CIS4301 Notes: Frontend Design

Ryan Roden-Corrent

April 7, 2014

1 JDBC

JDBC is a Java interface for interacting with a database. It tries to abstract away differences between various databases. See the docs HERE.

1.1 Driver Manager

java.sql.DriverManager
Used to load libraries needed to access database. You can get the Postgres JDBC driver HERE

Listing 1: loading the Postgres driver

```
Class.forName("org.postgresql.Driver")
```

use **java -classpath** to add the driver to Java's classpath

Listing 2: connecting to a Postgres DB

```
try {
  connection = DriverManager.getConnection(
    "jdbc:postgresql://5432/my_database_name", "username", "password"
  );
} catch ...
```

1.2 PreparedStatement

A regular statement builds a query, sends it to the database. The database parses the statement and turns it into a query plan. Normal flow is Query → Parser → Rewriter → Planner
A prepared statement is already partially formed - which is better for performance.

Listing 3: prepared statement

```
try {
  PreparedStatement ps =
    // prepare a query from a string containing SQL
    // the ? is a parameter
    String myQuery = "SELECT * FROM NCAA WHERE losers = ?";
    connection.prepareStatement(myQuery);
    ps.setString("the parameter input"); // adds quotes automatically
    ResultSet rs = ps.executeQuery();
    // The ResultSet has a 'cursor' that you can move through the results
    // use next() to iterate over the query results
    // the cursor could also be used to modify the Table
    while (rs.next() {
      //indexed from 1
      String result = rs.getDate(1);
} catch ...
```

1.2.1 SQL Injection

Listing 4: a simple query

```
SELECT *
FROM Table
WHERE User = $_GET['user'];
```

Suppose somebody inputs `"username; DROP DATABASE"`. By adding SQL statements to their username input, users could run commands on your database. Prepared statements can protect against this, as it is more difficult to insert malicious statements into a partially formed query that is only expecting input of a specific typerun commands on your database. Prepared statements can protect against this, as it is more difficult to insert malicious statements into a partially formed query that is only expecting input of a specific type.

1.2.2 Prepared Statements in Postgres

You can create prepared statements directly in Postgres. See PREPARE

2 EXPLAIN and ANALYZE

Used to check performance of SQL statements. See EXPLAIN

Listing 5: a simple query

```
EXPLAIN SELECT ... FROM ... WHERE ...;
```

If the result shows a high count for rows=, look into ways to limit the rows explored.