

Programming Advanced java

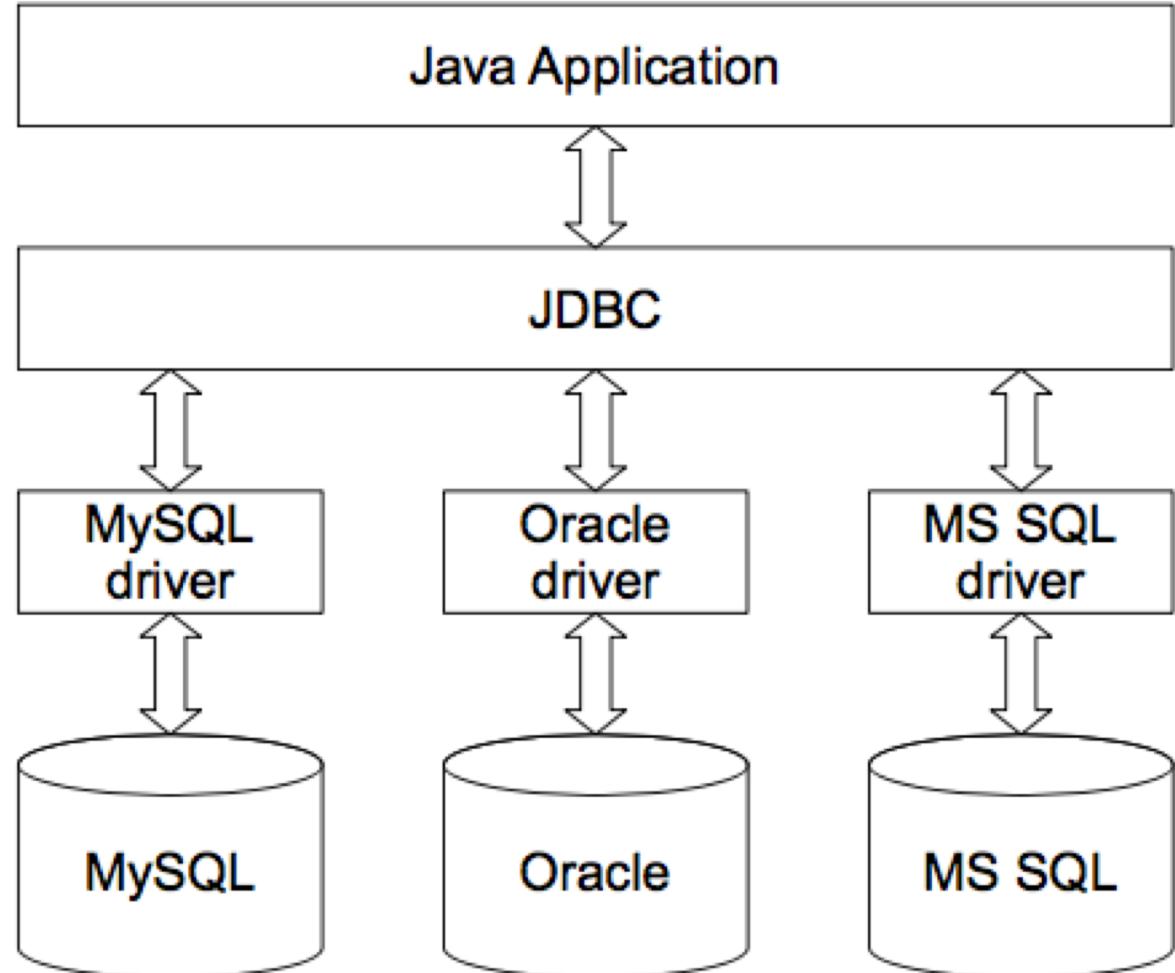
JDBC



Overzicht

1. Inleiding
 2. Database verbinding
 3. SQL commando's
 4. Transacties
-
- Oefeningen
 - **zonder Main**
 - met unit tests

