log...
101208 17:17:57  ibbackup: Log copied, lsn 2666737471.
ibbackup: We wait 1 second before starting copying the data files...
101208 17:17:58  ibbackup: Copying /sqldata/mts/ibdata1 (Barracuda file format).
101208 17:17:59  ibbackup: Copying /sqldata/mts/ibdata2 (Barracuda file format).
101208 17:17:59  ibbackup: Copying /sqldata/mts/ibdata3 (Barracuda file format).
101208 17:18:01  ibbackup: Copying /sqldata/mts/test/alex1.ibd (Antelope file format).
101208 17:18:03  ibbackup: Copying /sqldata/mts/test/alex2.ibd (Antelope file format).
101208 17:18:03  ibbackup: Copying /sqldata/mts/test/alex3.ibd (Antelope file format).
101208 17:18:03  ibbackup: Copying /sqldata/mts/test/blobt3.ibd (Antelope file format).
ibbackup: A copied database page was modified at 2666737471.
ibbackup: Scanned log up to lsn 2666737471.
ibbackup: Was able to parse the log up to lsn 2666737471.
ibbackup: Maximum page number for a log record 0

ibbackup: Compressed 147 MB of data files to 15 MB (compression 89%).

101208 17:18:04  ibbackup: Full backup completed!
```

The backup directory for the database `test` is shown below. The `.ibz` files are compressed per-table datafiles.

```
$ ls /sqldata-backup/test
alex1.ibz   alex2.ibz   alex3.ibz   blobt3.ibz
```

8.3 Restoring a Single .ibd File

A table with a table-specific tablespace (stored in an `.ibd` file) can be restored individually without taking down the MySQL server. If you have a clean backup of an `.ibd` file, you can restore it to the MySQL installation from which it originated as follows:

1. Prevent write operations for the table to be restored. This prevents users from modifying the table while the restore is in progress.

   ```
   LOCK TABLES tbl_name WRITE;
   ```

2. Issue this `ALTER TABLE` statement:

   ```
   ALTER TABLE tbl_name DISCARD TABLESPACE;
   ```

   Caution: This deletes the current `.ibd` file.

3. Put the backup `.ibd` file back in the proper database directory.

4. Issue this `ALTER TABLE` statement:

   ```
   ALTER TABLE tbl_name IMPORT TABLESPACE;
   ```

5. Restore is now complete and the write lock can be released:

   ```
   UNLOCK TABLES;
   ```

In this context, a clean`.ibd` file backup means:

- There are no uncommitted modifications by transactions in the `.ibd` file.
- There are no unmerged insert buffer entries in the `.ibd` file.
- Purge has removed all delete-marked index records from the `.ibd` file.
- `mysqld` has flushed all modified pages of the `.ibd` file from the buffer pool to the file.

You can make such a clean backup `.ibd` file with the following method:

1. Stop all activity from the `mysqld` server and commit all transactions.
2. Wait until `SHOW INNODB STATUS` shows that there are no active transactions in the database, and the main thread status of `InnoDB` is `Waiting for server activity`. Then you can make a copy of the `.ibd` file.

Another method for making a clean copy of an `.ibd` file is to use `ibbackup`:

1. Use `ibbackup` to back up the `InnoDB` installation.
2. Run `ibbackup --apply-log` to create a consistent version of the backup database.
3. Start a second (dummy) `mysqld` server on the backup and let it clean up the `.ibd` files in the backup. Wait for the cleanup to end.
4. Shut down the dummy mysqld server.
5. Take a clean `.ibd` file from the backup.

8.4 Backing Up Selected Databases

The `--databases` option of the `mysqlbackup` command lets you back up non-InnoDB tables only from selected databases, rather than across the entire MySQL instance. (To filter InnoDB tables, use the `--`

`include` option.) You can specify a space-separated list of database names, with the entire list enclosed in double quotation marks, or the absolute path (starting with a `/`) of a file containing the list of names, one per line.

Some or all of the database names can be qualified with table names, to only back up selected tables from those databases.

If you specify this option, make sure to include the same set of databases for every backup (especially incremental backups), so that you do not restore out-of-date versions of any databases.

8.5 Backing Up Files from Different Storage Engines

By default, all the files in the data directory are included in the backup, so any non-database files in that directory are backed up.

The `--only-known-file-types` option of the `mysqlbackup` command limits the backup to only those files that represent known data files from MySQL or the storage engines, such as `.frm`, `.myi`, `.mrg`, and so on. (See the full list of extensions [27].) By default, the `mysqlbackup` command backs up all file extensions within the data directory, which could include files produced by many different storage engines. Use this option if the additional data files from other storage engines should not be included in the backup, for performance or space reasons.

8.6 Backing Up In-Memory Database Data

The `--exec-when-locked` option of the `mysqlbackup` command lets you specify a command and arguments to run near the end of the backup, while the database is still locked. This command can copy or create additional files in the backup directory. For example, you can use this option to back up `MEMORY` tables with the `mysqldump` command, storing the output in the backup directory. To delay any redirection or variable substitution until the command is executed, enclose the entire parameter value within single quotes.

Chapter 9 Troubleshooting for MySQL Enterprise Backup

Table of Contents

To troubleshoot issues regarding backup and restore with the MySQL Enterprise Backup product, consider the following aspects:

- If the `mysqlbackup` command encounters problems during operating system calls, it returns the corresponding OS error codes. You might need to consult your operating system documentation for the meaning and solution of these error codes. (The `ibbackup` command only returns an error code of `1` regardless of the error, but it does display the OS error code in its error output.)
- Incremental backups require care to specify a sequence of time periods You must record the final LSN value at the end of each backup, and specify that value in the next incremental backup. You must also make sure that the full backup you restore is prepared correctly first, so that it contains all the changes from the sequence of incremental backups.
- As the `mysqlbackup` command proceeds, it writes progress information into the `mysql.backup_progress` table. When the command finishes the backup operation, it records status information in the `mysql.backup_history` table. You can query these tables to monitor ongoing jobs, see how much time was needed for various stages, and check if any errors occurred.
-

9.1 Error codes of MySQL Enterprise Backup

The return code of the MySQL Enterprise Backup (`ibbackup`) process is 0 if the backup or restore run succeeds. If the run fails for any reason, the return code is 1.

If `ibbackup` fails, because an operating system call fails, `ibbackup` usually displays the operating systems error code along with a detailed error message.

On Linux and other Unix-like systems, the operating system error codes are POSIX error codes. Those POSIX error codes that are possible with `ibbackup` are shown in Table 9.1, “OS Errors for Linux and other Unix-Like Systems”. A complete list of all POSIX errors is available in the file `/usr/include/errno.h` on your system.

Table 9.1 OS Errors for Linux and other Unix-Like Systems

Error code	Value	Description
EPERM	1	Operation not permitted
ENOENT	2	No such file or directory
ESRCH	3	No such process
EINTR	4	Interrupted system call
EIO	5	I/O error
ENXIO	6	No such device or address

Error code	Value	Description
EBADF	9	Bad file number
EAGAIN	11	Try again
ENOMEM	12	Out of memory
EACCES	13	Permission denied
EBUSY	16	Device or resource busy
EEXIST	17	File exists
ENODEV	19	No such device
ENOTDIR	20	Not a directory
EMFILE	24	Too many open files
EFBIG	27	File too large
ENOSPC	28	No space left on device
EROFS	30	Read-only file system
ENAMETOOLONG	36	File name too long
ENODATA	61	No data available
ETIME	62	Timer expired
EBADFD	77	File descriptor in bad state
EDQUOT	122	Quota exceeded

On Microsoft Windows, `ibbackup` uses Win32 API calls. The Windows System Error codes possible with `ibbackup` are listed in Table 9.2, “OS Errors for Windows Systems”. A complete list of all Windows System errors is available at http://msdn2.microsoft.com/en-us/library/ms681381(VS.85).aspx.

Table 9.2 OS Errors for Windows Systems

Error code	Value	Description
ERROR_SUCCESS	0	The operation completed successfully.
ERROR_FILE_NOT_FOUND	2	The system cannot find the file specified.
ERROR_PATH_NOT_FOUND	3	The system cannot find the path specified.
ERROR_TOO_MANY_OPEN_FILES	4	The system cannot open the file.
ERROR_ACCESS_DENIED	5	Access is denied.
ERROR_NOT_ENOUGH_MEMORY	8	Not enough storage is available to process this command.
ERROR_OUTOFMEMORY	14	Not enough storage is available to complete this operation.
ERROR_INVALID_DRIVE	15	The system cannot find the drive specified.
ERROR_WRITE_PROTECT	19	The media is write protected.
ERROR_BAD_UNIT	20	The system cannot find the device specified.
ERROR_NOT_READY	21	The device is not ready.

Error code	Value	Description
ERROR_SEEK	25	The drive cannot locate a specific area or track on the disk.
ERROR_WRITE_FAULT	29	The system cannot write to the specified device.
ERROR_READ_FAULT	30	The system cannot read from the specified device.
ERROR_GEN_FAILURE	31	A device attached to the system is not functioning.
ERROR_HANDLE_DISK_FULL	39	The disk is full.
ERROR_BAD_NETPATH	53	The network path was not found.
ERROR_DEV_NOT_EXIST	55	The specified network resource or device is no longer available.
ERROR_FILE_EXISTS	80	The file exists.

9.2 Working Around Corruption Problems

Sometimes the operating system or the hardware can corrupt a data file page, in a location that does not cause a database error, but prevents `ibbackup` from completing:

