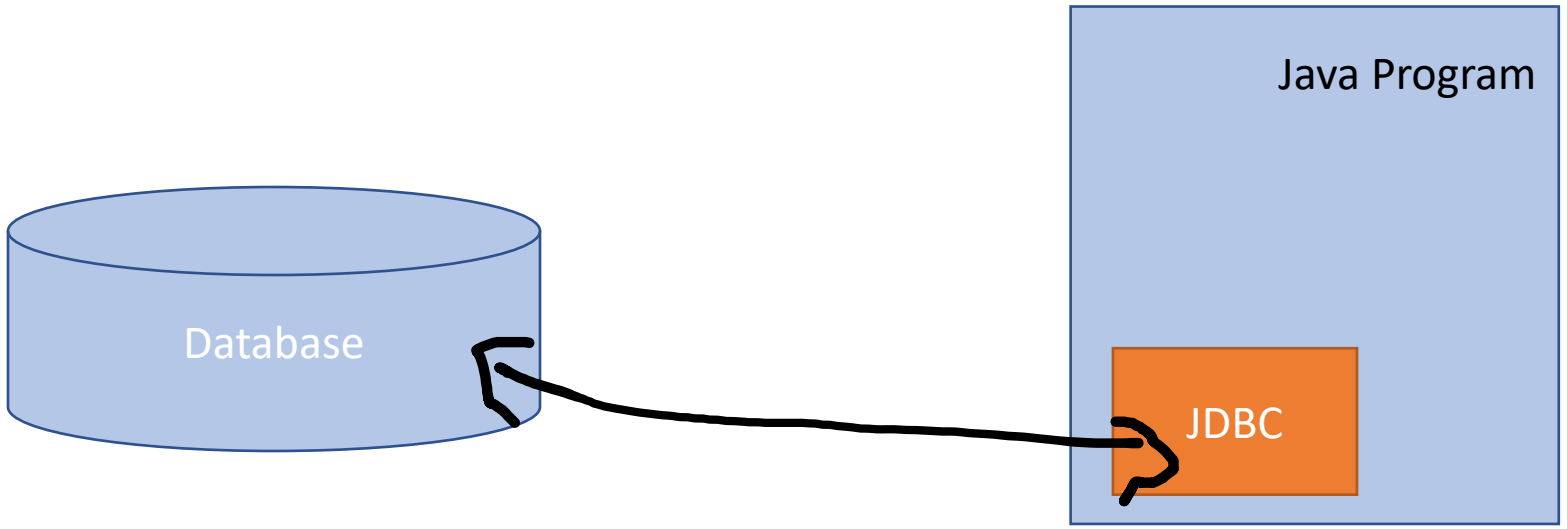
JDBC

Week 10 – Presentation 3

Java Database Connectivity (JDBC)