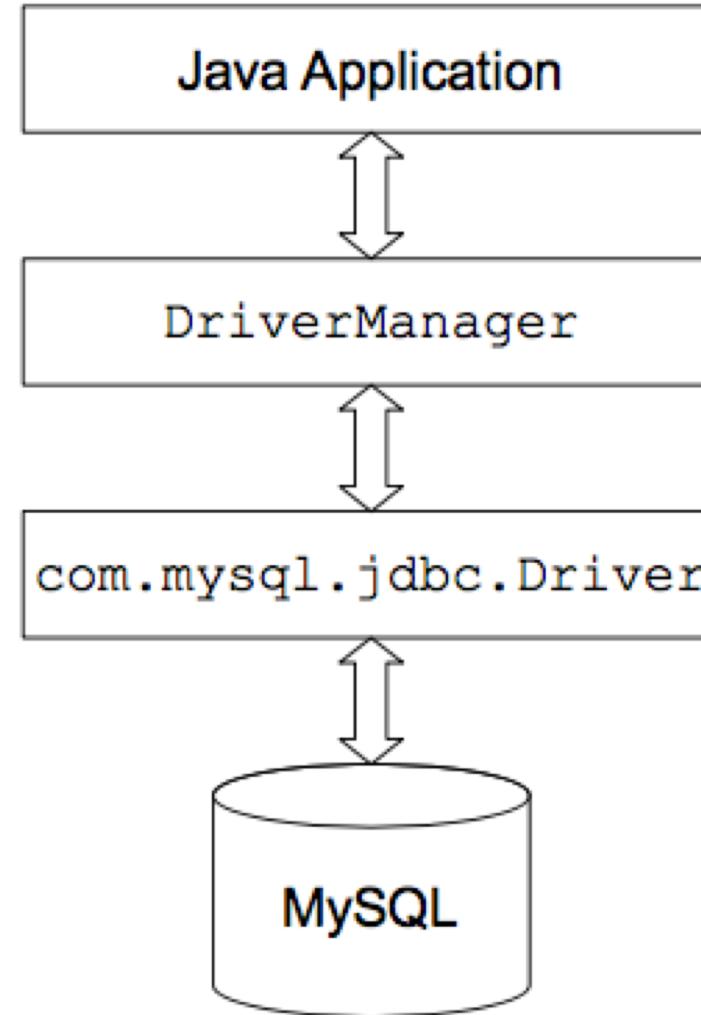
Inleiding



packages:

- `java.sql`: Classes voor standaard sql functionaliteit
 - `DriverManager`
 - `Connection`
 - `Statement`
 - ...
- `javax.sql`: Classes voor geavanceerde sql functionaliteit
 - `DataSource`
 - `PooledConnection`
 - `ConnectionEvent`
 - `StatementEvent`
 - ...

Databaseserver connectie



Databaseserver connectie

- Database-driver laden via `Class.forName("com.mysql.jdbc.driver")`
- Driver moet beschikbaar zijn op Classpath
- Maven dependency

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.41</version>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

- Manuele download: <https://dev.mysql.com/downloads/connector/j/>

Databaseserver connectie

- `DriverManager.getConnection(url, login, password)`
 - url = jdbc:subprotocol:subname
 - vb. jdbc:mysql://localhost/databasename
- Connection is AutoCloseable → try with resources
- Opdracht 1 p67

SQL-commando's - Statement

- Statement stmt = connection.createStatement();
 - Statement is Autoclosable → try with resources

Methode	Omschrijving
executeUpdate (String sql)	Voert een CREATE, INSERT, UPDATE of DELETE commando uit.
executeQuery (String sql)	Voert een commando uit dat als resultaat een tabel heeft (<i>resultset</i>). Dit is typisch het geval bij een SELECT-commando.
executeBatch ()	Voert een reeks van UPDATE-commando's uit. De commando's worden toegevoegd met de methode addBatch () .
execute ()	Voert om het even welk SQL-commando uit.