```
ibbackup: Re-reading page at offset 0 3185082368 in /sqldata/mts/ibdata15
ibbackup: Re-reading page at offset 0 3185082368 in /sqldata/mts/ibdata15
ibbackup: Error: page at offset 0 3185082368 in /sqldata/mts/ibdata15 seems corrupt!
```

Scrambled data in memory can produce this error, even though the data on disk is correct. Reboot the database server and storage device to see if the problem persists.

If the data really is corrupt on disk, you can restore from an earlier backup and “roll forward” the recent changes to bring the database back to its current state.

If you want to make an additional backup before investigating the cause of the corruption, you can compile and run a troubleshooting utility, `innodb_page_checksum_reset.c`, to reset the LSN and checksum fields in one data page, so that `ibbackup` can complete the backup.

Download `innodb_page_checksum_reset.c`.

The sample program resets page 22357 in a datafile `ibdata1`. Edit these values according to the values in your error message.

To compile on Linux:

```
$ gcc -o ibreset innodb_page_checksum_reset.c
```

If your data file is larger than 2 GB, compile with large file support:

```
$ gcc -D_XOPEN_SOURCE=600 D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -o ibreset innodb_page_checksum_reset.
```

The command produces an executable file called `ibreset`.

IMPORTANT: Do not treat corruption problems as a minor annoyance. Find out what is wrong with the OS or the hardware that causes corrupt pages to appear. (Such troubleshooting is beyond the scope of this manual.)

9.3 Using the MySQL Enterprise Backup Backup Logs

The `mysql.backup_progress` table lets you monitor backup jobs as they run. The `mysql.backup_history` table lets you see the results of completed jobs. Because these tables are created with the CSV storage engine, you can query them from SQL, or parse the text files from an application or script.

`backup_progress` Table

Each row in the `backup_progress` table records a state change or message from a running backup job. The `backup_progress` table has the following columns:

- `backup_id`
- `tool_name`
- `error_code`
- `error_message`
- `current_time`
- `current_state`

Because the CSV storage engine cannot represent `NULL` values directly, the logs use a -1 value instead, for example in the `binlog_pos` column if binary logging is not enabled.

Use the `backup_id` value to group the information for a single backup operation, and to correlate with the corresponding row in the `backup_history` table after the job is finished.

Use the `error_code` and `error_message` values to track the progress of the job, and detect if a serious error occurs that requires stopping the backup operation.

Use the `current_time` and `current_state` values to measure how long each part of the backup operation takes, to help with planning the time intervals for future backups.

`backup_history` Table

Each row in the `backup_history` table records the details of one completed backup job, produced by the `mysqlbackup` command. The `backup_history` table has the following columns:

- `backup_id`
- `tool_name`
- `start_time`
- `end_time`
- `binlog_pos`
- `binlog_file`
- `compression_level`
- `engines`

- innodb_data_file_path
- innodb_file_format
- start_lsn
- end_lsn
- incremental_base_lsn
- backup_type
- backup_format
- mysql_data_dir
- innodb_data_home_dir
- innodb_log_group_home_dir
- innodb_log_files_in_group
- innodb_log_file_size
- backup_destination
- lock_time
- exit_state
- last_error
- last_error_code

Use the end_lsn value to automate operations related to incremental backups. When you take a full or incremental backup, you specify the end LSN from that backup as the starting LSN for the next incremental backup.

Use the values that correspond to backup-related configuration settings, such as mysql_data_dir, innodb_data_home_dir, and backup_destination, to confirm that the backups are using the right source and destination directories.

Use the values exit_state, last_error, and last_error_code to evaluate the success or failure of each backup.

If last_error is 'NO_ERROR', the backup operation was successful. In case of any errors, you can retrieve the full list of errors for that backup operation from the backup_progress table.

Appendix A Known Bugs and Limitations

Table of Contents

Please refer to the MySQL Enterprise Backup version history in Appendix B, *MySQL Enterprise Backup Change History* for a list of fixed `ibbackup` and `mysqlbackup` bugs.

A.1 Limitations of `mysqlbackup` and `ibbackup` Commands

- In some cases, backups of non-transactional tables such as `MyISAM` tables could contain additional uncommitted data. If autocommit is turned off, and both `InnoDB` tables and non-transactional tables are modified within the same transaction, data can be written to the non-transactional table before the binlog position is updated. The binlog position is updated when the transaction is committed, but the non-transactional data is written immediately. If the backup occurs while such a transaction is open, the backup data contains the updates made to the non-transactional table.

- If the `mysqlbackup` process is interrupted, such as by a Unix `kill -9` command, a `FLUSH TABLES WITH READ LOCK` operation might remain running. In this case, use the `KILL QUERY` command from the `mysql` command line to kill the `FLUSH TABLES WITH READ LOCK` statement. This issue is more likely to occur if the `FLUSH TABLES` operation is stalled by a long-running query or transaction. Refer to Chapter 7, *`mysqlbackup` Command Reference* for guidelines about backup timing and performance.

