1 - TINA UMLANDT. 2015

WIEDERHOLUNG

TRANSAKTION

3 - TINA UMLANDT, 2015

ACID

TRANSAKTIONSVERWALTUNG

- KONSISTENZSICHERUNG
- RECOVERY
- MEHRBENUTZERSYNCHRONISATION

ANOMALIEN BEI UNKONTROLLIERTEM MEHRBENUTZERBETRIEB

LOST UPDATE

Zeit	Transaktion 1	Transaktion 2
1	Lese Betrag	-
2	-	Lese Betrag
3	Erhöhe Betrag um 1	-
4	-	Erhöhe Betrag um 1
5	Commit	-
6	-	Commit

UNCOMMITTED DEPENDENCY (DIRTY READ)

Zeit	Transaktion 1	Transaktion 2
1	Lese Betrag	-
2	Erhöhe Betrag um 1	-
3	-	Lese Betrag
4	Rollback	-

INCONSISTENCY READ (NONREPEATABLE READ)

Zeit	Transaktion 1	Transaktion 2
1	Lese Haben_Konto	-
2	-	Ändere Soll_Konto
3	-	Ändere Haben_Konto
4	-	Commit
5	Lese Soll_Konto	-
6	Commit	-

PHANTOMDATEN

Zeit	Transaktion 1	Transaktion 2
1	Lese Kontostand	-
2	-	Ändere Kontostand
4	-	Commit
5	Lese Kontostand	-
6	Commit	-

LOCKS

12 - TINA UMLANDT, 2015

READ LOCKS (SHARED)

WRITE LOCKS (EXCLUSIVE)

ZWEI-PHASEN-SPERRPROTOKOLL

15 - TINA UMLANDT, 2015

LOS GEHT'S

\o/

JDBC

AUFGABEN

- DATENBANKVERBINDUNGEN AUFBAUEN UND VERWALTEN.
- SQL-ANFRAGEN AN DIE DATENBANK WEITERLEITEN UND DIE
- ERGEBNISSE IN EINE FÜR JAVA NUTZBARE FORM UMZUWANDELN

JDBC IST TEIL DES JDK

`java.sql` **UND** `javax.sql`

JEDE DATENBANK HAT EIGENE TREIBER

```
package java.sql;

/**
 * The interface that every driver class must implement.
 *
 * ...
 *
 */
public interface Driver {
...
```

```
public class oracle.jdbc.driver.OracleDriver implements java.sql.Driver {
    ...
}
```

```
package java.sql;


/**
 * <P>The basic service for managing a set of JDBC drivers.<br>
 *
 * ...
 *
 */
public class DriverManager {
```

```
final Connection connection = DriverManager.getConnection(
                    "jdbc:oracle:thin:@ora14.informatik.haw-hamburg.de:1521:inf14", username, password);
```

java.sql.Connection

```
connection.setAutoCommit(autoCommit);
```

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

```
connection.rollback();
```

```
connection.setReadOnly(readOnly);
```

```
connection.setTransactionIsolation(level);
```

```
connection.setSavepoint(name)
connection.rollback(savepoint);
connection.releaseSavepoint(savepoint);
```

```
Statement createStatement();
```

java.sql.Statement

```
connection.createStatement().executeQuery(sql);
```

```
connection.createStatement().executeUpdate(sql)
```

JDO

```
<table name="PERSONAL">
    <column name="PNR" type="INTEGER" required="true" primaryKey="true"
        autoIncrement="true"/>
    <column name="VORNAME" type="VARCHAR" size="50" required="true"/>
    <column name="NACHNAME" type="VARCHAR" size="50" required="true"/>
    <column name="GEH_STUFE" type="VARCHAR" size="5"/>
    <column name="ABT_NR" type="VARCHAR" size="5"/>
    <column name="KRANKENKASSE" type="VARCHAR" size="3"/>
    <foreign-key foreignTable="GEHALT">
        <reference local="GEH_STUFE" foreign="GEH_STUFE"/>
    </foreign-key>
    <foreign-key foreignTable="ABTEILUNG">
        <reference local="ABT_NR" foreign="ABT_NR"/>
    </foreign-key>
</table>
```

CODE-BEISPIEL

ORACLE JDBC-TREIBER

HTTP://WWW.ORACLE.COM/TECHNETWORK/DATABASE/ENTERPRISE-EDITION/JDBC-112010-090769.HTML

AUFGABE 1

FÜGEN SIE DIE RESTLICHEN FELDER DER KLASSE `Personal` HINZU UND ERGÄNZEN SIE DIE `toString` - FUNKTION

AUFGABE 2

ERSTELLEN SIE DIE STRUKTUR FÜR PERSONAL ANALOG FÜR DIE TABELLE MASCHINE:

- LEGEN SIE EINE NEUE KLASSE AN
- GEBEN SIE ALLE MASCHINEN MIT ALLEN ATTRIBUTEN AUF DER KOMMANDOZEILE AUS

AUFGABE 3

FUGEN SIE EINE NEUE ABTEILUNG HINZU

AUFGABE 4

FUGEN SIE EINE NEUE MASCHINE UND EINEN NEUEN MITARBEITER HINZU. BAUEN SIE DIESE FUNKTION SO, DASS WENN EIN `INSERT` FEHLSCHLÄGT, KEINE DER BEIDEN EINTRÄGE IN DER DATENBANK STEHEN (SIEHE `rollback`)

DAS WAR'S FÜR HEUTE