

- Types of Backup in MySQL
- Back up a MySQL database using the mysqldump command
 - mysqldump to back up all databases
 - mysqldump to back up only data
 - mysqldump to back up only structure
 - mysqldump to back up all tables in the database
 - mysqldump to back up a single table
- MySQL Workbench to back up a database
- How to back up a database using dbForge Studio for MySQL
- Restore the database in MySQL from the file

Types of Backup in MySQL

In MySQL, there are different types of backup. Which backup strategy to choose depends on several factors, for example, data size, the hardware you use, performance you want to achieve, storage capacity of your database, etc. In addition, you should take into consideration how much time it will take to restore the backup.

MySQL supports the following backup types:

- Logical** backup outputs a database structure in a .sql file by generating the CREATE or INSERT statements. Later, this file can be restored with the help of the mysqldump utility. This type backs up only data without indexes, thus, has a small size. However, its recovery option is slower compared with its alternatives, because it should execute all statements one by one.
- Physical** backup copies database files in the same format, in which they are stored on the disk. It is faster than the logical type but can be restored only to the MySQL server from the same database engine.
- Consistent** backup copies the files at the exact moment of time – only after the server stops running or is locked.

If there may happen some interruptions during a copying operation, backups can be divided into the following types:

- Cold** backup blocks access to data during the backup and does not allow making any changes to data. It is simple, fast, and does not impact performance.

- Hot** backup copies files while the database is running. Users can read and manipulate data.
- Warm** backup makes a database copy while it is running. Users can read but cannot write data during the backup.

Whether data should be fully or partially copied, backups can be classified into:

- Full** backup copies all the data from the database. It can be logical or physical. The full backup can be restored on another server.
- Differential** backup copies all the changes that were made since the latest full backup. The differential backup can be restored only after the full backup was restored.
- Incremental** backup copies all the data changes made since the previous backup (either full or differential).

Back up a MySQL database using the mysqldump command

To back up a MySQL database, you can use either third-party tools or execute the mysqldump command from the command line.

Mysqldump is a command-line utility used to generate a MySQL logical database backup. It creates a single .sql file that contains a set of SQL statements. This file can be used to [create tables](#), objects, and/or insert data that were copied from the database. With the utility, you can dump tables, one or several databases, or the entire server with all databases and tables along with their objects or migrate them to another server.

In addition, the **mysqldump** utility can output the dump to the CSV or XML file formats. It should be noted that mysqldump cannot dump a database or data to separate .sql files.

To restore the database, the utility executes all SQL statements to recreate the tables and populate them with data that, in turn, requires a lot of time.

The basic syntax of the mysqldump command includes the following parameters:

```
mysqldump -u [user name] -p [password] -h [host name] [options] [database_name]
[tablename] > [dumpfilename.sql]
```

- `-u` (or `--user=[username]`) is a username to connect to a MySQL server.
- `-p` is a password for the username you use to connect to the server.
- `-h` (`--host=[hostname]`) is the name of the server you want to connect to dump data from.
- `Options` are additional parameters to configure the backup.
- `database_name` is the name of the database you want to back up.
- `Tablename` is the table name that you want to back up.
- `<` or `>` refers to parameters indicating the process of database backup (`>`) or restore (`<`).
- `dumpfilename.sql` is a path to a dump file containing the database backup.

Next, let's take a closer look at the examples describing how to take a backup of the MySQL database using the `mysqldump` command-line utility.

mysqldump to back up all databases

The example shows how to use `mysqldump` to back up all MySQL databases on the server. The scripts of databases are exported to the `.sql` file, which you can later use, for example, to migrate databases to a new server.

To create a backup of all databases, execute the following command by adding the `-all-databases` parameter:

```
mysqldump --host=IP --user=root --port=3306 --p --all-databases > /backup_all_databases.sql
```