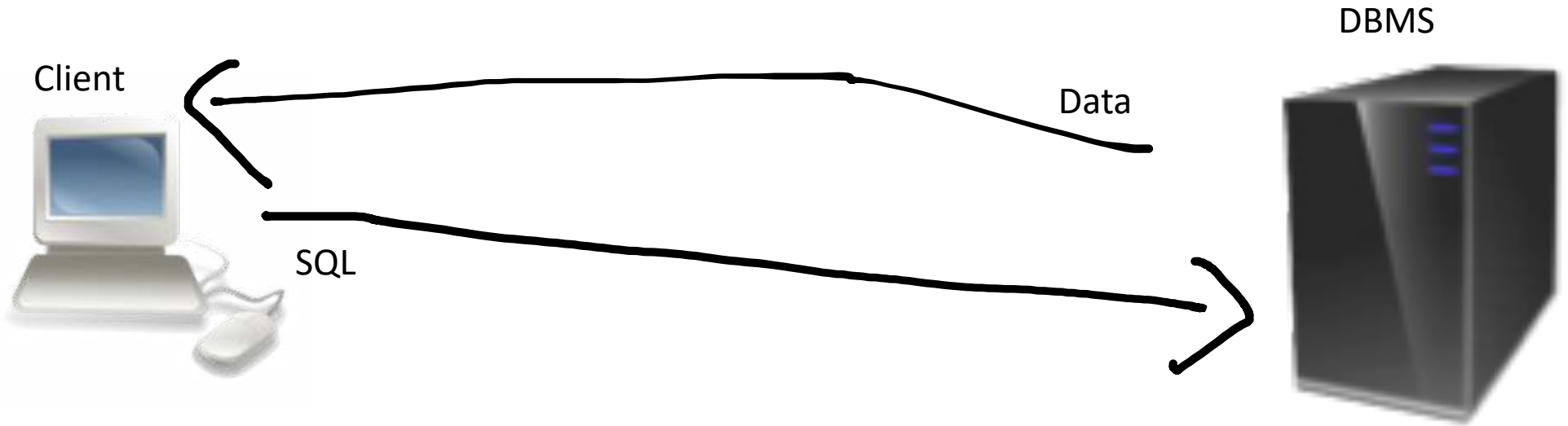


Accessing a database and sending/executing SQL from a Java program is easy, even without using some tools that try to write queries for you (sometimes these tools even write inefficient queries).

Accessing a database from a Java program

Classic case for a real database management server:

Your program must first connect to the database (which usually required supplying a username and a password) then will send strings that contain an SQL command and will be executed by the server. Some commands will only return success or failure (creating a table, for instance). A command such as an update will be able to tell you how many rows were changed. A query will return you rows, between 0 and n...



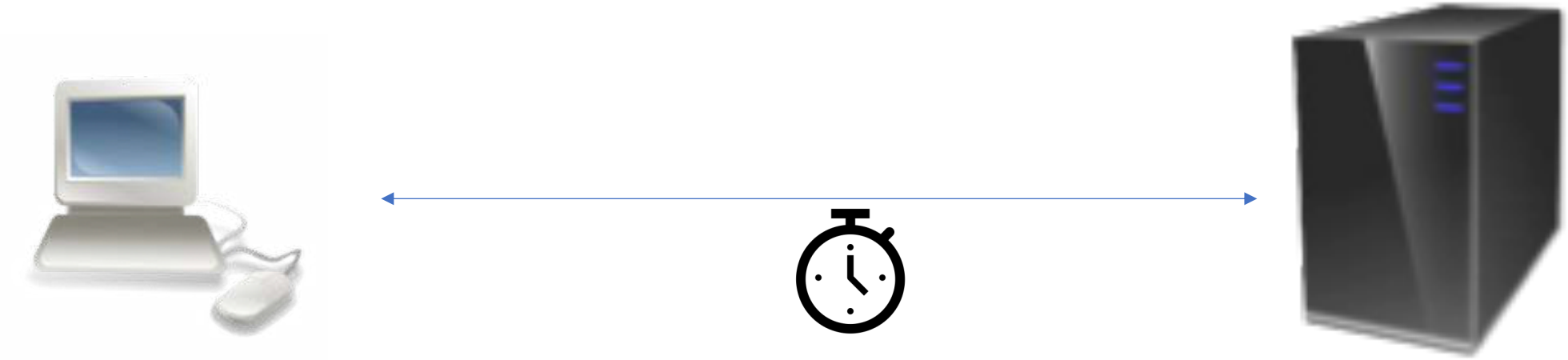
System Architecture: Classic Design Mistake

- Many Java developers do this wrong…
- Must SHARE the work between what the database does ...
  - Retrieve exactly what you need

  If you want a sum, don't retrieve all the data and iterate to sum it. Ask for the sum. The DBMS can compute it.

- And what the java program does
  - Presentation computations that cannot be performed by the DBMS

System Architecture: Consider Network Latency



Don't multiply exchanges between your application and the server. Travelling in networks takes time (it's called latency), even with a lot of bandwidth. Sometimes data centres are very far away.

