**Unit – 2**

**Database backup and CSV handling**

**SQLITE dump**

The SQlite dump command to backup and restore a database.

.**dump command** that gives you the ability to dump the entire database or tables into a text file.

* **Dump the entire database into a file**

To dump a database into a file, you use the .dump command. The .dump command converts the entire structure and data of an SQLite database into a single text file.

By default, the .dump command outputs the SQL statements on screen. To issue the output to a file, you use the .output FILENAME command.

The following command opens a new SQLite database connection to the chinook .db file.

C:\sqlite>sqlite3 c:/sqlite/chinook.db

SQLite version 3.13.0 2016-05-18 10:57:30

Enter ".help" for usage hints.

sqlite>

The following commands specify the output of the dump file to chinook.sql and dump the chinook database into the chinook.sql file.

sqlite> .output c:/sqlite/chinook.sql

sqlite> .dump

sqlite> .exit

## Dump a specific table using the SQLite dump command

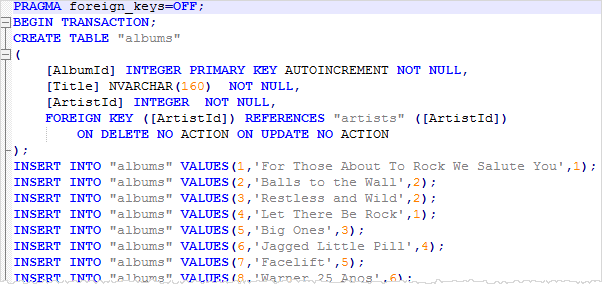
If you want to dump a specific table, you need to specify the table name followed the .dump command. For example, the following command saves the albums table to the albums.sql file.

sqlite> .output c:/sqlite/albums.sql

sqlite> .dump albums

sqlite> .quit

The following picture shows the content of the albums.sql file.



## Dump tables structure

If you want to dump the structures of tables in a database, you use the .schema command. The following commands set the output file to chinook\_structure.sql file and save the structures of tables into the chinook\_structure.sql file.

sqlite> .output c:/sqlite/chinook\_structure.sql

sqlite> .schema

sqlite> .quit

## Dump data of one or more tables into a file

To dump the data of a table into a text file, you use these steps:

First, set the mode to insert using the .mode command as follows:

sqlite> .mode insert

From now on, every [SELECT statement](https://www.sqlitetutorial.net/sqlite-select/) will issue the result as the [INSERT statements](https://www.sqlitetutorial.net/sqlite-insert/) instead of pure text data.

Second, set the output to a text file instead of the default standard output. The following command sets the output file to the data.sql file.

sqlite> .output data.sql

Third, issue the [SELECT](https://www.sqlitetutorial.net/sqlite-select/) statements to query data from a table that you want to dump. The following command returns data from the artists table.

sqlite> select \* from artists;

Check the content of the data.sql file.

To dump data from other tables, you need to issue the SELECT statements to query data from those tables.

## CSV file handling

## Import a CSV file into a table

## We want to import data from CSV file into a table that does not exist in the SQLite database.

* First, the sqlite3 tool creates the table. The sqlite3 tool uses the first row of the CSV file as the names of the columns of the table.
* Second, the sqlite3 tool import data from the second row of the CSV file into the table.

We will import a CSV file named city.csv with two columns: name and population. To import the c:\sqlite\city.csv file into the cities tables:

**First**, set the mode to CSV to instruct the command-line shell program to interpret the input file as a CSV file. To do this, you use the .mode command as follows:

sqlite> .mode csv

Second, use the command .import FILE TABLE to import the data from the city.csv file into the cities table.

sqlite>.import c:/sqlite/city.csv cities

## To verify the import, you use the command .schema to display the structure of the cities table.

## sqlite> .schema cities

## CREATE TABLE cities(

## "name" TEXT,

## "population" TEXT );

## The table is already available in the database and you just need to import the data.

## Second, use the following CREATE TABLE statement to create the table cities.

## CREATE TABLE cities(

## name TEXT NOT NULL,

## population INTEGER NOT NULL

## );

## If the table already exists, the sqlite3 tool uses all the rows, including the first row, in the CSV file as the actual data to import. Therefore, you should delete the first row of the CSV file.

## The following commands import the city\_without\_header.csv file into the cities table.

## sqlite> .mode csv

## sqlite> .import c:/sqlite/city\_no\_header.csv cities

## Export a CSV file from table

## By using the sqlite3 tool, you can use the SQL statements and dot-commands to interact with the SQLite database.

To export data from the SQLite database to a CSV file, you use these steps:

1. Turn on the header of the result set using the .header on command.
2. Set the output mode to CSV to instruct the sqlite3 tool to issue the result in the CSV mode.
3. Send the output to a CSV file.
4. Issue the query to select data from the table to which you want to export.

## The following commands select data from the customers table and export it to the data.csv file.

>sqlite3 c:/sqlite/chinook.db

sqlite> .headers on

sqlite> .mode csv

sqlite> .output data.csv

sqlite> SELECT customerid,

...> firstname,

...> lastname,

...> company

...> FROM customers;

sqlite> .quit

## If you check the data.csv file, you will see the following output.

## F:\bca\sem 3\SQLite-Export-CSV-example 2.png

## Besides using the dot-commands, you can use the options of the sqlite3 tool to export data from the SQLite database to a CSV file.

## For example, the following command exports the data from the tracks table to a CSV file named tracks.csv.

## >sqlite3 -header -csv c:/sqlite/chinook.db "select \* from tracks;" > tracks.csv

## F:\bca\sem 3\SQLite-Export-CSV-one-liner-option.png

If you have a file named query.sql that contains the script to query data, you can execute the statements in the file and export data to a CSV file.

**>sqlite3 -header -csv c:/sqlite/chinook.db < query.sql > data.csv**

**\*\*\*\*\*\*\*\*\*\*\***