- Do not run the DDL operations `ALTER TABLE`, `TRUNCATE TABLE`, `OPTIMIZE TABLE`, `REPAIR TABLE`, or `RESTORE TABLE` while a backup operation is going on. The resulting backup might be corrupted.

  The only `ALTER TABLE` operations that can be safely run in parallel with a backup are those that do not influence the physical representation of records on disk, such as changing column names or default column values.

A.2 Linux-2.4.18 kernel/driver Bugs

A MySQL Enterprise Backup user reported that in a 2-way Dell computer with a Red Hat kernel 2.4.18, concurrent running of `mysqld` and `ibbackup` could cause `mysqld` to crash. Crashes did not happen in a non-Dell computer. An upgrade to a Linux stock kernel 2.4.20 fixed the problem.

A.3 Known `ibbackup` and `mysqlbackup` Bugs

If you take a backup when there are `TEMPORARY` tables in the database, and you use those temporary tables to update or insert into normal tables, then applying the MySQL binlog to a backup can fail. That is, you may not be able to roll forward the backup using the MySQL binlog. The reason is that `TEMPORARY` tables are not copied to the backup. And, actually we cannot copy them to the backup, because the names of temporary table files `#sql*.frm` do not correspond to the logical table names that MySQL writes to the binlog. This problem might be removed in the future, if MySQL implements “row-level binlogging”.

`mysqlbackup` cannot back up `HEAP`, or `BDB` type tables.

Currently, mysqlbackup requires that innodb_data_file_path **contains only plain files, not paths**. That is, specifications like innodb_data_file_path=/dir1/ibdata1:100M will not work.

Currently, if the regular expression for the --include option does not match any table names, *all* file-per-table tables are included in the backup.

A.4 MySQL Bugs Affecting mysqlbackup

A.5 Compatibility with Older MySQL/InnoDB Versions

From time to time changes are made to the format of data and log files of MySQL/InnoDB. These changes may make older MySQL Enterprise Backup versions incompatible with the new MySQL/InnoDB version.

Currently, these are the major MySQL/InnoDB versions: 3.23 (first released in May 12, 2001), 4.0 (December 23, 2001), 4.1 (April 3, 2003), 5.0 (December 24, 2003), 5.1 (November 29, 2005), and 5.5 (December 15, 2010).

MySQL Enterprise Backup 3.5 is compatible with MySQL/InnoDB version 5.0 and up.

IMPORTANT: Backing up tables using the Barracuda file format, which is available with the combination of MySQL and the InnoDB Plugin, requires MySQL Enterprise Backup 3.5 or newer.

For MySQL versions prior to 5.0, the corresponding backup product is the InnoDB Hot Backup product, which is the ancestor of MySQL Enterprise Backup. InnoDB Hot Backup continues to be compatible with MySQL 5.0, 5.1, and 5.5, with the exception of InnoDB tables in the Barracuda format. For compatibility information, see the InnoDB Hot Backup documentation.

Appendix B MySQL Enterprise Backup Change History

Table of Contents

This appendix lists the changes to the MySQL Enterprise Backup, beginning with the most recent release. Each release section covers added or changed functionality, bug fixes, and known issues, if applicable. All bug fixes are referenced by bug number and include a link to the bug database. Bugs are listed in order of resolution. To find a bug quickly, search by bug number.

B.1 Changes in MySQL Enterprise Backup 3.5.4 (2011-04-21)

This section documents changes and bug fixes that have been applied in MySQL Enterprise Backup, version 3.5.4.

Bugs Fixed

- Minor fixes for copyright notices.

B.2 Changes in MySQL Enterprise Backup 3.5.2 (2010-12-16)

This section documents changes and bug fixes that have been applied in MySQL Enterprise Backup, version 3.5.2.

Functionality Added or Changed

- A call to `posix_fadvise()` can be used to reduce the flush cycle of the operating system cache and improve backup performance. This option is set on by default.
- The combined InnoDB and MyISAM backup functionality of the `innobackup` command is now available on Windows systems. The former Perl script is rewritten in C/C++ as the `mysqlbackup` command. This release continues to include the `innobackup` command, which may be deprecated by the next release. There are also some changes to the syntax as specified in the manual.
- Backup history and progress information is logged to the `mysql.backup_history` and `mysql.backup_progress` tables, so that it can be used by the MySQL Enterprise Monitor product and other tools to easily monitor backup operations. For the details of the backup history table, see Chapter 9, *Troubleshooting for MySQL Enterprise Backup*.

B.3 Changes in MySQL Enterprise Backup 3.5.1 (2010-11-01)

This section documents changes and bug fixes that have been applied in MySQL Enterprise Backup, version 3.5.1.

Functionality Added or Changed

- Incremental backup.
- Support for the Barracuda file format of InnoDB. MySQL Enterprise Backup can now backup tables that use recent InnoDB features such as table compression and the `dynamic` row format.

Appendix C Licenses for Third-Party Components

Table of Contents

Oracle acknowledges that certain Third Party and Open Source software has been used to develop or is incorporated in the MySQL Enterprise Backup product. This appendix includes required third-party license information.

C.1 RegEX-Spencer Library License

The following software may be included in this product:

```
Henry Spencer's Regular-Expression Library (RegEX-Spencer)

Copyright 1992, 1993, 1994, 1997 Henry Spencer. All rights reserved.
This software is not subject to any license of the American Telephone
and Telegraph Company or of the Regents of the University of
California.

Permission is granted to anyone to use this software for any purpose
on any computer system, and to alter it and redistribute it, subject
to the following restrictions:

1. The author is not responsible for the consequences of use of this
software, no matter how awful, even if they arise from flaws in it.

2. The origin of this software must not be misrepresented, either by
explicit claim or by omission. Since few users ever read sources,
credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must not be
misrepresented as being the original software.  Since few users ever
read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.
```

C.2 zlib License

The following software may be included in this product:

zlib

Oracle gratefully acknowledges the contributions of Jean-loup Gailly and Mark Adler in creating the zlib general purpose compression library which is used in this product.

```
zlib.h -- interface of the 'zlib' general purpose compression library
Copyright (C) 1995-2004 Jean-loup Gailly and Mark Adler
```

```
zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.3, July 18th, 2005
Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.5, April 19th, 2010
Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the
use of this software. Permission is granted to anyone to use this software
for any purpose,including commercial applications, and to alter it and
redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would
   be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not
   be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu
```

C.3 Percona Multiple I/O Threads Patch License

The following software may be included in this product:

Percona Multiple I/O threads patch

```
Copyright (c) 2008, 2009 Percona Inc
All rights reserved.

Redistribution and use of this software in source and binary forms,
with or without modification, are permitted provided that the
following conditions are met:

 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of Percona Inc. nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission of Percona Inc.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

C.4 Google SMP Patch License

The following software may be included in this product:

Google SMP Patch

```
Google SMP patch

Copyright (c) 2008, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Google Inc. nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

C.5 Google Controlling Master Thread I/O Rate Patch License

The following software may be included in this product:

Google Controlling master thread I/O rate patch

```
Copyright (c) 2009, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Google Inc. nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
```

```
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

C.6 RFC 3174 - US Secure Hash Algorithm 1 (SHA1) License

The following software may be included in this product:

RFC 3174 - US Secure Hash Algorithm 1 (SHA1)

```
RFC 3174 - US Secure Hash Algorithm 1 (SHA1)

Copyright (C) The Internet Society (2001).  All Rights Reserved.

This document and translations of it may be copied and furnished to
others, and derivative works that comment on or otherwise explain it
or assist in its implementation may be prepared, copied, published
and distributed, in whole or in part, without restriction of any
kind, provided that the above copyright notice and this paragraph are
included on all such copies and derivative works.  However, this
document itself may not be modified in any way, such as by removing
the copyright notice or references to the Internet Society or other
Internet organizations, except as needed for the purpose of
developing Internet standards in which case the procedures for
copyrights defined in the Internet Standards process must be
followed, or as required to translate it into languages other than
English.

The limited permissions granted above are perpetual and will not be
revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an
"AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING
TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING
BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement
Funding for the RFC Editor function is currently provided by the
Internet Society.
```

MySQL Enterprise Backup Glossary

These terms are commonly used in information about the MySQL Enterprise Backup product.

A

.ARM file
: Metadata for ARCHIVE tables. Contrast with **.ARZ file**. Files with this extension are always included in backups produced by the `mysqlbackup` command of the **MySQL Enterprise Backup** product.
See Also .ARZ file, MySQL Enterprise Backup.

.ARZ file
: Data for ARCHIVE tables. Contrast with **.ARM file**. Files with this extension are always included in backups produced by the `mysqlbackup` command of the **MySQL Enterprise Backup** product.
See Also .ARM file, MySQL Enterprise Backup.

Antelope
: The code name for the original InnoDB **file format**. It supports the **redundant** and **compact** row formats, but not the newer **dynamic** and **compressed** row formats available in the **Barracuda** file format.

  If your application could benefit from InnoDB table **compression**, or uses BLOBs or large text columns that could benefit from the dynamic row format, you might switch some tables to Barracuda format. You select the file format to use by setting the `innodb_file_format` option before creating the table.
  See Also Barracuda, compression, file format.

apply
: The operation that transforms a **raw backup** into a **prepared backup** by incorporating changes that occurred while the backup was running, using data from the **log**.
See Also log, prepared backup, raw backup.

B

backup
: The process of copying some or all table data and metadata from a MySQL instance, for safekeeping. Can also refer to the set of copied files. This is a crucial task for DBAs. The reverse of this process is the **restore** operation.

  With MySQL, **physical backups** are performed by the **MySQL Enterprise Backup** product, and **logical backups** are performed by the `mysqldump` command. These techniques have different characteristics in terms of size and representation of the backup data, and speed (especially speed of the restore operation).

