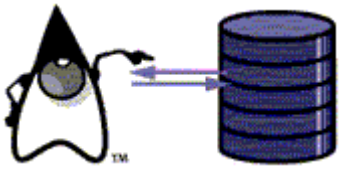
VDAB

samen sterk voor werk

Java

JDBC

Takenbundel



INHOUD

1 Taken

1.1 Bieren verwijderen

Je maakt een applicatie die in de database `bieren` alle bieren verwijdert
waarvan het alcohol% niet ingevuld is.

Je applicatie logt in met de username `cursist`.

1.2 Aantal bieren per brouwer

Je maakt een applicatie die een lijst toont.

De lijst bevat de brouwers die bieren hebben.

De lijst bevat per regel een brouwernaam en het aantal bieren van die brouwer.

Je sorteert de lijst op brouwernaam.

1.3 Bieren van tot alcohol

Je vraagt aan de gebruiker een minimum en een maximum alcohol%.

Je toont een lijst met de bieren waarvan het alcohol% ligt tussen dit minimum en maximum.

Je toont per bier het alcohol% en de naam.

Je sorteert de bieren op alcohol%. Je sorteert bieren met hetzelfde alcohol% op naam.

1.4 Bieren van tot alcohol 2

Je maakt een stored procedure met twee parameters: `minimumAlcohol` en `maximumAlcohol`.

De stored procedure zoekt alle bieren waarvan het alcohol% ligt tussen deze twee parameters.

De stored procedure geeft per bier de naam, de brouwernaam en het alcohol% terug.

Je sorteert de bieren op alcohol%. Je sorteert bieren met hetzelfde alcohol% op naam.

1.5 Failliet

Brouwer 1 gaat failliet.

Zijn bieren met een alcohol% vanaf 8.5 worden overgenomen door brouwer 2.
Zijn andere bieren worden overgenomen door brouwer 3.

Schrijf een applicatie die de nodige bewerkingen doet in de database. Op het einde verwijdert de applicatie brouwer 1. Doe al deze bewerkingen binnen één transactie.

1.6 Failliet 2

Gebruik in Failliet batch updates en het optimaalste transaction isolation level.

1.7 Bieren van een maand

Je laat de gebruiker een maandnummer (getal tussen 1 en 12) intikken.

Je toont de bieren die voor het eerst verkocht zijn in die maand (over alle jaren heen).

Je toont per bier de eigenschappen VerkochtSinds en Naam.

Je sorteert op VerkochtSinds.

1.8 Omzet niet gekend

Je toont per brouwer, waarvan de omzet niet gekend is, de naam en zijn aantal bieren.

1.9 Bieren van een soort

De gebruiker tikt de naam van een soort.
Jij toont de namen van de bieren van die soort, of een foutmelding als de soort niet bestaat.

1.10 Omzet leegmaken

Je laat de gebruiker brouwernummers intikken, tot hij 0 intikt.
Als hij daarbij een negatief nummer intikt, toon je een foutmelding.
Als hij daarbij een nummer intikt dat hij reeds ingetikt had, toon je een foutmelding.

Je maakt de omzet van alle geselecteerde brouwers leeg.

Als de gebruiker nummers intikte die niet bestaan in de database, toon je deze nummers.

2 Voorbeeldoplossingen

2.1 Bieren verwijderen

2.1.1 In de MySQL Workbench

```
grant all on bieren.* to cursist
```

2.1.2 De applicatie

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String SQL_DELETE =
    "delete from bieren where alcohol is null";
  public static void main(String[] args) {
    try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
         Statement statement = connection.createStatement()) {
      System.out.println(statement.executeUpdate(SQL_DELETE));
    } catch (SQLException ex) {
      ex.printStackTrace();
    }
  }
}
```

2.2 Aantal bieren per brouwer

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String SQL_SELECT =
    "select brouwers.naam,count(*) as aantal" +
    " from brouwers inner join bieren" +
    " on brouwers.id=bieren.brouwerid" +
    " group by brouwers.id, brouwers.naam" +
    " order by brouwers.naam";
  public static void main(String[] args) {
    try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
         Statement statement = connection.createStatement();
         ResultSet resultSet = statement.executeQuery(SQL_SELECT)) {
           // met left join lees je ook de brouwers die nul bieren hebben
      while (resultSet.next()) {
        System.out.printf("%s %d%n", resultSet.getString("naam"),
          resultSet.getInt("aantal"));
      }
    } catch (SQLException ex) {
      ex.printStackTrace();
   }
  }
}
```

