Postgres a JAVA

Databázové praktikum

JDBC – Java DataBase Connectivity

- Jednotné API pre prístup k „tabuľkovým“ dátam
- „... Access virtually any data source, from relational databases to spreadsheets and flat files“.
  - JDBC dokumentácia
- Použijeme to na pripojenie sa na našu Postgres databázu

- https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Základné kroky ako pracovať s databázou

1. Establish a **connection**
2. Create JDBC **statements**
3. Execute **SQL** statements
4. Get **ResultSet**
5. Close connection

1. Establish a connection

- **import java.sql.*;**
- **Načítame „vendor specific driver“**
  - Class.forName(“org.postgresql.Driver");
    - Dynamicky načíta triedu s driverom pre Postgres
- **Vytvoríme spojenie**
  - Connection con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/username", username, passwd);

    - Vytvorí sa TCP/IP spojenie medzi Vašim programom a Postgresom
    - Špecifikum pre naše prostredie:
      - Username: Vaše AIS prihlasovacie meno,
      - Passwd: heslo, ktoré si predtým musíte nastaviť (o tom neskôr).

2. Create JDBC statement(s)

- Statement stmt = con.createStatement() ;
- Vytvorí sa objekt typu Statement, ktorý budeme používať na posielanie SQL dotazov na databázu.

3. Executing SQL Statements

- ```
  String query = "Create table tblname "
      + "(id Integer not null, name VARCHAR(32), "
      + “marks Integer)";
  stmt.executeUpdate(query);
  ```

- ```
  String query = "Insert into tblname values“
      + "(123456789,’abc’,100)";
  stmt.executeUpdate(query);
  ```

4. Get ResultSet

```
String query = "select * from Lehigh";

ResultSet rs = Stmt.executeQuery(query);


while (rs.next()) {
  int id= rs.getInt(“id");
  String name = rs.getString(“name");
  int marks = rs.getInt(“marks");
}
```

5. Close connection

- stmt.close();
- con.close();

Tranzakcie a JDBC

- JDBC podporuje zoskupenie SQL dotazov do tranzakcií
- Za kontrolu nad tranzakciami zodpovedá objekt Connection object, default mód je auto-commit, t.j., každý SQL dotaz je zabalený ako samostatná tranzakcia
- Auto-commit môžeme vypnúť pomocou con.setAutoCommit(false);
- A tiež ho potom môžeme zapnúť con.setAutoCommit(true);
- Akonáhle je auto-commit vypnutý, žiadny SQL dotaz nebude comitovaný, kým nezavoláme con.commit();
- Po commitnutí tranzakcie sú všetky vykonané zmeny permanentné.

Spracovávanie chýb pomocou Exceptions

- Programs should recover and leave the database in a consistent state.
- If a statement in the try block throws an exception or warning, it can be caught in one of the corresponding catch statements
- How might a finally {…} block be helpful here?
- E.g., you could rollback your transaction in a catch { …} block or close database connection and free database related resources in finally {…} block

Typy JDBC - Java

JDBC Type	Java Type
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT DOUBLE	double
BINARY VARBINARY LONGVARBINARY	byte[]
CHAR VARCHAR LONGVARCHAR	String

JDBC Type	Java Type
NUMERIC DECIMAL	BigDecimal
DATE	java.sql.Date
TIME TIMESTAMP	java.sql.Timestamp
CLOB	Clob*
BLOB	Blob*
ARRAY	Array*
DISTINCT	mapping of underlying type
STRUCT	Struct*
REF	Ref*
JAVA_OBJECT	underlying Java class

*SQL3 data type supported in JDBC 2.0

Prepared statements

- https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html

- Predkompilovaný „parametrizovaný“ dotaz.
  - PreparedStatement stmt
    = con.preparedStatement(„SELECT * FROM student WHERE meno=? and priezvisko=?”)
  - stmt.setString(1,”jozef”);
  - stmt.setString(2,”mrkvicka”);
  - `ResultSet rs = stmt.executeQuery();`

Prepared statements

- Výhody prepared dotazov:
  - **Bezpečnosť –** zabraňujú SQL injection útokom
  - **Efektívnosť –** dotaz je „parsovaný“ len raz

- **Ak SQL dotaz obsahuje parametre, ktoré má pod kontrolou klient, určite použite Prepared statement**

- Ak nepoužívate prepared statement, musíte zabezpečiť korektné escape-ovanie vstupu.
  - JDBC na to nemá podporu (keďže je to špecifikum každého DB systému)
  - Napr. s využitím „Dollar quoting“ (špecialita Postgresu):
  - http://www.postgresql.org/docs/current/static/sql-syntax-lexical.html#SQL-SYNTAX-DOLLAR-QUOTING