Opdracht 2 - Concurrency

Verslag

Analyse 8 - Advanced Databases (2014-2015)

Namen:	HoYe Lam, Rinesh Ramadhin
Studentnummer:	0876814, 0882447
Klas:	INF2D
Vak:	INFANL01-8
Opdracht:	Concurrency
Datum:	19 – 05 - 2015

Inhoud

Database

Script

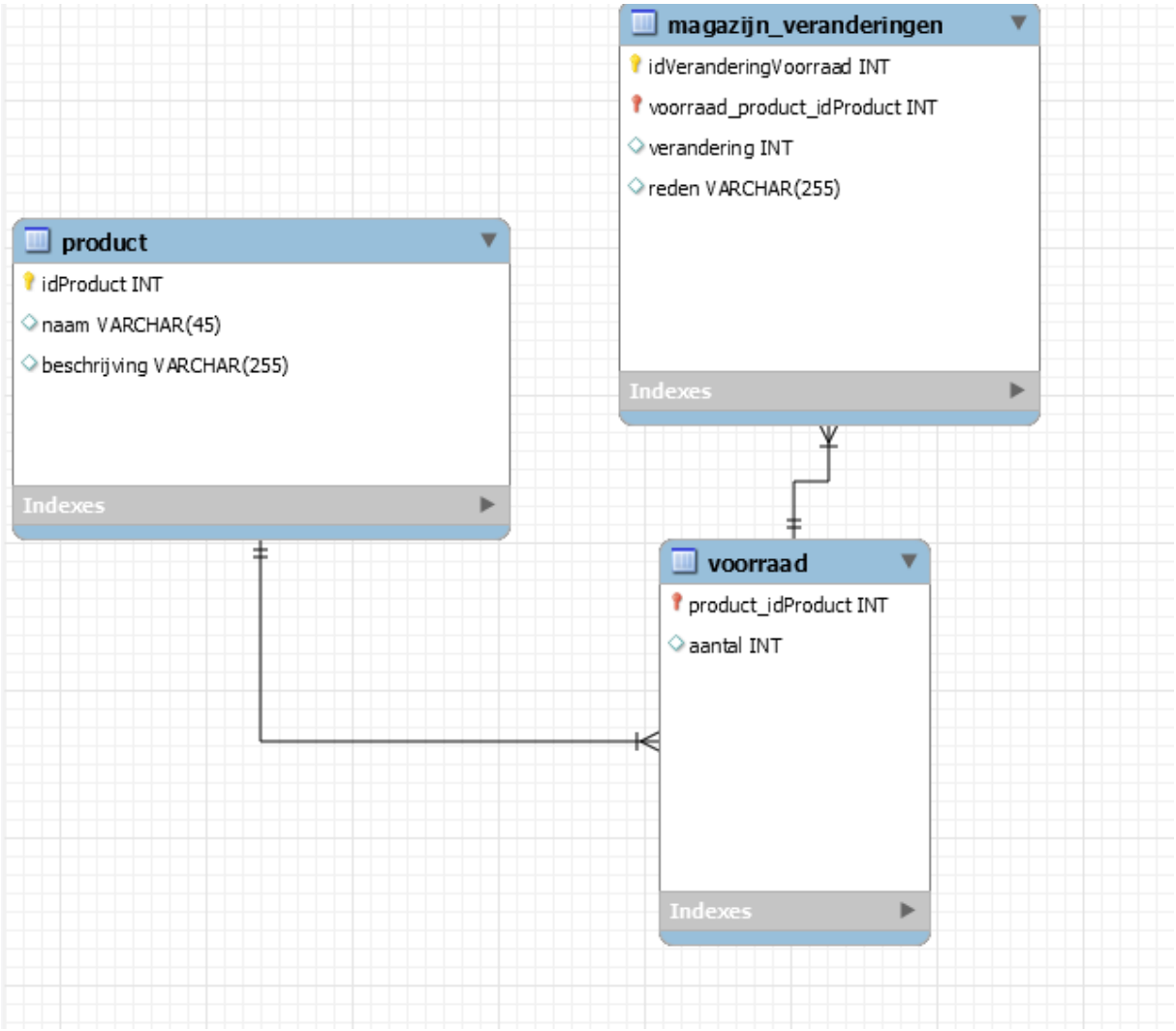
```
SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

DROP SCHEMA test;
CREATE SCHEMA test;

CREATE TABLE product(
        idProduct           INT           NOT NULL      PRIMARY KEY      AUTO_INCREMENT
    ,   naam                VARCHAR(45)   NOT NULL
    ,   beschrijving        VARCHAR(255)  NOT NULL
);

CREATE TABLE voorraad(
        product_idProduct   INT         NOT NULL        REFERENCES product(idProduct)
    ,   aantal              INT         NOT NULL
);

CREATE TABLE magazijn_veranderingen(
        idVeranderingVoorraad       INT             NOT NULL PRIMARY KEY   AUTO_INCREMENT
    ,    voorraad_product_idProduct INT             NOT NULL REFERENCES voorraad(product_idProduct)
    ,   verandering                 INT             NOT NULL
    ,   reden                       VARCHAR(45)     NOT NULL
);

DELIMITER //
CREATE TRIGGER before_magazijn_veranderingen_insert BEFORE INSERT ON magazijn_veranderingen
        FOR EACH ROW BEGIN
            UPDATE voorraad
            SET aantal = aantal + NEW.verandering
            WHERE product_idProduct = NEW.voorraad_product_idProduct;
            END;//
DELIMITER ;
```

