You have just inherited a production MySQL system and there is no confirmation that an existing MySQL backup strategy is in operation. What is the least you need to do? Before undertaking any backup strategy, you should consider some necessary prerequisites about your database size and storage engine usage that have a direct effect on your system availability during any backup approach.

In this chapter we will discuss the approach necessary to identify a minimum functionality backup including:

- Determining your database size
- Determining your storage engine usage
- Locking and downtime implications

Approaching a MySQL Backup

There is more than one strategy to backup a MySQL environment. These strategies also depend on the number of servers in the MySQL topology. There are a number of various open source and commercial tools available to perform backups. In Chapter 2 we will be discussing all these possible options in detail.

Suppose that you have an environment with a single server and you want to create a *consistent* backup. You have two immediate options at your disposal. The first option is to stop your MySQL instance and take a full file-system cold backup. This would result in your system being unavailable for an undetermined time, and you would need to ensure that you make a copy of all the right information including MySQL data, transaction and binary logs if applicable, and the current MySQL configuration.

Your second option is to use a client tool included with the standard MySQL installation. The `mysqldump` command can produce a consistent MySQL backup without stopping the MySQL instance. However, before running `mysqldump`, you must consider several important factors in order to make an informed decision about the best options to use. These factors are:

- What is the size of the backup?
- What locking strategy is necessary?
- How long will the backup take?

Determining Your Database Size

An important consideration for performing a MySQL backup is the size of your backup when backing up to local disk. This is required to ensure that you have available disk space to store your backup file.

The following SQL statement provides the total size in MB of your current data and indexes.

```
mysql> SELECT ROUND(SUM(data_length+index_length)/1024/1024)
    ->        AS total_mb,
    ->        ROUND(SUM(data_length)/1024/1024) AS data_mb,
    ->        ROUND(SUM(index_length)/1024/1024) AS index_mb
    -> FROM   INFORMATION_SCHEMA.tables;
+----------+---------+----------+
| total_mb | data_mb | index_mb |
+----------+---------+----------+
|      927 |     847 |       80 |
+----------+---------+----------+
```

Your mysqldump backup will be approximately the same size as your data with an appropriate safety margin of 10 to 15 percent. There is no precise calculation; however, your backup produces a text-based output of your data. It is possible to compress your backup concurrently or to transfer to a different network device. These options and their limitations are discussed in Chapter 2.

From this SQL statement the database data size is 847M. For later reference, the size of the backup file as described in the section running mysqldump reports a size of 818M using the common default options.

Choosing a Locking Strategy

The locking strategy chosen will determine whether your application can perform write operations with your database during the execution of a backup. By default, mysqldump performs a table level lock to ensure a consistent version of all data using the LOCK TABLES command. This occurs with the `--lock-tables` command line option, which is not enabled by default. This option is part of the `--opt` option, which is enabled by default. You can elect to not lock tables; however, this may not ensure a consistent backup. When you are using the MyISAM storage engine, `--lock-tables` is necessary to ensure a consistent backup.

Alternatively, mysqldump provides the `--single-transaction` option, which creates a consistent version snapshot of all tables in a single transaction. This option is only applicable when using a storage engine that supports multiversioning (MVCC). InnoDB is the only storage engine included in a default MySQL installation that is applicable. When specified, this option automatically turns off `--lock-tables`.

The following SQL statement will confirm the storage engines in use for your MySQL instance.

```
mysql> SELECT  table_schema, engine, COUNT(*) AS tables
    -> FROM    information_schema.tables
    -> WHERE    table_schema NOT IN
    ->         ('INFORMATION_SCHEMA','PERFORMANCE_SCHEMA')
    -> GROUP BY table_schema, engine
    -> ORDER BY 3 DESC;
+--------------------+--------+--------+
| table_schema       | engine | tables |
+--------------------+--------+--------+
| shopping_cart      | MyISAM |    109 |
| cust_db            | InnoDB |     48 |
| mysql              | MyISAM |     21 |
| analytics          | InnoDB |     20 |
| phpmyadmin         | MyISAM |      8 |
| newsletter         | MyISAM |      8 |
| cust_db            | MyISAM |      3 |
| mysql              | CSV    |      2 |
+--------------------+--------+--------+
```

