EDAF75
Database Technology

Lecture 5

Christian.Soderberg@cs.lth.se

February 2, 2026

Administration

- Lab 1 this week, lab 2 and lab 3 the following two weeks
- You'll have to sign up for each lab separately (so you don't have to be available at the same time each week)
- All groups will be in the same room (E:Jupiter)

Today's lecture

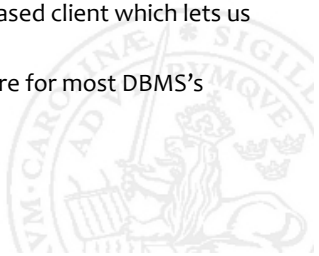
- A few words about keeping track of state
- Different ways of connecting to our database
- Connecting to a database from Python using the `sqlite3` package

Keeping track of state

- Using mutable state is often simple, but it comes with some problems:
  - We may need to lock our object before updating
  - We know what our state is, but can't explain why it is what it is
- A very popular alternative is to instead keep track of changes, and calculate our new state from the changes – this technique is called *event sourcing*
  - Just adding updates is much 'cheaper' than modifying values
  - We get the history for free, which allows us to explain the state in detail

Running SQLite3

- Traditional DBMS's such as PostgreSQL, MariaDB, Oracle, MySQL, etc., run as servers, often running on remote machines
- We can access our database servers using various kinds of clients – most of them offer command line clients which talks to the servers using simple text
- SQLite3 doesn't run as a server, but it has a simple text based client which lets us manipulate our databases just as traditional databases do
- There are also some nice GUIs for SQLite3, just as there are for most DBMS's

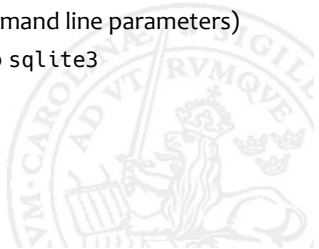


Task

Exercise: Use `sqlitebrowser` to look at the college application database from the previous lecture

SQLite3– command line client

- We can run `sqlite3` from a shell, it works in Linux, Mac and Windows
- When we start `sqlite3` with a filename as an argument, our database will be saved in the file
- We can either give commands from within, at its prompt, or send commands using redirection (we can actually even send them as command line parameters)
- It's very convenient to write SQL scripts and send them to `sqlite3`



Task

Exercise: Use the `sqlite3` CLI to look at the database from the previous lecture

SQLite3– command line client

Some useful SQLite3 commands:

- `.help`: makes this slide superfluous
- `.tables`: shows all tables in the current database
- `.schema <table>`: shows how a table is defined
- `.dump <table>`: gives INSERT statements for creating the specified table
- `.import <filename> <table>`: imports data into a table
- `.read <filename>`: reads a script from a file
- `.save <filename>`: writes the current database to file

SQLite3 format

We can get various output formats, using `.mode` – some examples (there are more):

- `csv`: Comma-separated values
- `column`: Left-aligned columns
- `html`: HTML code
- `insert`: SQL insert statements
- `line`: One value per line
- `list`: Values delimited by some separator
- `tabs`: Tab-separated values

Task

Exercise: Use the SQLite3 command line client to create a database with the college application data we've been using during the first few lectures.

Task

Exercise: Write an SQL-script showing the names and grades of the students who have applied for Computer Science at Stanford. Let SQLite3 run the script and generate output.

Python and `sqlite3`

- `sqlite3` is a lightweight standard library
- We create a `Connection` to our database
- We create a `Cursor` from our `Connection`, and call its `execute` method
- After the `execute` call, we can treat our `Cursor` as an iterator to fetch the results

Task

Exercise: Write a Python program showing the names and grades of the students who have applied for Computer Science at Stanford.

Task

Exercise: Write a Python program to update the gpa by 4% for all students who have applied to Stanford.

Task

Exercise: Write a Python program which accepts students into a given program.