Hier ziet dat wij de tabellen product, voorraad en magazijn_veranderingen creeeren. De kolommen van product en voorraad zijn standaard. Het product heeft een id, naam , beschrijving en aantal dat in de voorraad aanwezig is.

Bij magazijn_veranderingen hebben wij een aparte id voor elke verandering voor een verkocht/ingekocht bestelling. In het kolom ‘verandering’ zetten wij het aantal verkochte/inkgekochte product bijvoorbeeld +500.

Tot slot hebben wij nog een trigger gemaakt, waardoor als er in een verkopen/inkopen wordt gedaan, dan het aantal met dezelfde idProduct ook wordt toegenomen/afgenomen.

ERD



Na het bekijken van de opdracht hebben wij ervoor gekozen om de volgende tabellen te maken:

- Product
- Voorraad
- Magazijn_veranderingen

Hierbij ziet u dat product een voorraad heeft en dat voorraad meerdere producten bevat. Wij hebben voor dit gekozen, omdat onze winkel maar een voorraad heeft van ons eigen winkel. Dus er zijn geen andere voorraden zoals bij een leverancier.

Tot slot ziet u dat voorraad een magazijn veranderingen kan hebben en dat magazijn_veranderingen meerdere voorraden kan bezitten. Wij hebben voor dit gekozen, omdat de implementatie/realisatie makkelijker dan is dan een meer op meer relatie. Verder heeft deze beslissing geen invloed op het doel van de opdracht.

Java

Om de fenomenen te simuleren hebben we twee threads gebruikt. Deze staan voor gebruiker A en gebruiker B. In het begin van iedere thread wordt er een check uitgevoerd om te kijken of de connector aanwezig is. Hierna wordt er per thread een verbinding gemaakt met de MySQL database. Hierna wordt, waar nodig, de database geconfigureerd. Auto-commit wordt uitgezet en de isolatie leven wordt ingesteld op READ UNCOMMITED. Daarna volgen de acties die de gebruikers uit zullen voeren. Uiteindelijk wordt de connectie met de database gesloten.

In de onderstaande sub-hoofdstukken vindt U de code die we hebben geschreven om de fenomenen te stimuleren. Daaronder is beschreven wat de acties van de gebruikers waren.