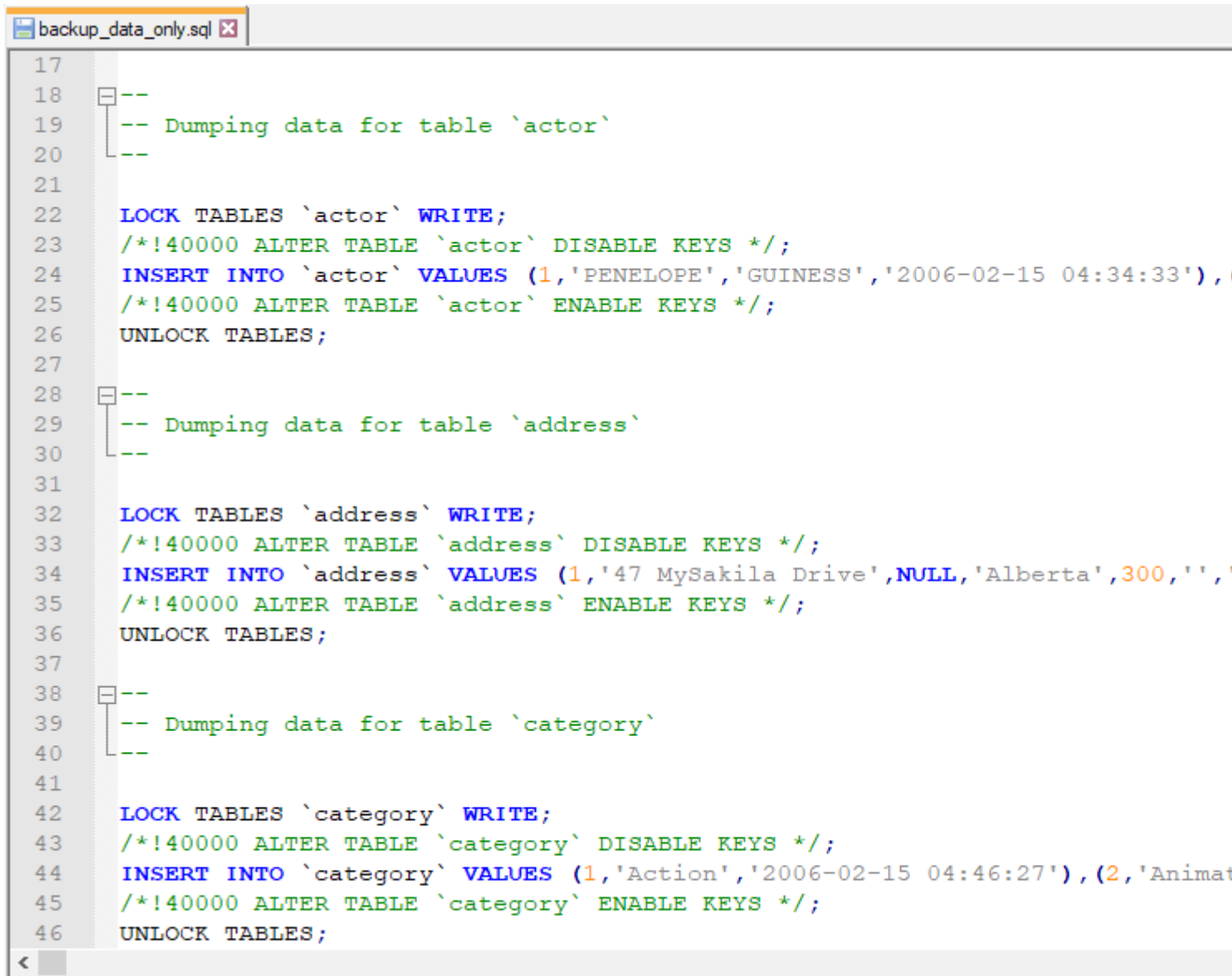
This will generate a backup of all databases with their structure and data to a `backup_all_databases.sql` file.

mysqldump to back up only data

If you want to take a backup of data without the database structure, execute the command with the `-no-create-info` parameter. For example, dump data for the **sakila** database.

```
mysqldump --host=IP --user=root --port=3306 -p --no-create-info sakila >
/backup_data_only.sql
```

The file will output the script containing only dumping table data:



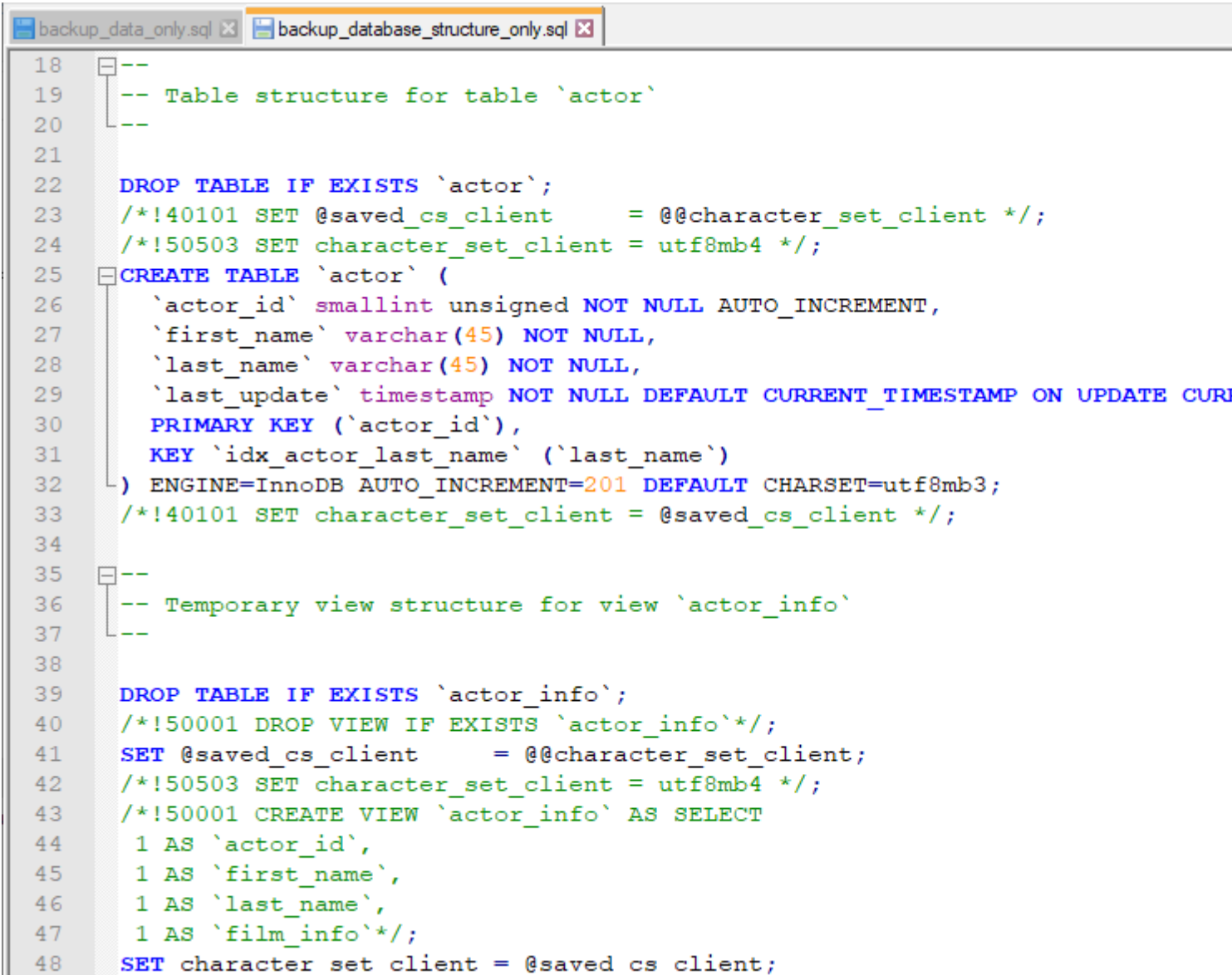
```
17
18  --
19  -- Dumping data for table `actor`
20  --
21
22  LOCK TABLES `actor` WRITE;
23  /*!40000 ALTER TABLE `actor` DISABLE KEYS */;
24  INSERT INTO `actor` VALUES (1,'PENELOPE','GUINNESS','2006-02-15 04:34:33'),
25  /*!40000 ALTER TABLE `actor` ENABLE KEYS */;
26  UNLOCK TABLES;
27
28  --
29  -- Dumping data for table `address`
30  --
31
32  LOCK TABLES `address` WRITE;
33  /*!40000 ALTER TABLE `address` DISABLE KEYS */;
34  INSERT INTO `address` VALUES (1,'47 MySakila Drive',NULL,'Alberta',300,''),
35  /*!40000 ALTER TABLE `address` ENABLE KEYS */;
36  UNLOCK TABLES;
37
38  --
39  -- Dumping data for table `category`
40  --
41
42  LOCK TABLES `category` WRITE;
43  /*!40000 ALTER TABLE `category` DISABLE KEYS */;
44  INSERT INTO `category` VALUES (1,'Action','2006-02-15 04:46:27'),(2,'Animat
45  /*!40000 ALTER TABLE `category` ENABLE KEYS */;
46  UNLOCK TABLES;
```

mysqldump to back up only structure

There may be cases when you need to export the database or table structure without data. To get that done, run the command with the `-no-data` parameter.

```
mysqldump --host=dbfmylast --user=root --port=3306 -p --no-data sakila >
/backup_database_structure_only.sql
```

The mysqldump command outputs only table structure to the file.



