EDAF75
Database Technology

Lecture 5

Christian.Soderberg@cs.lth.se

February 3, 2025

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

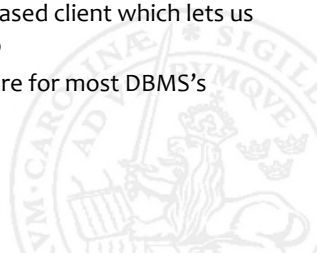
- A few words about keeping track of state
- Different ways of connecting to our database
- Connecting to a database from Java using `JDBC`
- Connecting to a database from Python using the `sqlite3` package

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

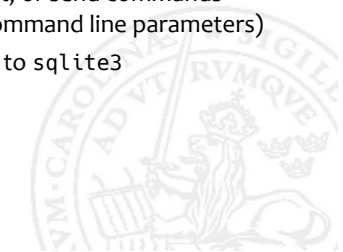


Task

Exercise: Use `sqlitebrowser` to look at the database from the first lectures

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 first lectures

SQLite3– command line client

Some useful SQLite3 commands:

- `.help`: makes this slide futile
- `.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

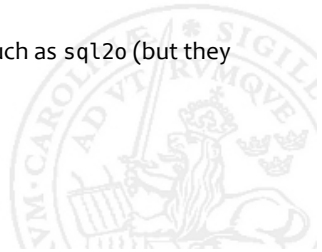
Exercise: Use the SQLite3 command line client to create a database with the college application data we've been using during the first few lectures.

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.

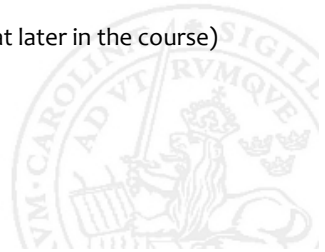
Java Database Connectivity (JDBC)

- Standard classes for handling database connections
- Can handle all relevant relational databases
- Based upon *connections*, *statements*, and *result sets*
- Lots of things can go wrong when we connect to databases, so JDBC requires lots of exception handling
- There are also some alternative, non-standard libraries, such as `sql2o` (but they often depend on `JDBC`)



`Connection`

- Used to set up a *session* to a database (in the case of `SQLite3`, we don't really need to connect, the database is in a file on our hard drive, or even in memory)
- Creates `Statement`-objects, which we can use to send SQL statements to our database
- Connections also handle *transactions* (we'll talk about that later in the course)



Statements

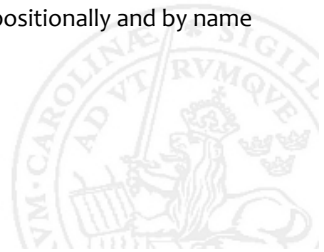
- There are two major kinds of statements:
  - `Statement`: a simple but unsafe kind of statement (amenable to *SQL injection*)
  - `PreparedStatement`: a precompiled statement, safer, and more efficient when executed multiple times
- We *always* use '*try-with-resources*' to create statements, to make sure that they are closed properly when we finish

`PreparedStatement`

- Created with `prepareStatement(str)` on a connection, where `str` is a `String` containing a query or statement
- The query or statement can contain parameters, marked as ? – they get their values with various `setXXX`-methods
- Some important methods:
  - `ResultSet executeQuery()`: executes a query
  - `int executeUpdate()`: executes an update

ResultSet

- A `ResultSet` which represents the table of data returned from an SQL query
- It's a kind of iterator (but doesn't implement any iterator interface), we call `next()` to jump to the next row, and it returns `false` if there is no next row
- We can use various `getXXX`-methods to fetch data, both positionally and by name



JDBC, code sample

```
var found = new ArrayList<ReturnType>();
var query =
    """
    SELECT  ...
    FROM    ...
    WHERE   ... = ?
    ...
    """;
try (var ps = conn.prepareStatement(query)) {
    ps.setString(1, ...);     // set parameter value
    var resultSet = ps.executeQuery();
    while (resultSet.next()) {
        found.add(ReturnType.fromRS(resultSet));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
return found;
```


Task

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

Task

Exercise: Write Java code to update the gpa by 4% for all students who have applied to Stanford.

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 which lets us add students and applications to the student database

Task

Exercise: Solve the previous problems in Python, using the `sqlite3` package.