Dirty Read

```
import java.sql.*;
import java.util.Date;

public class DirtyReadExample {
    public static void main(String[] args) {

        // Maak en start thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
                // maak verbinding met de driver
                try {
                    System.out.println("A: Loading driver...");
                    Class.forName("com.mysql.jdbc.Driver");
                    System.out.println("A: Driver loaded!");
                } catch (ClassNotFoundException e) {
                    throw new RuntimeException(
                            "Cannot find the driver in the classpath!", e);
                }

                // decladerdeer variabele
                String url = "jdbc:mysql://localhost:3306/test";
                String username = "hoye";
                String password = "hoye";

                //maak verbinding met de database
                Connection connection = null;
                try {
                    System.out.println("A: Connecting database...");
                    connection = DriverManager.getConnection(url, username, password);
                    System.out.println("A: Database connected!");
                    connection.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);

                    // set autocommit off
                    connection.setAutoCommit(false);

                    //Gebruiker A begint met een handeling

                    //sleep 4 sec
                    try {
                        Thread.sleep(4000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //retrieve aantal
                    Statement st = (Statement) connection.createStatement();
                    ResultSet x;

                    x = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                    int oud = 0;
                    if(x.next()){
                        oud = x.getInt("aantal");
                        System.out.println("A: Leest Aantal: " + oud);
                    }
```

```
                //sleep 4 sec
                try {
                    Thread.sleep(4000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                //update aantal (aantal+50)
                int nieuw = oud + 50;
                String QueryUpdate = ("UPDATE voorraad SET aantal = " + nieuw + " WHERE product_idProduct = 1");
                st.executeUpdate(QueryUpdate);
                System.out.println("A: Schrijft Aantal: " + nieuw);


                //commit
                connection.commit();
                System.out.println("A: Commit");

                //Sluit verbinding.
                connection.close();

            } catch (SQLException e) {
                throw new RuntimeException("Cannot connect the database!",
                        e);
            }
        }
    }, "Thread 1").start();
    // Maak en start thread 2. Deze draait tegelijkertijd met thread 1
    new Thread(new Runnable() {
        @Override
        public void run() {

            // maak verbinding met de driver
            try {
                System.out.println("B: Loading driver...");
                Class.forName("com.mysql.jdbc.Driver");
                System.out.println("B: Driver loaded!");
            } catch (ClassNotFoundException e) {
                throw new RuntimeException(
                        "Cannot find the driver in the classpath!", e);
            }

            // decladerdeer variabele
            String url = "jdbc:mysql://localhost:3306/test";
            String username = "hoye";
            String password = "hoye";

            //maak verbinding met de database
            Connection connection = null;
            try {
                System.out.println("B: Connecting database...");
                connection = DriverManager.getConnection(url, username, password);
                System.out.println("B: Database connected!");

                // set autocommit off
                connection.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);
                connection.setAutoCommit(false);

                //Gebruiker B begint met een handeling

                //UPDATE aantal
                Statement st = (Statement) connection.createStatement();

                ////
                ResultSet y;
                y = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                int oudy = 0;
                if(y.next()){
                    oudy = y.getInt("aantal");
                    System.out.println("B: templees = "+ oudy);
                }

                ////
                int nieuw =  125;
                String QueryUpdate = ("UPDATE voorraad SET aantal = " + nieuw + " WHERE product_idProduct = 1");
                st.executeUpdate(QueryUpdate);
                System.out.println("B: Schrijft Aantal: " + nieuw);

                ////
                ResultSet z;
                z = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                int oudz = 0;
                if(z.next()){
```

```
                    oudz = z.getInt("aantal");
                    System.out.println("B: templees = "+ oudz);
                }

                //Sleep 6
                try {
                    Thread.sleep(6000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                //ROLLBACK
                connection.rollback();

                //retrieve
                ResultSet x;

                x = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                int oud = 0;
                if(x.next()){
                    oud = x.getInt("aantal");
                    System.out.println("B: Rollback" + " Aantal = "+ oud);
                }

                //SLEEP 4
                try {
                    Thread.sleep(4000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                //Sluit verbinding.
                connection.close();

            } catch (SQLException e) {
                throw new RuntimeException("Cannot connect the database!",
                        e);
            }
        }
    }, "Thread 2").start();
  }
}
```

Gebruiker A

Gebruiker A begint met 4 seconden slapen. Hierna haalt het een waarde op. Wij hebben gekozen voor het aantal items op voorraad voor het product met product_id 1. Deze waarde wordt opgeslagen in een integer. Hierna slaapt het weer 4 seconden. Hierna wordt aan de opgeslagen integer 50 toegevoegd. Deze nieuwe waarde wordt dan naar de database gestuurd om het aantal items te updaten. Uiteindelijk wordt alles gecommit.

Gebruiker B

Gebruiker B begint met het updaten van het aantal items op voorraad voor het product met product_id 1. Dit wordt ingesteld op 125. Hierna slaapt het voor 6 seconden. Hierna wordt er een rollback gedaan. Hierna wordt het aantal items op voorraad voor het product met product_id 1 opgehaald. Hierna slaapt het voor 4 seconden.