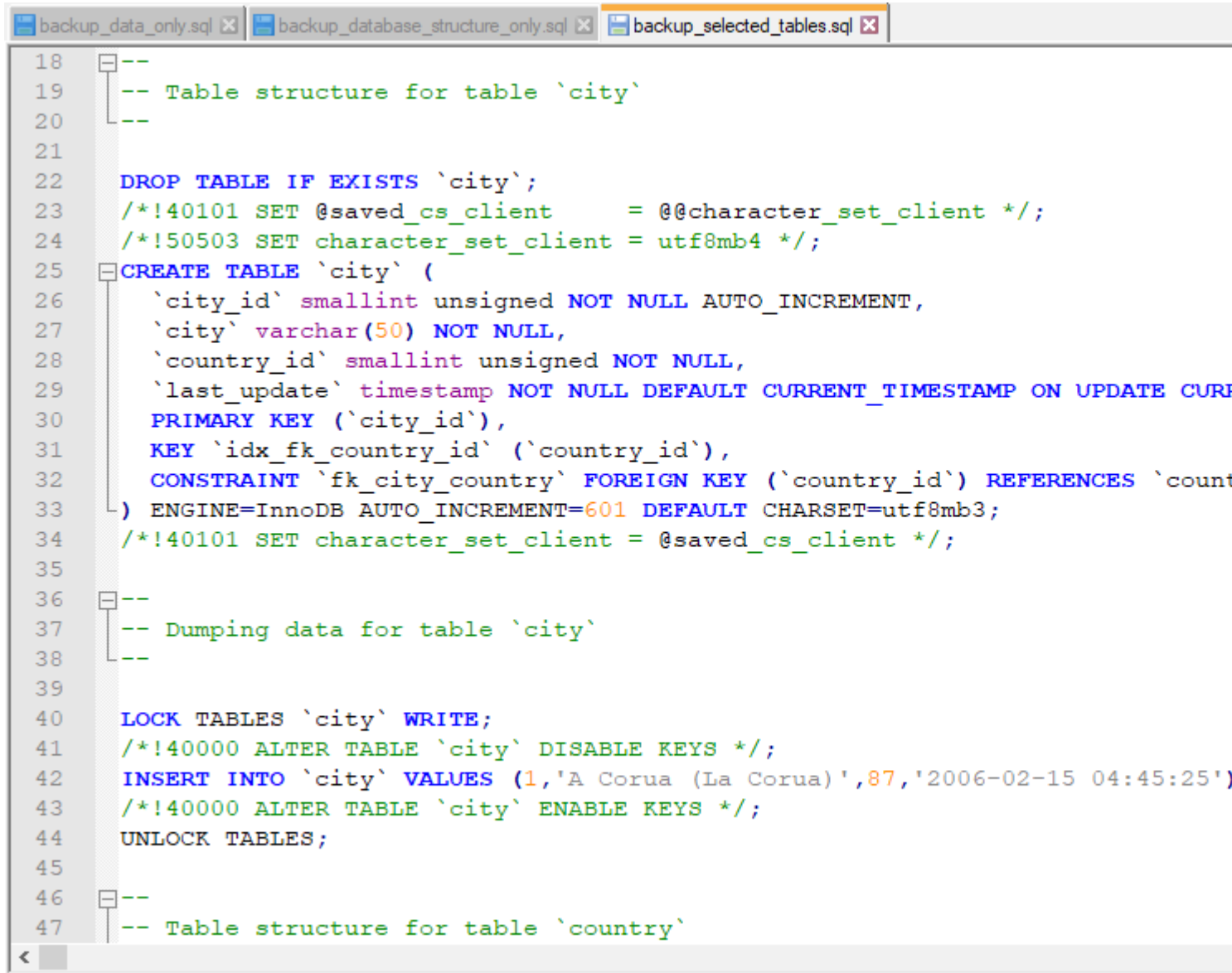
```
18  --
19  -- Table structure for table `actor`
20  --
21
22  DROP TABLE IF EXISTS `actor`;
23  /*!40101 SET @saved_cs_client      = @@character_set_client */;
24  /*!50503 SET character_set_client = utf8mb4 */;
25  CREATE TABLE `actor` (
26    `actor_id` smallint unsigned NOT NULL AUTO_INCREMENT,
27    `first_name` varchar(45) NOT NULL,
28    `last_name` varchar(45) NOT NULL,
29    `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
30    PRIMARY KEY (`actor_id`),
31    KEY `idx_actor_last_name` (`last_name`)
32  ) ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8mb3;
33  /*!40101 SET character_set_client = @saved_cs_client */;
34
35  --
36  -- Temporary view structure for view `actor_info`
37  --
38
39  DROP TABLE IF EXISTS `actor_info`;
40  /*!50001 DROP VIEW IF EXISTS `actor_info`*/;
41  SET @saved_cs_client      = @@character_set_client;
42  /*!50503 SET character_set_client = utf8mb4 */;
43  /*!50001 CREATE VIEW `actor_info` AS SELECT
44    1 AS `actor_id`,
45    1 AS `first_name`,
46    1 AS `last_name`,
47    1 AS `film_info`*/;
48  SET character_set_client = @saved_cs_client;
```