  Backups are further classified as **hot**, **warm**, or **cold** depending on how much they interfere with normal database operation. (Hot backups have the least interference, cold backups the most.)
  See Also cold backup, hot backup, logical backup, MySQL Enterprise Backup, mysqldump, physical backup, warm backup.

backup repository
: Contrast with **server repository**.
See Also repository, server repository.

backup-my.cnf
: A small **configuration file** generated by **MySQL Enterprise Backup**, containing a minimal set of configuration parameters. This file records the settings that apply to this backup data. Subsequent operations, such as the **apply** process, read options from this file to determine how the backup data is structured. This file always has the extension `.cnf`, rather than `.cnf` on Unix-like systems and `.ini` on Windows systems.
See Also apply, configuration file.

Barracuda
: The code name for an InnoDB **file format** that supports compression for table data. This file format was first introduced in the InnoDB Plugin. It supports the **compressed** row format that enables InnoDB table compression, and the **dynamic** row format that improves the storage layout for BLOB and large text columns. You can select it through the `innodb_file_format` option.

  Because the InnoDB **system tablespace** is stored in the original **Antelope** file format, to use the Barracuda file format you must also enable the **file-per-table** setting, which puts newly created tables in their own tablespaces separate from the system tablespace.

  The **MySQL Enterprise Backup** product version 3.5 and above supports backing up tablespaces that use the Barracuda file format.
  See Also Antelope, file format, MySQL Enterprise Backup, row format, system tablespace.

binary log
: A file containing a record of all statements that attempt to change table data. These statements can be replayed to bring slave servers up to date in a **replication** scenario, or to bring a database up to date after restoring table data from a backup. The binary logging feature can be turned on and off, although Oracle recommends always enabling it if you use replication or perform backups.

  You can examine the contents of the binary log, or replay those statements during replication or recovery, by using the `mysqlbinlog` command. For full information about the binary log, see The Binary Log. For MySQL configuration options related to the binary log, see Binary Log Options and Variables.

  For the **MySQL Enterprise Backup** product, the file name of the binary log and the current position within the file are important details. To record this information for the master server when taking a backup in a replication context, you can specify the `--slave-info` option.

  Prior to MySQL 5.0, a similar capability was available, known as the update log. In MySQL 5.0 and higher, the binary log replaces the update log.
  See Also binlog, MySQL Enterprise Backup, replication.

binlog
: An informal name for the **binary log** file. For example, you might see this abbreviation used in e-mail messages or forum discussions.
  See Also binary log.

C

cold backup
: A **backup** taken while the database is shut down. For busy applications and web sites, this might not be practical, and you might prefer a **warm backup** or a **hot backup**.
  See Also backup, connection, hot backup, warm backup.

compression
: A technique that produces smaller **backup** files, with size reduction influenced by the **compression level** setting. Suitable for keeping multiple sets of non-critical backup files. (For recent backups of critical data, you might leave the data uncompressed, to allow fast restore speed in case of emergency.)

  MySQL Enterprise Backup can apply compression to the contents of **InnoDB** tables during the backup process, turning the **.ibd** files into **.ibz** files.

  Compression adds CPU overhead to the backup process, and requires additional time and disk space during the **restore** process.
  See Also backup, compression level, .ibd file, .ibz file, InnoDB, MySQL Enterprise Backup, restore.

compression level
: A setting that determines how much **compression** to apply to a compressed backup. This setting ranges from 0 (none), 1 (default level when compression is enabled) to 9 (maximum). The amount of compression for a given compression level depends on the nature of your data values. Higher compression levels do impose additional CPU overhead, so ideally you use the lowest value that produces a good balance of compression with low CPU overhead.
See Also compression.

configuration file
: The file that holds the startup options of the MySQL server and related products and components. Often referred to by its default file name, **my.cnf** on Linux, Unix, and OS X systems, and **my.ini** on Windows systems. The **MySQL Enterprise Backup** stores its default configuration settings in this file, under a `[mysqlbackup]` section. For convenience, MySQL Enterprise Backup can also read settings from the `[client]` section, for configuration options that are common between MySQL Enterprise Backup and other programs that connect to the MySQL server.
See Also my.cnf, my.ini, MySQL Enterprise Backup.

connection
: The mechanism used by certain backup operations to communicate with a running MySQL **server**. For example, the `mysqlbackup` command can log into the server being backed up to insert and update data in the **progress table** and the **history table**. A **hot backup** typically uses a database connection for convenience, but can proceed anyway if the connection is not available. A **warm backup** always uses a database connection, because it must put the server into a read-only state. A **cold backup** is taken while the MySQL server is shut down, and so cannot use any features that require a connection.
See Also cold backup, history table, hot backup, progress table, server, warm backup.

crash recovery
: The cleanup activities for InnoDB tables that occur when MySQL is started again after a crash. Changes that were committed before the crash, but not yet written to the tablespace files, are reconstructed from the **doublewrite buffer**. When the database is shut down normally, this type of activity is performed during shutdown by the **purge** operation.

D

data dictionary
: A set of tables, controlled by the InnoDB storage engine, that keeps track of InnoDB-related objects such as tables, indexes, and table columns. These tables are part of the InnoDB **system tablespace**.

  Because the **MySQL Enterprise Backup** product always backs up the system tablespace, all backups include the contents of the data dictionary.
  See Also hot backup, MySQL Enterprise Backup, system tablespace.

database
: A set of tables and related objects owned by a MySQL user. Equivalent to “schema” in Oracle Database terminology. **MySQL Enterprise Backup** can perform a **partial backup** that includes some databases and not others. The full set of databases controlled by a MySQL server is known as an **instance**.
See Also instance, MySQL Enterprise Backup, partial backup.

downtime
: A period when the database is unresponsive. The database might be entirely shut down, or in a read-only state when applications are attempting to insert, update, or delete data. The goal for your backup strategy is to minimize downtime, using techniques such as **hot backup** for **InnoDB** tables, **cold backup** using **slave** servers in a **replication** configuration, and minimizing the duration of the **suspend** stage where you run customized backup logic while the MySQL server is **locked**.
See Also cold backup, hot backup, InnoDB, locking, replication, slave, suspend.