Non-repeatable/Unrepeatable Read

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class NonRepeatableRead {
public static void main(String[] args) {

        // Maak en start thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
                // maak verbinding met de driver
                try {
                    System.out.println("A: Loading driver...");
                    Class.forName("com.mysql.jdbc.Driver");
                    System.out.println("A: Driver loaded!");
                } catch (ClassNotFoundException e) {
                    throw new RuntimeException(
                            "Cannot find the driver in the classpath!", e);
                }

                // decladerdeer variabele
                String url = "jdbc:mysql://localhost:3306/test";
                String username = "hoye";
                String password = "hoye";

                //maak verbinding met de database
                Connection connection = null;
                try {
                    System.out.println("A: Connecting database...");
                    connection = DriverManager.getConnection(url, username,
                            password);
                    System.out.println("A: Database connected!");
                    connection.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);

                    // set autocommit off
                    connection.setAutoCommit(false);

                    //Gebruiker A begint met een handeling

                    //Retrieve aantal
                    Statement st = (Statement) connection.createStatement();
                    ResultSet x;

                    x = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                    int oud = 0;
                    if(x.next()){
                        oud = x.getInt("aantal");
                        System.out.println("A: Leest Aantal: " + oud);
                    }
                    //SLEEP 4 sec
                    try {
                        Thread.sleep(4000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //Retrieve aantal
                    ResultSet y;

                    y = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                    int oudy = 0;
                    if(y.next()){
                        oudy = y.getInt("aantal");
                        System.out.println("A: Leest Aantal: " + oudy);
                    }
                    //SLEEP 4 sec
                    try {
                        Thread.sleep(4000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //commit
                    connection.commit();
```

```
                System.out.println("A: Commit");

                //Sluit verbinding.
                connection.close();

            } catch (SQLException e) {
                throw new RuntimeException("Cannot connect the database!",
                        e);
            }
        }
    }, "Thread 1").start();
    // Maak en start thread 2. Deze draait tegelijkertijd met thread 1
    new Thread(new Runnable() {
        @Override
        public void run() {

            // maak verbinding met de driver
            try {
                System.out.println("B: Loading driver...");
                Class.forName("com.mysql.jdbc.Driver");
                System.out.println("B: Driver loaded!");
            } catch (ClassNotFoundException e) {
                throw new RuntimeException(
                        "Cannot find the driver in the classpath!", e);
            }

            // decladerdeer variabele
            String url = "jdbc:mysql://localhost:3306/test";
            String username = "hoye";
            String password = "hoye";

            //maak verbinding met de database
            Connection connection = null;
            try {
                System.out.println("B: Connecting database...");
                connection = DriverManager.getConnection(url, username,
                        password);
                System.out.println("B: Database connected!");

                // set autocommit off
                connection.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);
                connection.setAutoCommit(false);

                //Gebruiker B begint met een handeling
                Statement st = (Statement) connection.createStatement();

                //SLEEP 2 SEC
                try {
                    Thread.sleep(2000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                //UPDATE aantal
                int nieuw =  125;
                String QueryUpdate = ("UPDATE voorraad SET aantal = " + nieuw + " WHERE product_idProduct = 1");
                st.executeUpdate(QueryUpdate);
                System.out.println("B: Schrijft Aantal: " + nieuw);

                //SLEEP 6 SEC
                try {
                    Thread.sleep(6000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                //COMMIT
                connection.commit();
                System.out.println("B: Commit");

                //SLEEP 2 SEC
                try {
                    Thread.sleep(2000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                //Sluit verbinding.
                connection.close();

            } catch (SQLException e) {
                throw new RuntimeException("Cannot connect the database!",
                        e);
            }
        }
```