mysqldump to back up all tables in the database

The mysqldump command can generate a backup of all or specific tables in the database by adding the selected table names to the command. Keep in mind that the names of the tables should be separated by a space. For example, dump the category, city, and country tables by running the following command:

```
mysqldump --host=dbfmylast --user=root --port=3306 -p sakila city country category > /backup_selected_tables.sql
```

The output is as follows:



```
18  --
19  -- Table structure for table `city`
20  --
21
22  DROP TABLE IF EXISTS `city`;
23  /*!40101 SET @saved_cs_client      = @@character_set_client */;
24  /*!50503 SET character_set_client = utf8mb4 */;
25  CREATE TABLE `city` (
26    `city_id` smallint unsigned NOT NULL AUTO_INCREMENT,
27    `city` varchar(50) NOT NULL,
28    `country_id` smallint unsigned NOT NULL,
29    `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
30    PRIMARY KEY (`city_id`),
31    KEY `idx_fk_country_id` (`country_id`),
32    CONSTRAINT `fk_city_country` FOREIGN KEY (`country_id`) REFERENCES `country` (`country_id`),
33  ) ENGINE=InnoDB AUTO_INCREMENT=601 DEFAULT CHARSET=utf8mb3;
34  /*!40101 SET character_set_client = @saved_cs_client */;
35
36  --
37  -- Dumping data for table `city`
38  --
39
40  LOCK TABLES `city` WRITE;
41  /*!40000 ALTER TABLE `city` DISABLE KEYS */;
42  INSERT INTO `city` VALUES (1,'A Corua (La Corua)',87,'2006-02-15 04:45:25');
43  /*!40000 ALTER TABLE `city` ENABLE KEYS */;
44  UNLOCK TABLES;
45
46  --
47  -- Table structure for table `country`
```

mysqldump to back up a single table

To take a backup of a single table in the database, indicate the name of the specific table in the mysqldump command. For example, dump the inventory table from the **sakila** database.

Restore the database in MySQL from the file

Now, let's restore the **sakila** database from the .sql output file using the mysqldump utility.

First, you need to create an empty database on the target server and restore the database using the `mysql` command that will generate a schema with data.

```
mysql --host=IP --user=root --port=3306 -p sakila < /dumps/sakila.sql
```

where `sakila` is an empty database that will contain a database structure with data after the import from the `sakila.sql` backup file.

Note: If there is a database with the same name on the target server, then first, you need to drop it and then create an empty database.

To view the list of tables located in the **sakila** database, execute the following `mysql` command:

```
mysql> use sakila;  
  
...  
  
mysql> show tables;
```


Start Command Prompt with Ruby - mysql.exe --host=dbfmylast --user=root -p

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> use sakila;
```

Database changed

```
mysql> show tables;
```

```
+-----+
| Tables_in_sakila |
+-----+
| actor              |
| actor_info         |
| address            |
| category           |
| city               |
| country            |
| customer           |
| customer_list      |
| employees          |
| film               |
| film_actor         |
| film_category      |
| film_list          |
| film_text          |
| inventory          |
| language           |
| my_calendars       |
| nicer_but_slower_film_list |
| payment            |
| rental             |
| sales_by_film_category |
| sales_by_store     |
| staff              |
| staff_list         |
| store              |
| working_hours      |
+-----+
```

26 rows in set (0.00 sec)

```
mysql> _
```