2.3 Bieren van tot alcohol

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String SQL_SELECT =
    "select alcohol,naam from bieren where alcohol between ? and ?" +
    " order by alcohol, naam";
  public static void main(String[] args) {
    try (Scanner scanner = new Scanner(System.in)) {
      System.out.print("Minimum alcohol%:");
      BigDecimal minimum = scanner.nextBigDecimal();
      System.out.print("Maximum alcohol%:");
      BigDecimal maximum = scanner.nextBigDecimal();
      try (Connection connection = DriverManager.getConnection(URL,USER,PASSWORD);
           PreparedStatement statement =connection.prepareStatement(SQL_SELECT)) {
        statement.setBigDecimal(1, minimum);
        statement.setBigDecimal(2, maximum);
        try (ResultSet resultSet = statement.executeQuery()) {
          while (resultSet.next()) {
            System.out.printf("%5.2f %s%n", resultSet.getBigDecimal("alcohol"),
              resultSet.getString("naam"));
          }
        }
      } catch (SQLException ex) {
        ex.printStackTrace();
      }
    }
  }
}
```

2.4 Bieren van tot alcohol 2

2.4.1 De stored procedure

```
create procedure bieren.BierenVanTotAlcohol
  (minAlcohol decimal(10,2), maxAlcohol decimal(10,2))
begin
select bieren.naam as biernaam, brouwers.naam as brouwernaam, alcohol
from bieren inner join brouwers on bieren.brouwerid = brouwers.id
where alcohol between minAlcohol and maxAlcohol
order by alcohol, biernaam;
end
```

2.4.2 De applicatie

```
// enkele imports
class Main {
  private static final String URL =
    "jdbc:mysql://localhost/bieren?noAccessToProcedureBodies=true";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String CALL = "{call BierenVanTotAlcohol(?,?)}";
  public static void main(String[] args) {
    try (Scanner scanner = new Scanner(System.in)) {
      System.out.print("Minimum alcohol%:");
      BigDecimal minimum = scanner.nextBigDecimal();
```

```
      System.out.print("Maximum alcohol%:");
      BigDecimal maximum = scanner.nextBigDecimal();
      try (Connection connection = DriverManager.getConnection(URL,USER,PASSWORD);
           CallableStatement statement = connection.prepareCall(CALL)) {
        statement.setBigDecimal(1, minimum);
        statement.setBigDecimal(2, maximum);
        try (ResultSet resultSet = statement.executeQuery()) {
          while (resultSet.next()) {
            System.out.printf("%s%s%5.2f%n", resultSet.getString("biernaam"),
              resultSet.getString("brouwernaam") ,
              resultSet.getBigDecimal("alcohol"));
        }
      }
      } catch (SQLException ex) {
          ex.printStackTrace();
      }
    }
  }
}
```

2.5 Failliet

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String SQL_UPDATE1 =
    "update bieren set brouwerid=2 where brouwerid=1 and alcohol>=8.5";
  private static final String SQL_UPDATE2 =
    "update bieren set brouwerid=3 where brouwerid=1";
  private static final String SQL_DELETE = "delete from brouwers where id=1";
  public static void main(String[] args) {
    try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
         Statement statement = connection.createStatement()) {
      connection.setAutoCommit(false);
      statement.executeUpdate(SQL_UPDATE1);
      statement.executeUpdate(SQL_UPDATE2);
      statement.executeUpdate(SQL_DELETE);
      connection.commit();
    } catch (SQLException ex) {
      ex.printStackTrace();
    }
  }
}
```

2.6 Failliet 2

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String SQL_UPDATE1 =
    "update bieren set brouwerid=2 where brouwerid=1 and alcohol>=8.5";
  private static final String SQL_UPDATE2 =
    "update bieren set brouwerid=3 where brouwerid=1";
  private static final String SQL_DELETE = "delete from brouwers where id=1";
```

```
  public static void main(String[] args) {
    try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
         Statement statement = connection.createStatement()) {
      connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
      connection.setAutoCommit(false);
      statement.addBatch(SQL_UPDATE1);
      statement.addBatch(SQL_UPDATE2);
      statement.addBatch(SQL_DELETE);
      statement.executeBatch();
      connection.commit();
    } catch (SQLException ex) {
      ex.printStackTrace();
    }
  }
}
```

2.6.1 Bieren van een maand

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String SQL_SELECT =
    "select VerkochtSinds, Naam from bieren where" +
    " {fn month(VerkochtSinds)} = ? order by VerkochtSinds";
  public static void main(String[] args) {
    try (Scanner scanner = new Scanner(System.in)) {
      System.out.print("Maandnummer:");
      int maand = scanner.nextInt();
      if (maand >= 1 && maand <= 12) {
        try (Connection connection=DriverManager.getConnection(URL,USER,PASSWORD);
             PreparedStatement statement=connection.prepareStatement(SQL_SELECT)){
          statement.setInt(1, maand);
          try (ResultSet resultSet = statement.executeQuery()) {
            while (resultSet.next()) {
             System.out.printf("%s %s%n", resultSet.getDate("VerkochtSinds"),
               resultSet.getString("Naam"));
            }
          }
        } catch (SQLException ex) {
          ex.printStackTrace();
        }
      } else {
        System.out.println("Tik een getal tussen 1 en 12");
      }
    }
  }
}
```