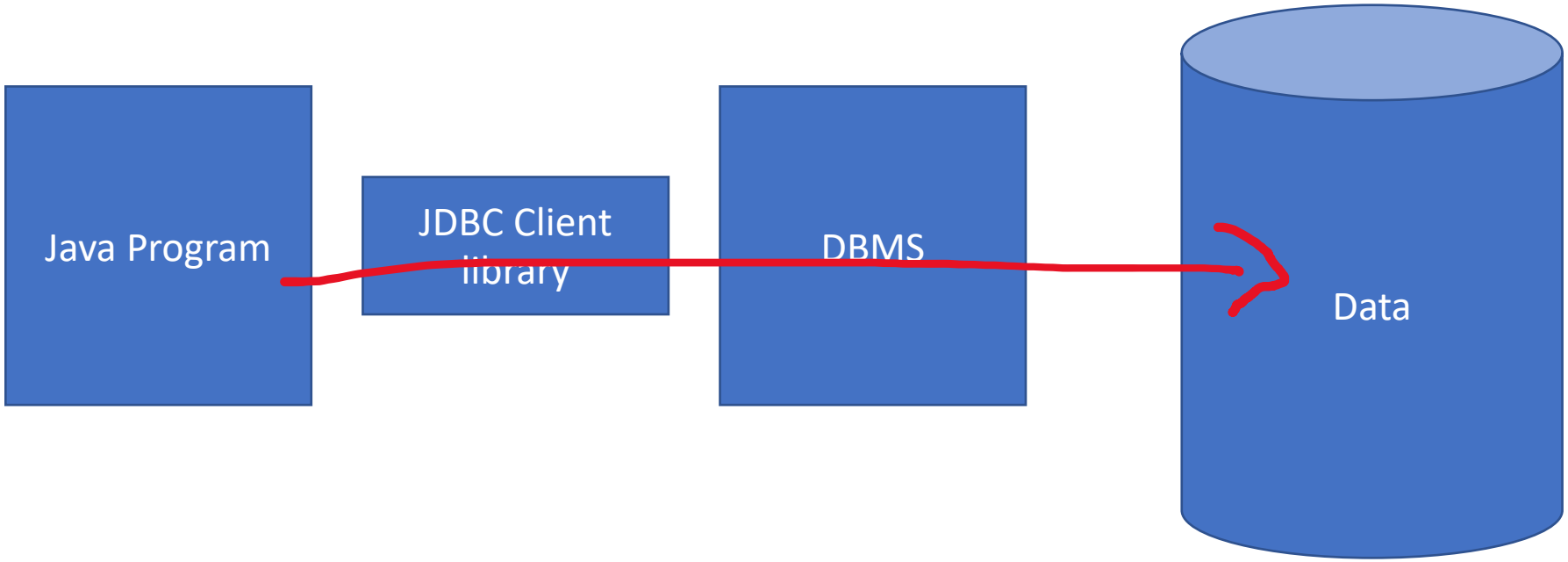
System Architecture: One INSERT is okay but 10K INSERTs is BAD

If you want to insert many rows into a remote database, send all the rows to the server and ask it to insert all the rows at once (this is called "batching"); it will do it very fast. If you insert rows one by one, you have to wait each time for the response from the server. A return trip between Singapore and Tokyo takes about 0.075s. If you assume that inserting 10,000 rows takes 0.050s* doing it as one batch between Singapore and Tokyo would take 0.125s. Inserting one row about 0.003s*. Inserting 10000 rows row by row would be (0.075 + 0.003) * 10000 = 780s, or 13 minutes ...

USE BATCHING

*tests by Stephane Faroult in 2017

Even when you are close…



Even when running on the same server, switches between a program and the DBMS are costly.

System Architecture Rules

- Share work between the DBMS and the program
- Consider network latency
- Use batching

Embedded Databases

An alternative to real database servers are "embedded databases", systems that let you use of file as if it were a database.

No server just for a single-user

"Connection" same as opening a file

Everything else like the real thing

Apache Derby

- Pure Java
- Part of the JDK
- Server or embedded

- Java is shipped with "Derby", sometimes called "Java DB"