In this example, the MySQL instance has several different schemas that support various functions including a shopping cart and administration tool. An InnoDB only application may look like this:

```
+--------------------+--------+--------+
| table_schema       | engine | tables |
+--------------------+--------+--------+
| prod_db            | InnoDB |    122 |
| mysql              | MyISAM |     21 |
| mysql              | CSV    |      2 |
+--------------------+--------+--------+
```

As you see in the example, the `mysql` meta schema uses MyISAM. It is not possible to change this. The two CSV tables in the `mysql` schema are used for SQL logging and the data is not necessary for any backup strategy. If your database is all InnoDB, you will have two options regarding the MyISAM `mysql` tables, which we will discuss later in this chapter.

Execution Time

The most important requirement is to determine how long your backup will take. There is no calculation that can give an accurate answer. The size of your database, the amount of system RAM, the storage engine(s) in use, the MySQL configuration, the hard drive speed, and current workload all contribute to the calculation. What is important when performing a backup is to gather this information for future reference.

The execution time is important, as this is an effective maintenance window for your database. During a database backup there may be a limitation of application functionality, there may be a performance overhead in performing the backup, and your backup may limit other operations, including batch processing or software maintenance.

Combining Information

The following is a recommended SQL statement that combines all information for an audit of your database size.

```
$ cat storage_engines.sql
SELECT  table_schema, engine,
       ROUND(SUM(data_length+index_length)/1024/1024) AS total_mb,
       ROUND(SUM(data_length)/1024/1024) AS data_mb,
       ROUND(SUM(index_length)/1024/1024) AS index_mb,
       COUNT(*) AS tables
FROM  information_schema.tables
GROUP BY table_schema, engine
ORDER BY 3 DESC;

mysql> source storage_engines.sql
+--------------------+--------+----------+---------+----------+--------+
| table_schema       | engine | total_mb | data_mb | index_mb | tables |
+--------------------+--------+----------+---------+----------+--------+
| analytics          | InnoDB |    10903 |   10525 |      378 |     20 |
| cust_db            | InnoDB |     1155 |     962 |      194 |     48 |
| newsletter         | InnoDB |      514 |     278 |      237 |      7 |
| shopping_cart      | MyISAM |       27 |      19 |        8 |    109 |
| cust_db            | MyISAM |        9 |       3 |        7 |      3 |
| mysql              | MyISAM |        1 |       0 |        0 |     21 |
| information_schema | MyISAM |        0 |       0 |        0 |      8 |
| information_schema | MEMORY |        0 |       0 |        0 |     20 |
| mysql              | CSV    |        0 |       0 |        0 |      2 |
+--------------------+--------+----------+---------+----------+--------+
```

Performing a MySQL Backup

Now that you have gathered the prerequisite information, you have the details necessary to make an informed decision.

The choice of how to perform a backup, when to perform it, and how you monitor and verify is a more complex process, which is discussed in more detail starting with Chapter 2.

One additional consideration during a backup process is to disable any cron or batch processes during the backup to minimize additional workload. This can minimize database contention and shorten the window of time needed.

Running mysqldump

In the simplest form, you can perform a backup using `mysqldump` with the following syntax.

```
$ time mysqldump -uroot -p[password] --all-databases > backup.sql
$ echo $?
$ ls -lh backup.sql
```

- The first command runs the `mysqldump` for all databases, producing an ASCII dump in the `backup.sql` file.
- The second command confirms the exit status of the first command. A non-zero result is an indication of a problem during the backup process. If any errors occur, these are generally shown in the screen output.
- The third command shows the size of your backup file for later reference.

For example:

```
$ time mysqldump -uroot -p[password] --all-databases > backup.sql
real    0m35.493s
user    0m9.808s
sys     0m3.021s
$ echo $?
0
$ ls -lh backup.sql
-rw-rw-r-- 1 usr grp 818M Aug 10 21:37 backup.sql
```

This is a successful backup file totaling 818M that took 35 seconds to execute. The original size of the database data as shown previously for this MySQL instance was 847M.

TIP

Prefixing the `mysqldump` command with the `time` command will provide valuable information on the actual time taken. Recording your backup time and size is an important administration step that all DBAs should do. This time information is useful for scheduling other system requirements, for an additional verification step if a successful backup has a significantly different time, and it is helpful in benchmarking with using different arguments, MySQL configuration settings, or changes in physical hardware.