```
        }, "Thread 2").start();
    }
}
```

Gebruiker A

Gebruiker A begint met het ophalen van het aantal items op voorraad voor het product met product_id 1. Hierna slaapt het voor 4 seconden. Hierna wordt nogmaals het aantal items op voorraad voor het product met product_id 1 opgehaald. Nogmaals slaapt het voor 4 seconden. Uiteindelijk wordt er een commit gedaan.

Gebruiker B

Gebruiker B begint met het slapen van 2 seconden. Hierna update het de waarde van het aantal items op voorraad voor het product met product_id 1. Dit wordt geüpdatet naar 125. Hierna slaapt het voor 6 seconden. Dan wordt er een commit gedaan. Hierna slaapt het nogmaals voor twee seconden.

Phantom Read

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class PhantomRead {
public static void main(String[] args) {

        // Maak en start thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
                // maak verbinding met de driver
                try {
                    System.out.println("A: Loading driver...");
                    Class.forName("com.mysql.jdbc.Driver");
                    System.out.println("A: Driver loaded!");
                } catch (ClassNotFoundException e) {
                    throw new RuntimeException(
                            "Cannot find the driver in the classpath!", e);
                }

                // decladerdeer variabele
                String url = "jdbc:mysql://localhost:3306/test";
                String username = "hoye";
                String password = "hoye";

                //maak verbinding met de database
                Connection connection = null;
                try {
                    System.out.println("A: Connecting database...");
                    connection = DriverManager.getConnection(url, username,
                            password);
                    System.out.println("A: Database connected!");
                    connection.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);

                    // set autocommit off
                    connection.setAutoCommit(false);

                    //Gebruiker A begint met een handeling

                    // lees tabel
                    String query = "SELECT * FROM voorraad";
                    Statement st = connection.createStatement();
                    ResultSet rs = st.executeQuery(query);

                    while(rs.next())
                    {
                        int product_id = rs.getInt("product_idProduct");
                        int aantalintabel = rs.getInt("aantal");
```

```
                System.out.format("%s, %s\n", product_id, aantalintabel);
            }

            // sleep 4sec
            try {
                Thread.sleep(4000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            //lees tabel
            ResultSet rs2 = st.executeQuery(query);

            while(rs2.next())
            {
                int product_id = rs2.getInt("product_idProduct");
                int aantalintabel = rs2.getInt("aantal");

                System.out.format("%s, %s\n", product_id, aantalintabel);
            }
            // sleep 4sec
            try {
                Thread.sleep(4000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            // commit
            connection.commit();
            System.out.println("A: Commit");

            //Sluit verbinding.
            connection.close();

        } catch (SQLException e) {
            throw new RuntimeException("Cannot connect the database!",
                    e);
        }
    }
}, "Thread 1").start();
// Maak en start thread 2. Deze draait tegelijkertijd met thread 1
new Thread(new Runnable() {
    @Override
    public void run() {

        // maak verbinding met de driver
        try {
            System.out.println("B: Loading driver...");
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("B: Driver loaded!");
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(
                    "Cannot find the driver in the classpath!", e);
        }

        // decladerdeer variabeleS
        String url = "jdbc:mysql://localhost:3306/test";
        String username = "hoye";
        String password = "hoye";

        //maak verbinding met de database
        Connection connection = null;
        try {
            System.out.println("B: Connecting database...");
            connection = DriverManager.getConnection(url, username,
                    password);
            System.out.println("B: Database connected!");

            // set autocommit off
            connection.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);
            connection.setAutoCommit(false);

            //Gebruiker B begint met een handeling
            Statement st = (Statement) connection.createStatement();

            // sleep 2sec
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
```

```
                    // insert nieuwe regel in tabel
                    st.executeUpdate("INSERT INTO product " + "VALUES (11, 'Samsung S6', 'Telefoon')");
                    st.executeUpdate("INSERT INTO voorraad " + "VALUES (11, 50)");

                    // commit
                    connection.commit();
                    System.out.println("B: Commit");
                    //sleep 8sec
                    try {
                        Thread.sleep(8000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //Sluit verbinding.
                    connection.close();

                } catch (SQLException e) {
                    throw new RuntimeException("Cannot connect the database!",
                            e);
                }
            }
        }, "Thread 2").start();
    }
}
```

Gebruiker A

Gebruiker A begint met het lezen en weergeven van de inhoud van de voorraad tabel. Hierna slaapt het voor 4 seconden. Hierna leest en weergeeft het nogmaals de inhoud van de voorraad tabel. Hierna slaapt het nogmaals voor 4 seconden. Hierna commit het.

Gebruiker B

Gebruiker B begint met het slapen van 2 seconden. Hierna insert het een nieuwe regel in de voorraad tabel. Vanwege de opbouw van de database moet er ook eerst een insert worden gedaan in de product tabel. Hierna wordt er een commit gedaan. Uiteindelijk slaapt de thread voor 8 seconden.