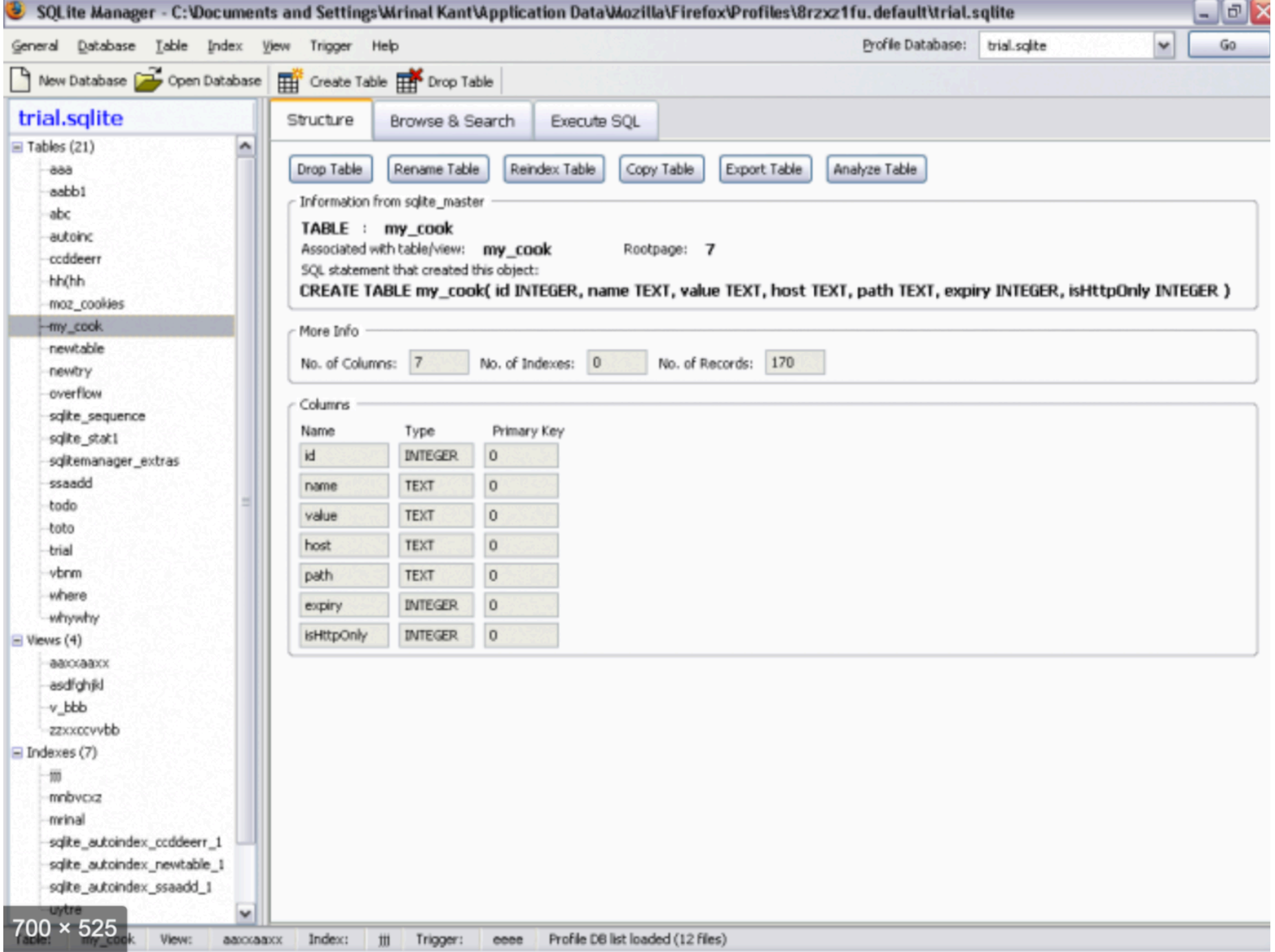
SQLite

- Public domain
  One file to download

- Used in mobile apps – and by Mozilla

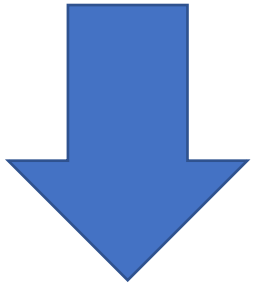
- Create tables in *SQLite Manager* (a firefox plugin)

The most popular embedded database is probably SQLite.


SQLite Manager - C:\Documents and Settings\Mrinal Kant\Application Data\Mozilla\Firefox\Profiles\8rzxz1fu.default\trial.sqlite
General Database Table Index View Trigger Help
Profile Database: trial.sqlite
Go
New Database
Open Database
Create Table
Drop Table
trial.sqlite
Tables (21)
aaa
aabb1
abc
autoinc
ccddeerr
hh(hh
moz_cookies
my_cook
newtable
newtry
overflow
sqlite_sequence
sqlite_stat1
sqlitemanager_extras
ssaadd
todo
toto
trial
vbnm
where
whywhy
Views (4)
aaxxaaxx
asdfghjkl
v_bbb
zzxxccvvbb
Indexes (7)
jjjj
mnbvcxz
mrinal
sqlite_autoindex_ccddeerr_1
sqlite_autoindex_newtable_1
sqlite_autoindex_ssaadd_1
uytre
Structure
Browse & Search
Execute SQL
Drop Table
Rename Table
Reindex Table
Copy Table
Export Table
Analyze Table
Information from sqlite_master
TABLE : my_cook
Associated with table/view: my_cook Rootpage: 7
SQL statement that created this object:
CREATE TABLE my_cook(id INTEGER, name TEXT, value TEXT, host TEXT, path TEXT, expiry INTEGER, isHttpOnly INTEGER)
More Info
No. of Columns: 7
No. of Indexes: 0
No. of Records: 170
Columns
Name Type Primary Key
id INTEGER 0
name TEXT 0
value TEXT 0
host TEXT 0
path TEXT 0
expiry INTEGER 0
isHttpOnly INTEGER 0
Table: my_cook View: aaxxaaxx Index: jjjj Trigger: eeee Profile DB list loaded (12 files)

700 × 525

www.sqlite.org



Only for C programming language

JDBC: https://github.com/xerial/sqlite-jdbc

You just need a .jar file available from this address…

Java Database Connectivity (JDBC)

sqlite-jdbc refers to JDBC, which is a protocol allowing access to almost all databases from Java. SQL is often slightly different from database to database, not JDBC calls.

JDBC

- Load a specific connection driver (a .jar file)

  Database specific connection parameters

- Then java methods are the same with every database

- SQL is slightly different between databases

3 Main Objects

- **Connection**
  Link to a database
- **PreparedStatement**
  SQL commands:
- **ResultSet**
  Returned results

You can execute a query and do things like loop on rows returned. There is also a Statement object. A PreparedStatement can be re-executed with different parameters, but a Statement is only sent once.

Java/JDBC Example

```
import java.util.Properties;
import java.sql.*;
import java.util.Scanner;

class DBExample {
    static Connection con = null;

    public static void main(String arg[]) {
        Properties info = new Properties();
        String url = "jdbc:oracle:thin:@localhost:1521:orcl
        Scanner input = new Scanner(System.in);

        try {
            Class.forName("oracle.jdbc.OracleDriver");
        } catch(Exception e) {
            System.err.println("Cannot find the driver.");
            System.exit(1);
        }
```

- JDBC
- Caution: the driver must be in the classpath
- You connect to the database using a url string not a URL object…

Java/JDBC Example

```
try {
    Class.forName("oracle.jdbc.OracleDriver");
} catch(Exception e) {
    System.err.println("Cannot find the driver.");
    System.exit(1);
}

try {
    System.out.print("Username: ");
    String username = input.nextLine();
    System.out.print("Password: ");
    String password = input.nextLine();
    info.put("user", username);
    info.put("password", password);
    con = DriverManager.getConnection(url, info);
    con.setAutoCommit(false);
    System.out.println("Successfully connected.");
} catch (Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
}
```

- MySQL wants username and password independent parameters,
- Oracle passes them as Properties.
- Of course the connection can fail.

Java/JDBC Example

```
System.out.print("Date: ");
String utcDate = input.nextLine();
try {
    PreparedStatement stmt = con.prepareStatement("select region, count(*)"
            + " from quakes"
            + " where UTC_date >= ?" + " group by region");
    stmt.setString(1, utcDate);
    ResultSet rs = stmt.executeQuery();
    while (rs.next()) {
        System.out.println(rs.getString(1) + "\t" + rs.getString(2));
}
rs.close();
```

- Queries are simple. Placemarkers for parameters in a PreparedStatement are ? marks, implicitly numbered (from 1). Return columns are also numbered from 1. One can often only work with Strings, the DBMS can convert types.

Java/JDBC Example

```
            rs.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
        try {
            con.close();
        } catch (SQLException sqlE) {
        // Ignore
        }
        System.exit(1);
    }
    try {
        con.close();
    } catch (SQLException sqlE) {
    // Ignore
    }
```

Queries can fail too, but returning nothing or updating or deleting or inserting no rows doesn't throw any exception because the empty set is a valid set … However trying to insert the same key for instance will throw an exception

Common drivers

```
Class.forName("oracle.jdbc.OracleDriver");
Class.forName("com.mysql.jdbc.Driver");
Class.forName("org.postgresql.Driver");
Class.forName("org.sqlite.JDBC");
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

These are JDBC driver names for commonly use Database Management systems (note that for Derby there are two modes, embedded – single user – and not embedded). Note also that although most tutorials use reflection to load the driver, you can also use import with it, especially with Derby, as the driver will always be in your class path.

Connection Examples

```
con = DriverManager.getConnection(url, info);
    "jdbc:oracle:thin:@hostname:port:dbname"
    "jdbc:postgresql://hostname:port/dbname"
     user password

 con = DriverManager.getConnection(url,
                                   username,
                                    password);
    "jdbc:mysql://hostname:port/dbname"

 con = DriverManager.getConnection(url);
    "jdbc:sqlite:filename"
```

- And here are connection examples.
- With Sqlite you just provide a filename. It will be created if it doesn't exist already.

Additional practice: practical