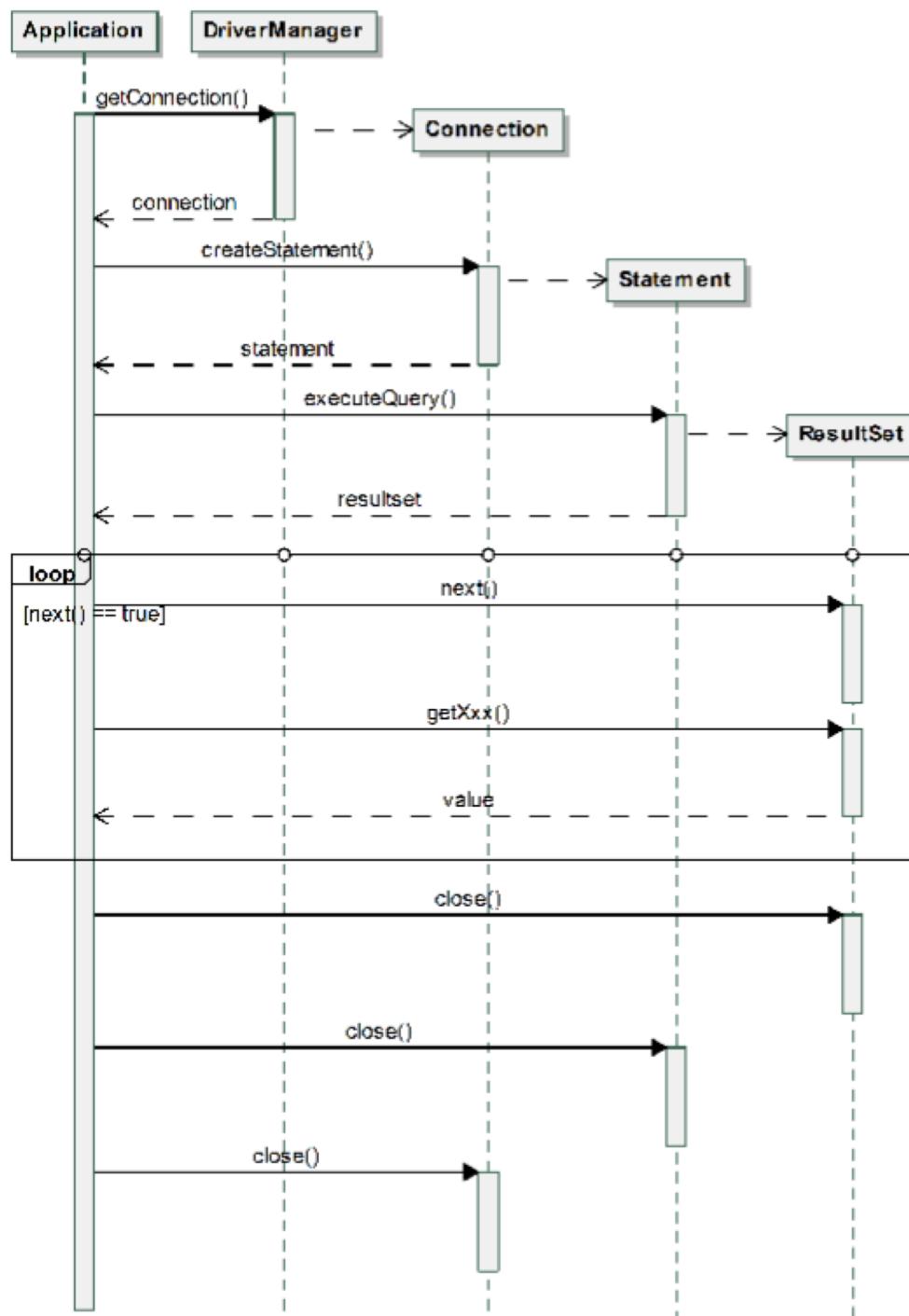
SQL-commando's - ResultSet

Methode	Omschrijving
next()	Zet de cursor één positie verder.
first()	Zet de cursor op de eerste rij.
last()	Zet de cursor op de laatste rij.
previous()	Zet de cursor één positie terug.
afterLast()	Zet de cursor net na de laatste rij.
beforeFirst()	Zet de cursor net voor de eerste rij.
absolute()	Zet de cursor op een bepaalde rij.
relative()	Zet de cursor een aantal rijen vooruit of achteruit ten opzichte van de huidige positie.

```
while(rs.next()) {  
    // get by index (starting from 1)  
    // get by column name  
}
```