E

exclude
: In a **partial backup**, to select a set of tables, databases, or a combination of both to be omitted from the backup. Contrast with **include**.
See Also partial backup.

extract
: The operation that retrieves some content from an **image** file produced by a **single-file backup**. It can apply to a single file (unpacked to an arbitrary location) or to the entire backup (reproducing the original directory structure of the backup data). These two kinds of extraction are performed by the `mysqlbackup` options `extract` and `image-to-backup-dir`, respectively.
See Also image, single-file backup.

F

.frm file
: A file containing the metadata, such as the table definition, of a MySQL table.

For backups, you must always keep the full set of `.frm` files along with the backup data to be able to restore tables that are altered or dropped after the backup.

Although each InnoDB table has a `.frm` file, InnoDB maintains its own table metadata in the system tablespace; the `.frm` files are not needed for InnoDB to operate on InnoDB tables.

These files are backed up by the **MySQL Enterprise Backup** product. These files must not be modified by an `ALTER TABLE` operation while the backup is taking place, which is why backups that include non-InnoDB tables perform a `FLUSH TABLES WITH READ LOCK` operation to freeze such activity while backing up the `.frm` files. Restoring a backup can result in `.frm` files being created, changed, or removed to match the state of the database at the time of the backup.
See Also MySQL Enterprise Backup.

file format
: The format used by InnoDB for its data files named `ibdata1`, `ibdata2`, and so on. Each file format supports one or more row formats.
See Also Antelope, Barracuda, ibdata file, row format.

full backup
: A **backup** that includes all the **tables** in each MySQL database, and all the databases in a MySQL instance. Contrast with **partial backup** and **incremental backup**. Full backups take the longest, but also require the least amount of followup work and administration complexity. Thus, even when you primarily do partial or incremental backups, you might periodically do a full backup.
See Also backup, incremental backup, partial backup, table.

H

history table
: The table `mysql.backup_history` that holds details of completed **backup** operations. While a backup job is running, the details (especially the changing status value) are recorded in the **progress table**.
See Also backup, progress table.

hot backup
: A backup taken while the MySQL **instance** and is running and applications are reading and writing to it. Contrast with **warm backup** and **cold backup**.

A hot backup involves more than simply copying data files: it must include any data that was inserted or updated while the backup was in process; it must exclude any data that was deleted while the backup was in process; and it must ignore any changes started by **transactions** but not committed.

The Oracle product that performs hot backups, of **InnoDB** tables especially but also tables from MyISAM and other storage engines, is **MySQL Enterprise Backup**.

The hot backup process consists of two stages. The initial copying of the InnoDB data files produces a **raw backup**. The **apply** step incorporates any changes to the database that happened while the backup was running. Applying the changes produces a **prepared** backup; these files are ready to be restored whenever necessary.

A **full backup** consists of a hot backup phase that copies the InnoDB data, followed by a **warm backup** phase that copies any non-InnoDB data such as MyISAM tables and **.frm** files.
See Also apply, cold backup, .frm file, full backup, InnoDB, instance, MySQL Enterprise Backup, prepared backup, raw backup, warm backup.

I

.ibd file
Each InnoDB **tablespace** created using the **file-per-table** setting has a filename with a `.ibd` extension. This extension does not apply to the **system tablespace**, which is made up of files named `ibdata1`, `ibdata2`, and so on.
See Also .ibz file, system tablespace, tablespace.

.ibz file
When the **MySQL Enterprise Backup** product performs a **compressed backup**, it transforms each **tablespace** file that is created using the **file-per-table** setting from a **`.ibd`** extension to a **`.ibz`** extension.

The compression applied during backup is distinct from the **compressed row format** that keeps table data compressed during normal operation. An InnoDB tablespace that is already in compressed row format is not compressed a second time, because that would save little or no space.
See Also .ibd file, .ibz file, MySQL Enterprise Backup, tablespace.

ibdata file
A set of files with names such as `ibdata1`, `ibdata2`, and so on, that make up the InnoDB **system tablespace**. These files contain metadata about InnoDB tables, and can contain some or all of the table and index data also (depending on whether the **file-per-table option** is in effect when each table is created). For backward compatibility these files always use the **Antelope** file format.
See Also Antelope, system tablespace.

image
The file produced as part of a **single-file backup** operation. It can be a real file that you store locally, or standard output (specified as `-`) when the backup data is **streamed** directly to another command or remote server. This term is referenced in several `mysqlbackup` options such as `backup-dir-to-image` and `image-to-backup-dir`.
See Also single-file backup, streaming.

include
In a **partial backup**, to select a set of tables, databases, or a combination of both to be backed up. Contrast with **exclude**.
See Also partial backup.

incremental backup
A backup that captures only data changed since the previous backup. It has the potential to be smaller and faster than a **full backup**. The incremental backup data must be merged with the contents of the previous backup before it can be restored. See Section 7.7, “Example: Making an Incremental Backup of InnoDB and MyISAM

tables" for usage details. Related `mysqlbackup` options are `--incremental`, `--incremental-with-redo-log-only`, `--incremental-backup-dir`, `--incremental-base`, and `--start-lsn`.
See Also full backup.

InnoDB
: The type of MySQL **table** that works best with **MySQL Enterprise Backup**. These tables can be backed up using the **hot backup** technique that avoids interruptions in database processing. For this reason, and because of the higher reliability and concurrency possible with InnoDB tables, most deployments should use InnoDB for the bulk of their data and their most important data. In MySQL 5.5 and higher, the `CREATE TABLE` statement creates InnoDB tables by default.
See Also hot backup, MySQL Enterprise Backup, table.

instance
: The full contents of a MySQL server, possibly including multiple **databases**. A **backup** operation can back up an entire instance, or a **partial backup** can include selected databases and tables.
See Also database, partial backup.

L

locking

See Also suspend, warm backup.

log
: Several types of log files are used within the MySQL Enterprise Backup product. The most common is the InnoDB **redo log** that is consulted during **incremental backups**.
See Also incremental backup, redo log.

log sequence number
: See LSN.

logical backup
: A **backup** that reproduces table structure and data, without copying the actual data files. For example, the `mysqldump` command produces a logical backup, because its output contains statements such as `CREATE TABLE` and `INSERT` that can re-create the data. Contrast with **physical backup**.
See Also backup, physical backup.

LSN
: Acronym for **log sequence number**. This arbitrary, ever-increasing value represents a point in time corresponding to operations recorded in the **redo log**. (This point in time is regardless of transaction boundaries; it can fall in the middle of one or more transactions.) It is used internally by InnoDB during **crash recovery** and for managing the buffer pool.

  In the **MySQL Enterprise Backup** product, you can specify an LSN to represent the point in time from which to take an **incremental backup**. The relevant LSN is displayed by the output of the `mysqlbackup` command. Once you have the LSN corresponding to the time of a full backup, you can specify that value to take a subsequent incremental backup, whose output contains another LSN for the next incremental backup.
  See Also crash recovery, hot backup, incremental backup, redo log.

M

.MRG file
: A file containing references to other tables, used by the `MERGE` storage engine. Files with this extension are always included in backups produced by the `mysqlbackup` command of the **MySQL Enterprise Backup** product.

See Also MySQL Enterprise Backup.

.MYD file
A file that MySQL uses to store data for a MyISAM table.
See Also .MYI file, MySQL Enterprise Backup.

.MYI file
A file that MySQL uses to store indexes for a MyISAM table.
See Also .MYD file, MySQL Enterprise Backup.

manifest
The record of the environment (for example, command-line arguments) and data files involved in a backup, stored in the files `meta/backup_create.xml` and `meta/backup_content.xml`, respectively. This data can be used by management tools during diagnosis and troubleshooting procedures.

master
In a **replication** configuration, a database server that sends updates to a set of **slave** servers. It typically dedicates most of its resources to write operations, leaving user queries to the slaves. With **MySQL Enterprise Backup**, typically you perform backups on the slave servers rather than the master, to minimize any slowdown of the overall system.
See Also MySQL Enterprise Backup, replication, slave.

media management software
A class of software programs for managing backup media, such as libraries of tape backups. One example is **Oracle Secure Backup**. Abbreviated **MMS**.
See Also Oracle Secure Backup.

my.cnf
The typical name for the MySQL **configuration file** on Linux, Unix, and OS X systems.
See Also configuration file, my.ini.

my.ini
The typical name for the MySQL **configuration file** on Windows systems.
See Also configuration file, my.cnf.

MyISAM
A MySQL storage engine, formerly the default for new tables. In MySQL 5.5 and higher, **InnoDB** becomes the default storage engine. MySQL Enterprise Backup can back up both types of tables, and tables from other storage engines also. The backup process for InnoDB tables (**hot backup**) is less disruptive to database operations than for MyISAM tables (**warm backup**).
See Also hot backup, InnoDB, MySQL Enterprise Backup, warm backup.

MySQL Enterprise Backup
A licensed products that performs **hot backups** of MySQL databases. It offers the most efficiency and flexibility when backing up **InnoDB** tables; it can also back up MyISAM and other kinds of tables. It is included as part of the MySQL Enterprise Edition subscription.
See Also Barracuda, hot backup, InnoDB.

mysqlbackup
The primary command of the **MySQL Enterprise Backup** product. Different options perform **backup** and **restore** operations.
See Also backup, MySQL Enterprise Backup, restore.

mysqldump
A MySQL command that performs **logical backups**, producing a set of SQL commands to recreate tables and data. Suitable for smaller backups or less critical data, because the **restore** operation takes longer than with a **physical backup** produced by **MySQL Enterprise Backup**.

See Also logical backup, MySQL Enterprise Backup, physical backup, restore.

O

.opt file
: A file containing database configuration information. Files with this extension are always included in backups produced by the backup operations of the **MySQL Enterprise Backup** product.
See Also MySQL Enterprise Backup.

offline
: A type of operation performed while the database server is stopped. With the **MySQL Enterprise Backup** product, the main offline operation is the **restore** step. You can optionally perform a **cold backup**, which is another offline operation. Contrast with **online**.
See Also cold backup, MySQL Enterprise Backup, online, restore.

online
: A type of operation performed while the database server is running. A **hot backup** is the ideal example, because the database continues to run and no read or write operations are blocked. For that reason, sometimes “hot backup” and “online backup” are used as synonyms. A **cold backup** is the opposite of an online operation; by definition, the database server is shut down while the backup happens. A **warm backup** is also a kind of online operation, because the database server continues to run, although some write operations could be blocked while a warm backup is in progress. Contrast with **offline**.
See Also cold backup, hot backup, offline, warm backup.

Oracle Secure Backup
: An Oracle product for managing **backup** media, and so classified as **media management software** (**MMS**). Abbreviated **OSB**. For **MySQL Enterprise Backup**, OSB is typically used to manage tape backups.
See Also backup, media management software, MySQL Enterprise Backup, OSB.

OSB
: Abbreviation for **Oracle Secure Backup**, a **media management software** product (**MMS**).
See Also Oracle Secure Backup.

P

.par file
: A file containing partition definitions. Files with this extension are always included in backups produced by the `mysqlbackup` command of the **MySQL Enterprise Backup** product.
See Also MySQL Enterprise Backup.

parallel backup
: The default processing mode in MySQL Enterprise Backup 3.8 and higher, employing multiple threads for different classes of internal operations (read, process, and write). See Section 1.3, “Performance and Space Considerations” for an overview, Performance / Scalability / Capacity Options for the relevant `mysqlbackup` options, and Performance Considerations for MySQL Enterprise Backup for performance guidelines and tips.

partial backup
: A **backup** that contains some of the **tables** in a MySQL database, or some of the databases in a MySQL instance. Contrast with **full backup**.
See Also backup, full backup, partial restore, table.

partial restore
: A **restore** operation that applies to one or more **tables** or **databases**, but not the entire contents of a MySQL server. The data being restored could come from either a **partial backup** or a **full backup**.
See Also database, full backup, partial backup, restore, table.

physical backup
A **backup** that copies the actual data files. For example, the **MySQL Enterprise Backup** command produces a physical backup, because its output contains data files that can be used directly by the `mysqld` server. Contrast with **logical backup**.
See Also backup, logical backup, MySQL Enterprise Backup.

point in time
The time corresponding to the end of a **backup** operation. A **prepared backup** includes all the changes that occurred while the backup operation was running. **Restoring** the backup brings the data back to the state at the moment when the backup operation completed.
See Also backup, prepared backup, restore.

prepared backup
The set of backup data that is entirely consistent and ready to be restored. It is produced by performing the **apply** operation on the **raw backup**.
See Also apply, raw backup.

progress table
The table `mysql.backup_progress` that holds details of running **backup** operations. When a backup job finishes, the details are recorded in the **history table**.
See Also backup, history table.