An example of an error condition may look like this:

```
$ time mysqldump -uroot -p[password] --all-databases > backup.sql
mysqldump: Got error: 1142: SELECT,LOCK TABL command denied to user
'root'@'localhost' for table 'cond_instances' when using LOCK TABLES
real    0m7.692s
user    0m1.780s
sys     0m0.313s
$ echo $?
2
$ ls -lh backup.sql
-rw-rw-r-- 1 usr grp 94M Aug 10 21:28 backup.sql
```

A backup file as per this example may in isolation appear to be completely *valid*. That is, this file contains valid and *complete* SQL statements, and can be successfully used to restore data in one or more schemas; however, it is incomplete as a full backup of all data. The execution time, error status, and size are all important information for verification of a successful backup.

Creating a backup is only the first step in a suitable strategy that should support both a static and point in time recovery. It is important that this backup file can be used successfully in recovery. This is discussed in Chapter 5.

Securing Your Backup

The final step in a minimal backup approach is to ensure the security of your data. The backup is currently on the same system as your data. A loss of this system would include the data and your backup. The minimum you should undertake is to copy your backup to a secondary location: For example:

```
$ time gzip backup.sql
$ scp backup.sql.gz  another-server:backup-dir
```

Benefits with mysqldump

The `mysqldump` command provides a SQL-based backup file. This can be ideal for creating a backup that can be executed on different versions of MySQL, and on different operating systems. For example, you can view this file directly and see SQL statements:

```
$ more backup.sql

--
-- Current Database: `mysql`
--
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `mysql` /*!40100 DEFAULT CHAR-
ACTER SET latin1 */;
```

```
USE `mysql`;
--
-- Table structure for table `help_topic`
--
DROP TABLE IF EXISTS `help_topic`;
/*!40101 SET @saved_cs_client  = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `help_topic` (
  `help_topic_id` int(10) unsigned NOT NULL,
  `name` char(64) NOT NULL,
  `help_category_id` smallint(5) unsigned NOT NULL,
  `description` text NOT NULL,
  `example` text NOT NULL,
  `url` char(128) NOT NULL,
  PRIMARY KEY (`help_topic_id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='help topics';
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Dumping data for table `help_topic`
--
LOCK TABLES `help_topic` WRITE;/*!40000 ALTER TABLE `help_topic` DIS-
ABLE KEYS */;
INSERT INTO `help_topic` VALUES (0,'MIN',16,'Syntax:\nMIN([DISTINCT]
expr)\n\n
Returns the minimum value of expr. MIN() may take a string argument;
in\nsuch cases,
it returns the minimum string value. See\nhttp://dev.mysql.com/doc/
refman/5.1/
en/mysql-indexes.html. The DISTINCT\nkeyword can be used to find the
minimum of ...
```

More Information

For more information about the various options with `mysqldump`, you can obtain a list of valid options with the following syntax:

```
$ mysqldump --help
```

You can find detailed information in the MySQL reference manual at http://dev.mysql.com/doc/refman/5.5/en/mysqldump.html.

Other Options

If your database uses all InnoDB tables, the default locking strategy is restrictive. You have to consider the impact of the `mysql` schema MyISAM tables. Under normal circumstances, you can generally ignore the consistency requirement, providing you do not perform operations that change the metadata. This includes adding or changing users and privileges, and creating or dropping database schemas. Alternatively, you may elect to perform two separate backups. The first backup excludes the `mysql` schema using the `--single-transaction` option. The second backup only includes the `mysql` schema and uses the default locking approach. This will be discussed more in Chapter 2.

Conclusion

An appropriate MySQL backup strategy is an essential component for any running production system. For a simple installation, the implementation of a backup strategy can occur in minutes as demonstrated in this chapter. However, a backup strategy is only as good as the process to perform a successful, timely, and complete recovery using the backup strategy. Chapter 4 will provide a detailed explanation for a successful recovery.

There are a number of important considerations when using the output of `mysqldump` for recovery that may affect how you execute your backup command. Chapter 2 will discuss these points.

This chapter has also introduced a number of common terms including consistent, valid, complete, and point in time. We will define these terms in greater detail in Chapter 2.