2.7 Omzet niet gekend

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
```

```
  private static final String SELECT_SQL =
    "select brouwers.naam, count(*) as aantalbieren" +
    " from brouwers inner join bieren on brouwers.id = bieren.brouwerid" +
    " where omzet is null" +
    " group by brouwers.naam";
  public static void main(String[] args) {
    try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
         Statement statement = connection.createStatement();
         ResultSet resultSet = statement.executeQuery(SELECT_SQL)) {
      while (resultSet.next()) {
        System.out.printf("%s %d%n", resultSet.getString("naam"),
          resultSet.getInt("aantalbieren"));
      }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
  }
}
```

2.8 Bieren van een soort

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String SELECT_SOORT_SQL =
    "select id from soorten where naam = ?";
  private static final String SELECT_BIEREN_SQL =
    " select naam from bieren where soortid= ?";
  public static void main(String[] args) {
    try (Scanner scanner = new Scanner(System.in)) {
      System.out.print("soortnaam:");
      String soortNaam = scanner.nextLine();
      try (Connection connection = DriverManager.getConnection(URL,USER,PASSWORD);
           PreparedStatement statementSoort =
             connection.prepareStatement(SELECT_SOORT_SQL)) {
        statementSoort.setString(1, soortNaam);
        int soortId = 0;
        try (ResultSet resultSetSoort = statementSoort.executeQuery()) {
          if (resultSetSoort.next()) {
            soortId = resultSetSoort.getInt("id");
          }
        }
        if (soortId == 0) {
          System.out.println("Niet gevonden");
        } else {
          try (PreparedStatement statementBieren =
            connection.prepareStatement(SELECT_BIEREN_SQL)) {
            statementBieren.setInt(1, soortId);
            try (ResultSet resultSetBieren = statementBieren.executeQuery()) {
              while (resultSetBieren.next()) {
                System.out.println(resultSetBieren.getString("naam"));
              }
            }
          }
        }
```

```
      } catch (SQLException ex) {
        ex.printStackTrace();
      }
    }
  }
}
```

2.9 Omzet leegmaken

```
// enkele imports
class Main {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  private static final String UPDATE_SQL_BEGIN =
    "update brouwers set omzet = null where id in (";
  private static final String SELECT_SQL_BEGIN =
    " select id from brouwers where id in (";
  private static Set<Integer> vraagIds() {
    Set<Integer> ids = new LinkedHashSet<Integer>();
    try (Scanner scanner = new Scanner(System.in)) {
      System.out.print("Tik brouwernummers, stop met 0:");
      for (int id; (id = scanner.nextInt()) != 0;) {
        if (id < 0) {
          System.out.println("nummer moet positief zijn");
        } else {
          if ( ! ids.add(id)) {
            System.out.printf("%d reeds ingetikt%n", id);
          }
        }
      }
    }
  return ids;
  }
  private static int updateBrouwers(Set<Integer> ids, Connection connection)
    throws SQLException {
    int aantalAangepasteBrouwers = 0;
    StringBuffer updateSQL = new StringBuffer(UPDATE_SQL_BEGIN);
    for (int id : ids) {
      updateSQL.append("?,");
    }
    updateSQL.setCharAt(updateSQL.length() - 1, ')');
    try (PreparedStatement statementUpdate =
           connection.prepareStatement(updateSQL.toString())) {
      int index = 1;
      for (int id : ids) {
        statementUpdate.setInt(index++, id);
      }
      connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
      connection.setAutoCommit(false);
      aantalAangepasteBrouwers = statementUpdate.executeUpdate();
      connection.commit();
    }
  return aantalAangepasteBrouwers;
  }
```

```
  private static void toonNietGevondenIds(Set<Integer> ids, Connection connection)
    throws SQLException {
    StringBuffer selectSQL = new StringBuffer(SELECT_SQL_BEGIN);
    for (int id : ids) {
      selectSQL.append("?,");
    }
    selectSQL.setCharAt(selectSQL.length() - 1, ')');
    try (PreparedStatement statementSelect =
           connection.prepareStatement(selectSQL.toString())) {
      int index = 1;
      for (int id : ids) {
        statementSelect.setInt(index++, id);
      }
      try (ResultSet resultSet = statementSelect.executeQuery()) {
        while (resultSet.next()) {
          ids.remove(resultSet.getInt("id"));
        }
      }
      System.out.print("Volgende nummers werde niet gevonden:");
      for (int id : ids) {
        System.out.printf("%d ", id);
      }
      System.out.println();
    }
  }
  public static void main(String[] args) {
    Set<Integer> ids = vraagIds();
    if (ids.size() != 0) {
      try (Connection connection=DriverManager.getConnection(URL,USER,PASSWORD)) {
        if (updateBrouwers(ids, connection) != ids.size()) {
          toonNietGevondenIds(ids, connection);
        }
      }
      catch (SQLException ex) {
        ex.printStackTrace();
      }
    }
  }
}
```

COLOFON

Domeinexpertisemanager	Jean Smits
Moduleverantwoordelijke	
Auteurs	Hans Desmet
Versie	26/6/2015
Codes	Peoplesoftcode: Wettelijk depot:

Omschrijving module-inhoud

Abstract	Doelgroep	Opleiding Java Ontwikkelaar
	Aanpak	Zelfstudie
	Doelstelling	JDBC kunnen gebruiken
Trefwoorden		JDBC
Bronnen/meer info		