SQL-commando's - ResultSet

	TINYINT	SMALLINT	INTEGER	BIGINT	FLOAT	DOUBLE	DECIMAL	NUMERIC	BIT	CHAR	VARCHAR	LONGVARCHAR	BINARY	VARBINARY	LONGVARBINARY	DATE	TIME	TIMESTAMP	CLOB	BLOB	ARRAY	REF	STRUCT	JAVA OBJECT
getByte()	x	x	x	x	x	x	x	x	x	x	x	x												
getArray()	x	x	x	x	x	x	x	x	x	x	x	x												
getShort()	x	x	x	x	x	x	x	x	x	x	x	x												
getInt()	x	x	x	x	x	x	x	x	x	x	x	x												
getLong()	x	x	x	x	x	x	x	x	x	x	x	x												
getFloat()	x	x	x	x	x	x	x	x	x	x	x	x												
getDouble()	x	x	x	x	x	x	x	x	x	x	x	x												
getBigDecimal()	x	x	x	x	x	x	x	x	x	x	x	x												
getBoolean()	x	x	x	x	x	x	x	x	x	x	x	x												
getString()	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
getBytes()													x	x	x									
getDate()										x	x	x				x	x							
getTime()										x	x	x						x	x					
getTimestamp()										x	x	x				x	x	x						
getAsciiStream()										x	x	x	x	x	x									
getBinaryStream()													x	x	x									
getCharacterStream()									x	x	x	x	x	x										
getClob()																x								
getBlob()																		x						



SQL-commando's – Prepared Statement

Voordelen t.o.v. Statement

- Sneller -> precompiled
 - Kan meerdere keren gebruikt worden zonder prestatie verlies
- SQL Injection proof
- Samenstellen met objects en parameters i.p.v. string concatenatie
 - Clean code
 - Minder foutgevoelig

SQL-commando's – Prepared Statement

- interface java.sql.PreparedStatement
- Connection.prepareStatement(java.lang.String)
- Voorbeeld

```
String sql = "UPDATE Beers SET Price = ? WHERE Name = ?";  
try (Connection con = DriverManager.getConnection(  
    "jdbc:mysql://noelvaes.eu/StudentDB", "xxxx", "xxxx");  
    PreparedStatement stmt = con.prepareStatement(sql)) {  
    stmt.setFloat(1, 3.5F);  
    stmt.setString(2, "Zulte");  
    int result = stmt.executeUpdate();  
}
```

Transacties

- Standaard is ieder sql commando een aparte transactie
 - autoCommit=true
- Uitvoeren van verschillende sql commando's als geheel
- Annuleer alle commando's binnen hetzelfde geheel (Rollback)

Transacties

```
try (Connection con = DriverManager.getConnection("url",
                                              "login","password")) {
    try (Statement stmt = con.createStatement()) {
        con.setAutoCommit(false);
        stmt.executeUpdate(SQL1);      // Transaction start
        stmt.executeUpdate(SQL2);
        con.commit();                  // Transaction commit
    } catch (Exception e) {
        con.rollback();                // Transaction rollback
    }
}
```

Transacties

Methode	Omschrijving
<code>setAutoCommit (false)</code>	Schakelt transacties in.
<code>executeUpdate ()</code> <code>executeQuery ()</code>	SQL-commando's worden opgestapeld.
<code>setSavePoint ()</code>	Stelt een <i>savepoint</i> in.
<code>rollback ()</code>	Opgestapelde commando's worden geannuleerd.
<code>commit ()</code>	Opgestapelde commando's worden als geheel definitief gemaakt.
<code>setAutoCommit (true)</code>	Schakelt transacties opnieuw uit.