Deadlock

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class DeadLock {
public static void main(String[] args) {

        // Maak en start thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
                // maak verbinding met de driver
                try {
                    System.out.println("A: Loading driver...");
                    Class.forName("com.mysql.jdbc.Driver");
                    System.out.println("A: Driver loaded!");
                } catch (ClassNotFoundException e) {
                    throw new RuntimeException(
                            "Cannot find the driver in the classpath!", e);
                }

                // decladerdeer variabele
                String url = "jdbc:mysql://localhost:3306/test";
                String username = "hoye";
                String password = "hoye";

                //maak verbinding met de database
                Connection connection = null;
                try {
                    System.out.println("A: Connecting database...");
                    connection = DriverManager.getConnection(url, username,
                            password);
                    System.out.println("A: Database connected!");
                    connection.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);

                    // set autocommit off
                    connection.setAutoCommit(false);

                    //Gebruiker A begint met een handeling

                    //RETRIEVE AANTAL
                    Statement st = (Statement) connection.createStatement();
                    ResultSet x;

                    x = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                    int oud = 0;
                    if(x.next()){
                        oud = x.getInt("aantal");
                        System.out.println("A: Leest Aantal: " + oud);
                    }
                    //SLEEP 4 SEC
                    try {
                        Thread.sleep(4000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //UPDATE AANTAL

                    int nieuw =  75;
                    String QueryUpdate = ("UPDATE voorraad SET aantal = " + nieuw + " WHERE product_idProduct = 1");
                    st.executeUpdate(QueryUpdate);
                    System.out.println("A: Schrijft Aantal: " + nieuw);


                } catch (SQLException e) {
                    throw new RuntimeException("Cannot connect the database!",
                            e);
                }
            }
        }, "Thread 1").start();
        // Maak en start thread 2. Deze draait tegelijkertijd met thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
```

```
                // maak verbinding met de driver
                try {
                    System.out.println("B: Loading driver...");
                    Class.forName("com.mysql.jdbc.Driver");
                    System.out.println("B: Driver loaded!");
                } catch (ClassNotFoundException e) {
                    throw new RuntimeException(
                            "Cannot find the driver in the classpath!", e);
                }

                // decladerdeer variabele
                String url = "jdbc:mysql://localhost:3306/test";
                String username = "hoye";
                String password = "hoye";

                //maak verbinding met de database
                Connection connection = null;
                try {



                    System.out.println("B: Connecting database...");
                    connection = DriverManager.getConnection(url, username,
                            password);
                    System.out.println("B: Database connected!");

                    // set autocommit off
                    connection.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);
                    connection.setAutoCommit(false);


                    //Gebruiker B begint met een handeling
                    Statement st = (Statement) connection.createStatement();

                    //SLEEP 2 SEC
                    try {
                        Thread.sleep(2000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //RETRIEVE AANTAL
                    ResultSet x;

                    x = st.executeQuery("SELECT aantal FROM voorraad WHERE product_idProduct = 1");
                    int oud = 0;
                    if(x.next()){
                        oud = x.getInt("aantal");
                        System.out.println("B: Leest Aantal: " + oud);
                    }

                    //SLEEP 6
                    try {
                        Thread.sleep(6000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //UPDATE AANTAL

                    int nieuw =  200;
                    String QueryUpdate = ("UPDATE voorraad SET aantal = " + nieuw + " WHERE product_idProduct = 1");
                    st.executeUpdate(QueryUpdate);
                    System.out.println("B: Schrijft Aantal: " + nieuw);


                } catch (SQLException e) {
                    System.out.println(e);
                    throw new RuntimeException("Cannot connect the database!",
                            e);
                }
            }
        }, "Thread 2").start();
    }
}
```

Gebruiker A

In het begin van de thread wordt het isolatie level ingesteld op SERIALIZABLE in plaats van READ UNCOMMITED. Gebruiker A begint met het ophalen van het aantal items op voorraad voor het product met product_id 1. Hierna slaapt het voor 4 seconden. Hierna (probeert) het deze waarde te updaten. Op het einde van deze thread wordt de connectie met de database NIET gesloten.

Gebruiker B

In het begin van de thread wordt het isolatie level ingesteld op SERIALIZABLE in plaats van READ UNCOMMITED. Gebruiker B begint met het slapen van 2 seconden. Hierna wordt het het aantal items op voorraad voor het product met product_id 1 opgehaald. Hierna slaapt het voor 6 seconden. Hier (probeert) het het aantal items op voorraad voor het product met product_id 1 te updaten. Op het einde van deze thread wordt de connectie met de database NIET gesloten.

De (verwachte) verkregen error vindt U hieronder: