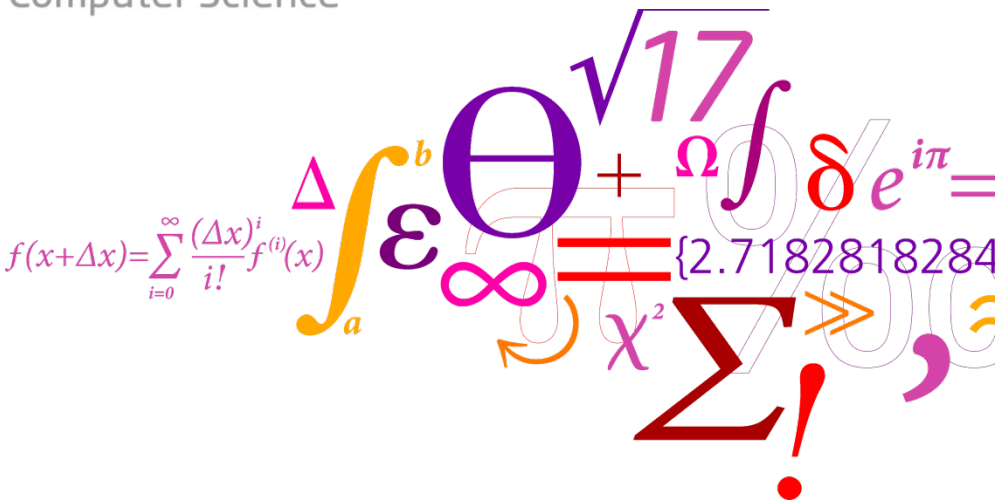
Slides PiSU-L07.3

Projekt i software-udvikling (02362)
Forår 2020

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



5. JDBC: Prepared Statements

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Prepared statements er en nem (og lidt mindre fejlbehæftede) metode for at interagere med en database.
- Man kan bruge SELECT-statements for at opdatere tabeller
- I min kode har jeg kun et eneste INSERT-Statement og ingen UPDATE-statements; alle andre er SELECT-statements, som også bliver brugt til at opdatere og endda til at tilføje data til databasen (→ slide 11)

Connector

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class Connector {

    private static final String HOST     = "localhost";
    private static final int    PORT     = 3306;
    private static final String DATABASE = "pisu20";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "";

    private static final String DELIMITER = ";;";

    private Connection connection;

    Connector() {
        try {
         String url = "jdbc:mysql://" + HOST + ":" + PORT + "/" +
                      DATABASE + "?serverTimezone=UTC";
         connection = DriverManager.getConnection(url, USERNAME, PASSWORD

         createDatabaseSchema();
      } catch (SQLException e) {
         // TODO we should try to diagnose and fix some problems here and
         //      exit in a more graceful way
         e.printStackTrace();
      }
    }
```

Connector …

```
private void createDatabaseSchema() {

    String createTablesStatement;
    try {
        ClassLoader classLoader = Connector.class.getClassLoader();
        URI uri = classLoader.
             getResource("schemas/createschema.sql").toURI();
        byte[] bytes = Files.readAllBytes(Paths.get(uri));
        createTablesStatement = new String(bytes);
    } catch (URISyntaxException | IOException e) {
        e.printStackTrace();
        return;
    }

    try {
        connection.setAutoCommit(false);

        Statement statement = connection.createStatement();
        for (String sql : createTablesStatement.split(DELIMITER)) {
            if (!StringUtils.isEmptyOrWhitespaceOnly(sql)) {
                statement.executeUpdate(sql);
            }
        }
        statement.close();

        connection.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        // TODO error handling
        try {
          connection.rollback();
        } catch (SQLException e1) {}
    } finally {
        try {
          connection.setAutoCommit(true);
        } catch (SQLException e) {}
    }
}
```

Koden sørger for at tabellerne bliver automatisk kreeret i databasen, hvis de ikke eksisterer i forvejen! Dertil læses skemate fra "createschema.sql" resourcen.

Det hele skal foregår transaktionelt (helt eller ingenting).

```
        try {
            connection.setAutoCommit(false);
            Statement statement = connection.createStatement();
            for (String sql : createTablesStatement.split(DELIMITER)
                if (!StringUtils.isEmptyOrWhitespaceOnly(sql)) {
                    statement.executeUpdate(sql);
                }
            }

            statement.close();
            connection.commit();
        } catch (SQLException e) {
            e.printStackTrace();
            // TODO error handling
            try {
              connection.rollback();
            } catch (SQLException e1) {}
        } finally {
            try {
                connection.setAutoCommit(true);
            } catch (SQLException e) {}
        }
    }
```

SQL (Excerpt) …

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
CREATE TABLE IF NOT EXISTS Game (

  gameID int NOT NULL UNIQUE AUTO_INCREMENT,

  name varchar(255),

  currentPlayer tinyint NULL,
  phase tinyint,
  step tinyint,

  PRIMARY KEY (gameID),
  ...

);;
```

Her er et udtræk af "createschema.sql" resourcen.