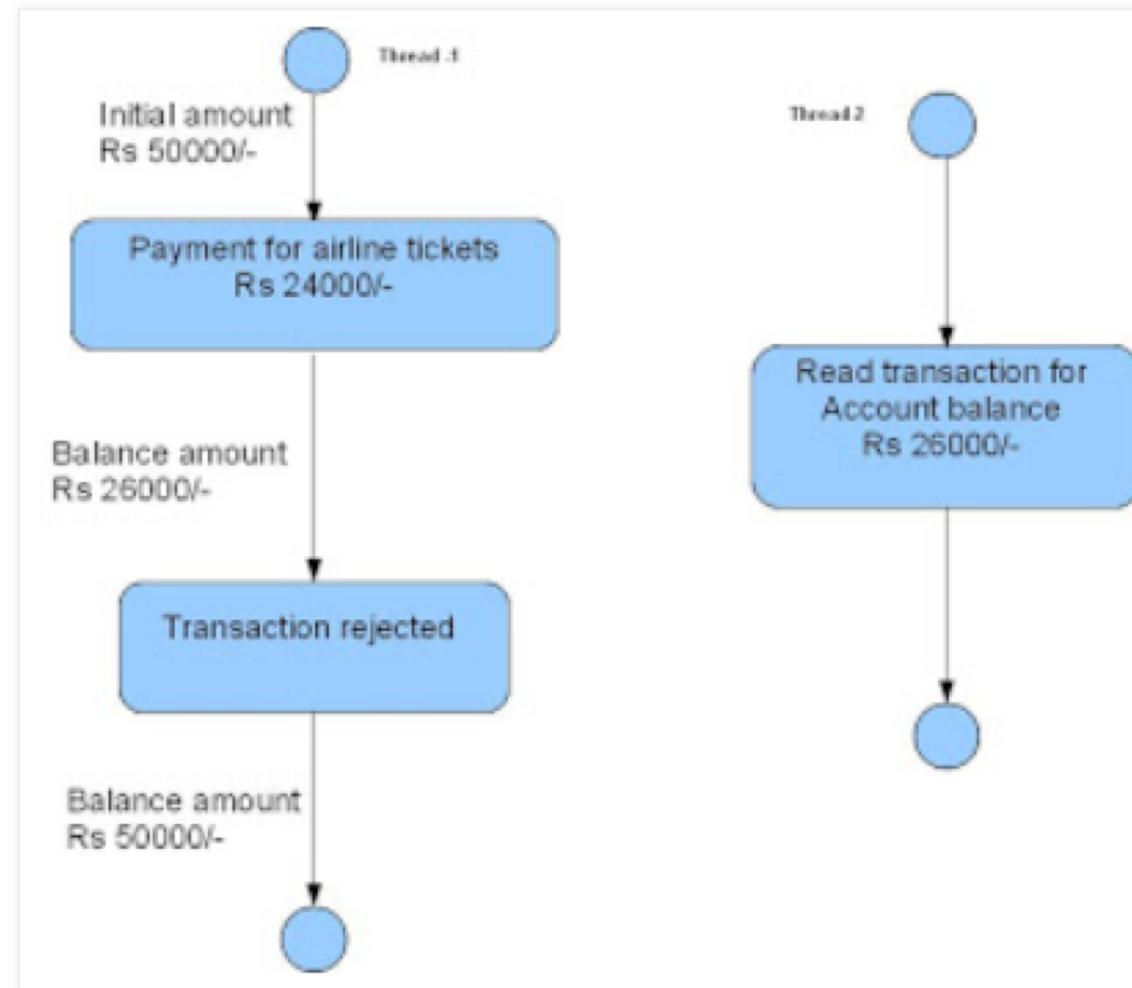
savepoints:

- binnen een transactie
- rollback tot savepoint: `rollback (SavePoint sp)`

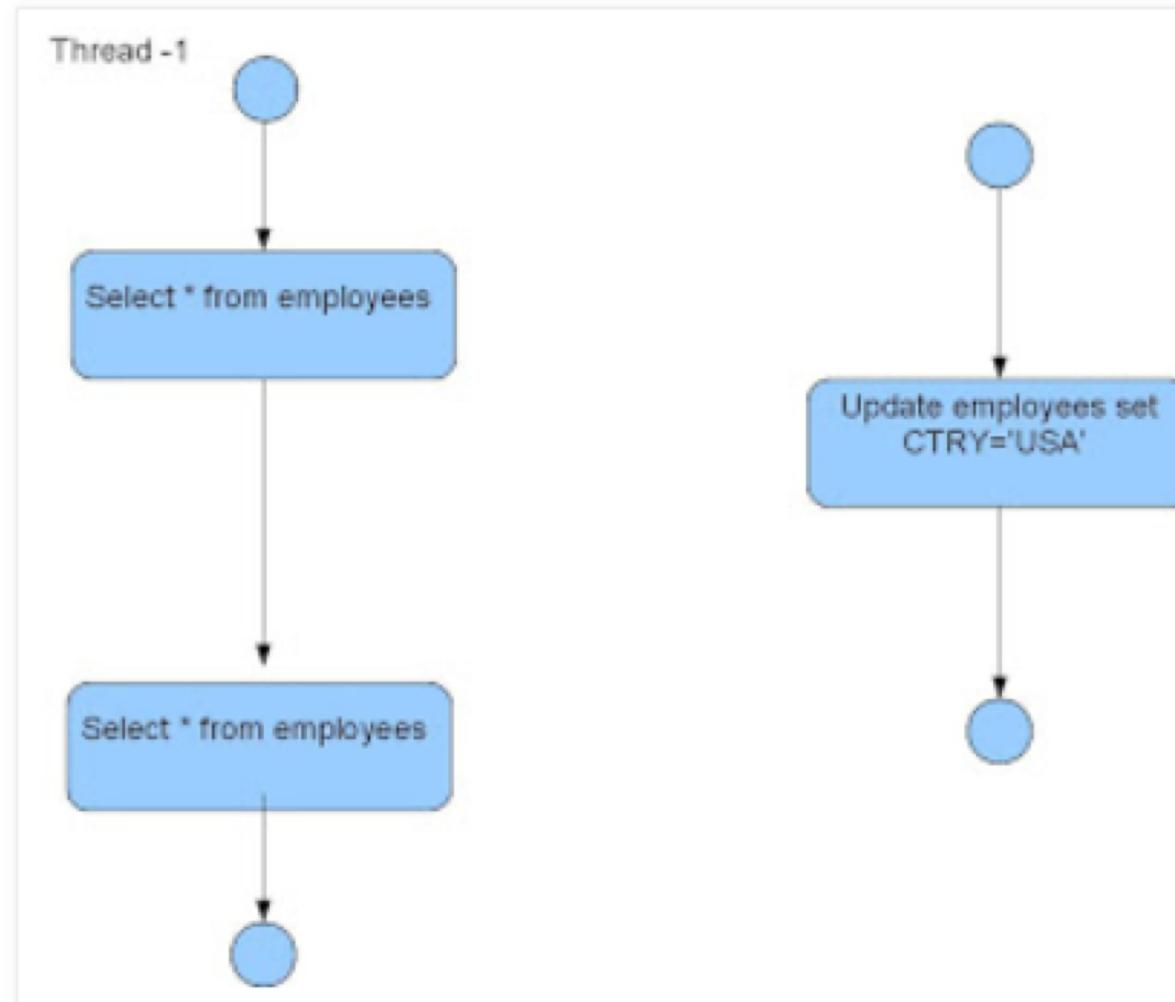
Transacties – isolation level

Dirty read	Transactie leest gegevens die nog niet gecommit zijn door andere transactie
Non-repeatable read	Transactie voert een query 2 keer uit. De resultset bevat andere waarden. -> UPDATE uit andere transactie
Phantom read	Transactie voert een query 2 keer uit. De resultset verschilt. -> INSERT of DELETE uit andere transactie