R

raw backup
The initial set of backup data, not yet ready to be restored because it does not incorporate changes that occurred while the backup was running. The **apply** operation transforms the backup files into a **prepared backup** that is ready to be restored.
See Also apply, prepared backup.

redo log
A set of files, typically named `ib_logfile0` and `ib_logfile1`, that record statements that attempt to change data in InnoDB tables. These statements are replayed automatically to correct data written by incomplete transactions, on startup following a crash. The passage of data through the redo logs is represented by the ever-increasing **LSN** value. The 4GB limit on maximum size for the redo log is raised in MySQL 5.6.
See Also LSN.

regular expression
Some MySQL Enterprise Backup features use POSIX-style regular expressions, for example to specify tables, databases, or both to **include** or **exclude** from a **partial backup**. Regular expressions require escaping for dots in filenames, because the dot is the single-character wildcard; no escaping is needed for forward slashes in path names. When specifying regular expressions on the command line, surround them with quotation marks as appropriate for the shell environment, to prevent expansion of characters such as asterisks by the shell wildcard mechanism.
See Also exclude, include, partial backup.

replication
A common configuration for MySQL deployments, with data and DML operations from a **master** server synchronized with a set of **slave** servers. With MySQL **Enterprise Backup,** you might take a backup on one server, and restore on a different system to create a new slave server with the data already in place. You might also back up data from a slave server rather than the master, to minimize any slowdown of the overall system.
See Also master, MySQL Enterprise Backup, slave.

repository
We distinguish between the **server repository** and the **backup repository**.

See Also backup repository, server repository.

restore
The converse of the **backup** operation. The data files from a **prepared backup** are put back into place to repair a data issue or bring the system back to an earlier state.
See Also backup, prepared backup.

row format
The disk storage format for a row from an InnoDB table. As InnoDB gains new capabilities such as compression, new row formats are introduced to support the resulting improvements in storage efficiency and performance.

Each table has its own row format, specified through the `ROW_FORMAT` option. To see the row format for each InnoDB table, issue the command `SHOW TABLE STATUS`. Because all the tables in the system tablespace share the same row format, to take advantage of other row formats typically requires setting the `innodb_file_per_table` option, so that each table is stored in a separate tablespace.

S

SBT
Acronym for **system backup to tape**.
See Also system backup to tape.

server
A MySQL **instance** controlled by a `mysqld` daemon. A physical machine can host multiple MySQL servers, each requiring its own **backup** operations and schedule. Some backup operations communicate with the server through a **connection**.
See Also connection, instance.

server repository
Contrast with **backup repository**.
See Also backup repository, repository.

single-file backup
A backup technique that packs all the backup data into one file (the backup **image**), for ease of storage and transfer. The **streaming** backup technique requires using a single-file backup.
See Also image, streaming.

slave
In a **replication** configuration, a database server that receives updates from a **master** server. Typically used to service user queries, to minimize the query load on the master. With **MySQL Enterprise Backup**, you might take a backup on one server, and restore on a different system to create a new slave server with the data already in place. You might also back up data from a slave server rather than the master, to minimize any slowdown of the overall system.
See Also master, replication.

streaming
A backup technique that transfers the data immediately to another server, rather than saving a local copy. Uses mechanisms such as Unix pipes. Requires a **single-file backup**, with the destination file specified as `-` (standard output).
See Also single-file backup.

suspend
An optional stage within the backup where the MySQL Enterprise Backup processing stops, to allow for user-specific operations to be run. The `mysqlbackup` command has options that let you specify commands to be run while the backup is suspended. Most often used in conjunction with backups of **InnoDB** tables only, where you might do your own scripting for handling **.frm files**.

See Also .frm file, InnoDB.

system backup to tape
: An API for **media management software**. Abbreviated **SBT**. Several `mysqlbackup` options (with **sbt** in their names) pass information to **media management software** products such as **Oracle Secure Backup**.
See Also Oracle Secure Backup, SBT.

system tablespace
: By default, this single data file stores all the table data for a database, as well as all the metadata for InnoDB-related objects (the **data dictionary**).

  Turning on the **innodb_file_per_table** option causes each newly created table to be stored in its own **tablespace**, reducing the size of, and dependencies on, the system tablespace.

  Keeping all table data in the system tablespace has implications for the **MySQL Enterprise Backup** product (backing up one large file rather than several smaller files), and prevents you from using certain InnoDB features that require the newer **Barracuda** file format. on the
  See Also Barracuda, data dictionary, file format, ibdata file, tablespace.

T

.TRG file
: A file containing **trigger** parameters. Files with this extension are always included in backups produced by the `mysqlbackup` command of the **MySQL Enterprise Backup** product.
See Also MySQL Enterprise Backup.

table
: Although a table is a distinct, addressable object in the context of SQL, for **backup** purposes we are often concerned with whether the table is part of the **system tablespace**, or was created under the **file-per-table** setting and so resides in its own **tablespace**.
See Also backup, system tablespace, tablespace.

tablespace
: For **InnoDB** tables, the file that holds the data and indexes for a table. Can be either the **system tablespace** containing multiple tables, or a table created with the **file-per-table** setting that resides in its own tablespace file.
See Also InnoDB, system tablespace.

W

warm backup
: A **backup** taken while the database is running, but that restricts some database operations during the backup process. For example, tables might become read-only. For busy applications and web sites, you might prefer a **hot backup**.
See Also backup, cold backup, hot backup.

Index

Symbols

A

B

C

D

E

F

H

I

L

M

O

P

R

S

T

U

W