SQL (Excerpt)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
CREATE TABLE IF NOT EXISTS Player (

  gameID int NOT NULL,
  playerID tinyint NOT NULL,

  name varchar(255),
  colour varchar(31),

  positionX int,
  positionY int,
  heading tinyint,

  PRIMARY KEY (gameID, playerID),
  FOREIGN KEY (gameID) REFERENCES Game(gameID)
  ...

);;

...
```

DAL: Interface

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface IRepository {

   boolean createGameInDB(Board game);

   boolean updateGameInDB(Board game);

   Board loadGameFromDB(int id);

   List<GameInDB> getGames();

}
```

DAL: Implementering

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class Repository implements IRepository {

   private static final String GAME_GAMEID = "gameID";

   private static final String GAME_NAME = "name";

   private static final String GAME_CURRENTPLAYER =
      "currentPlayer";

   private static final String GAME_PHASE = "phase";

   private static final String GAME_STEP = "step";

   ...
```

```
public boolean createGameInDB(Board game) {
    if (game.getGameId() == null) {
        Connection connection = connector.getConnection();
        try {
            connection.setAutoCommit(false);

            PreparedStatement ps = getInsertGameStatementRGK();
            ps.setString(1, "Date: " +  new Date());
            ps.setNull(2, Types.TINYINT); //
            ps.setInt(3, game.getPhase().ordinal());
            ps.setInt(4, game.getStep());


            int affectedRows = ps.executeUpdate();
            ResultSet generatedKeys = ps.getGeneratedKeys();
            if (affectedRows == 1 && generatedKeys.next()) {
                game.setGameId(generatedKeys.getInt(1));
            }
            generatedKeys.close();

            createPlayersInDB(game);
            createCardFieldsInDB(game);
```

Den aktuelle spiller bliver sat til NULL for nu!

Se `getInsertGameStatementRGK()` for rækkefølgen (→ slide 68).

```
        ps = getSelectGameStatementU();
        ps.setInt(1, game.getGameId());

        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            rs.updateInt(GAME_CURRENTPLAYER,
              game.getPlayerNumber(game.getCurrentPlayer()));
            rs.updateRow();
        } else {
            // TODO error handling
        }
        rs.close();
        connection.commit();
        connection.setAutoCommit(true);
        return true;
    } catch (SQLException e) {
        // TODO error handling
        try {
            connection.rollback();
            connection.setAutoCommit(true);
        } catch (SQLException e1) {
            // TODO error handling
            e1.printStackTrace();
        }
    }
} else {
    System.err.println("Game cannot be created in DB, " +
      "since it has a game id already!");
```

Her bliver den aktuelle spiller endelig opdateret! Hvorfor først her?

Alle ændringer bliver først gennemført her ..

… eller hvis der slog noget fej rullet tilbage her!

```
private static final String SQL_INSERT_GAME =
      "INSERT INTO Game(name, currentPlayer, phase, step) " +
        "VALUES (?, ?, ?, ?)";

private PreparedStatement insert_game_stmt = null;

private PreparedStatement getInsertGameStatementRGK() {
   if (insert_game_stmt == null) {
      Connection connection = connector.getConnection();
      try {
         insert_game_stmt = connection.prepareStatement(
               SQL_INSERT_GAME,
               Statement.RETURN_GENERATED_KEYS);
      } catch (SQLException e) {
         // TODO error handling
         e.printStackTrace();
      }
   }
   return insert_game_stmt;
}
```

Da tabellen Game har automatisk genererede nøgler, er vi interesseret i at få dem og tilføje det til Board.

```
private static final String SQL_SELECT_GAME =
      "SELECT * FROM Game WHERE gameID = ?";

private PreparedStatement select_game_stmt = null;

private PreparedStatement getSelectGameStatementU() {
   if (select_game_stmt == null) {
      Connection connection = connector.getConnection();
      try {
         select_game_stmt = connection.prepareStatement(
               SQL_SELECT_GAME,
               ResultSet.TYPE_FORWARD_ONLY,
               ResultSet.CONCUR_UPDATABLE);
      } catch (SQLException e) {
         // TODO error handling
         e.printStackTrace();
      }
   }
   return select_game_stmt;
}
```

Bemærk at dette SELECT-statement bliver brugt til at opdatere game-tabellen!