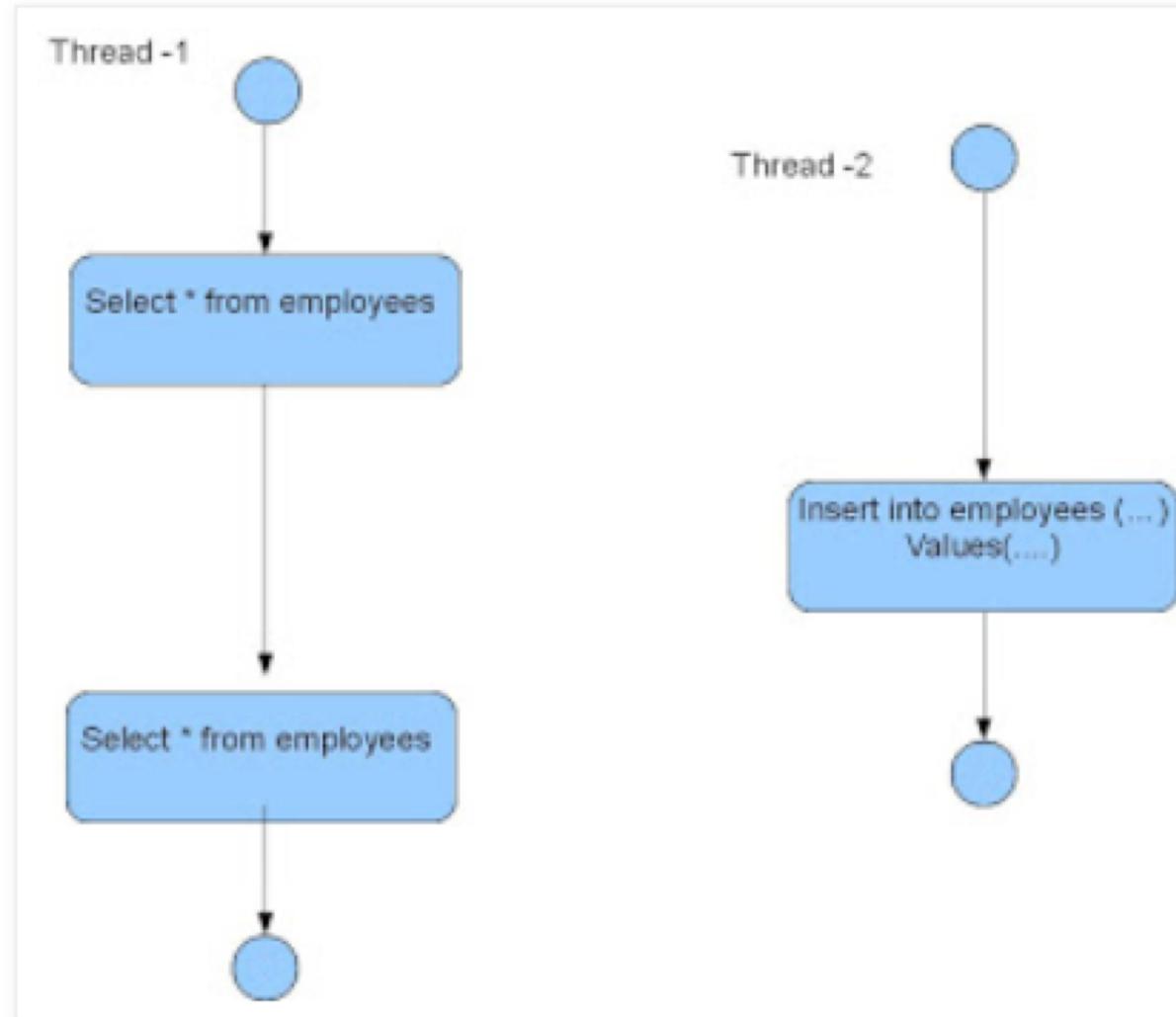
Transacties – Dirty Read



Transacties – Non Repeatable Read



Transacties – Phantom Read



Transacties – isolation level

<i>Level</i>	<i>Dirty read</i>	<i>Non-repeatable read</i>	<i>Phantom read</i>
TRANSACTION_NONE	Geen transacties		
TRANSACTION_READ_UNCOMMITTED	X	X	X
TRANSACTION_READ_COMMITTED	O	X	X
TRANSACTION_REPEATABLE_READ	O	O	X
TRANSACTION_SERIALIZABLE	O	O	O

Hogere Transaction Level:

- Hogere data consistentie
- Lagere performantie

In memory database

- H2 database : <http://www.h2database.com>
- JDBC url
 - jdbc:h2:mem
- JDBC url voor test doeleinden op basis van test data
 - jdbc:h2:mem:test;INIT=RUNSCRIPT FROM 'classpath:StudentDB.sql'