Datenbanken

Aufgabenblatt 9 (Transaktionen)

Prof. Dr.-Ing. Heiko Tapken / M.Ed. Friedhelm Tappe

Testat: KW 51

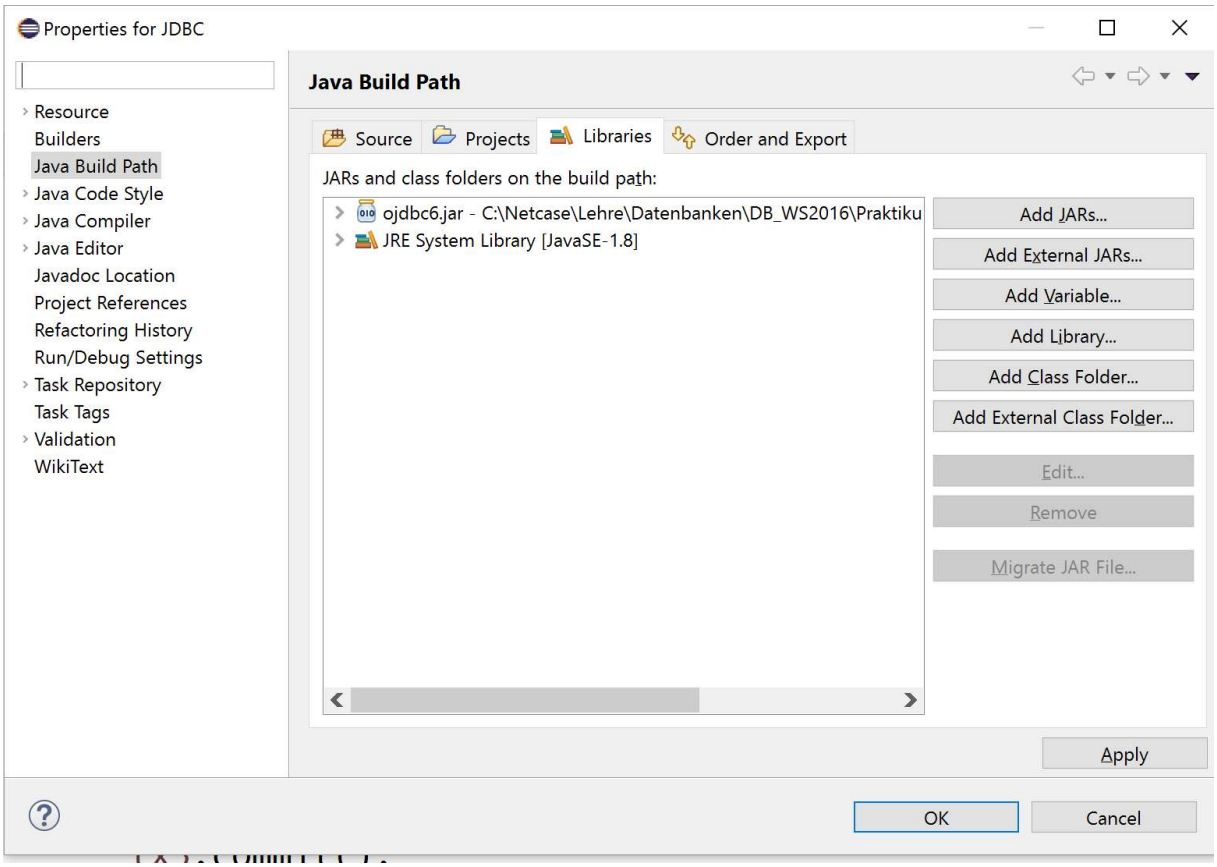
Wintersemester 2019/20

Bestehensgrenze: 12 Punkte

Hochschule Osnabrück
University of Applied Sciences

In der Vorlesung „Exkurs: Zugriffsmöglichkeiten“ haben wir uns u.a. kurz den Zugriff mittels JDBC und ein kleines Beispiel angesehen. Ihre Aufgabe besteht zunächst darin, sich neben dem in der Vorlesung besprochenen Beispiel den Vorlesungs-Foliensatz zu JDBC (OSCA) anzusehen und nachfolgende Aufgaben zu lösen.

Wie in der Vorlesung gezeigt benötigen wir für den Zugriff auf die Datenbank einen Datenbank-Treiber. Dieser wird unter Project-> Properties -> Add External JARs eingebunden. Den Treiber finden Sie im OSCA.



Aufgabe 1 JDBC (2 Punkte)

In dieser Aufgabe geht es darum, einen Datenbankzugriff mittels JDBC zu realisieren, um diese Erkenntnis in späteren Projekten nutzen zu können.

Ihre Aufgabe ist es, ein kleines Lagerverwaltungssystem für Kartoffeln zu entwickeln.

Eine Kartoffelqualität zeichnet sich aus durch eine Sorte, eine Größe, ein Gewicht, einen Stärkegehalt und einer Fleckigkeit. Ferner können Kartoffeln durch Sonnenlicht Grün werden.

Kartoffeln, die in Gebinden (mehrere Kartoffeln zusammen) der gleichen Qualität zugehörig sind, werden entweder im Produktionslager der Fabrik oder aber in externen Großraumlägern gelagert. Das Produktionslager im Bunker besteht aus verschiedenen Bunkern. In jedem Bunker lagern Kartoffeln einer Sorte und einer Größenklasse. In einem Großraumlager, von dem wir die Adresse und die Kapazität kennen, wird vermerkt, an welcher Stelle welche Kartoffelgröße gelagert wird und der Einlagerungszeitpunkt. Zur Bestimmung der Lagerstelle wird eine Unterteilung analog zu einem Schachbrett vorgenommen.

- Modellieren Sie das Datenbankschema
- Überführen Sie dieses in SQL, denken Sie an geeignete Domänen
- Erstellen Sie ein kleines Java-Programm, mit dem Sie – OHNE SQL Deloper o.ä. ein DDL-Skript auf der Datenbank ausführen können. Sie entscheiden, ob das Programm eine Oberfläche zur Auswahl der Datei und zur Darstellung von Fehlern bekommt.
- Schreiben Sie ein Programm für das Kartoffellager. Man muss:
    a. Kartoffeln einlagern können (sowohl in den Bunkern der Produktionsläger und in Großraumlägern.
    b. Kartoffeln Großraumlager in einen speziellen Bunker im Produktionslager umlagern können.
    c. Jederzeit einen über die Bestände jedes Lagers haben (Qualitäten, Mengen, …) Überraschen sie uns.
    d. Kartoffeln für die Produktion aus den Bunkern entnehmen können (nach Qualitäten).

Das Programm soll eine ansprechende Menüführung und nutzbare Oberfläche besitzen.

Aufgabe 2 Transaktionen (4 Punkte)

Installieren Sie bei jedem an der Aufgabe beteiligten Nutzer (sie benötigen zwei) die Tabelle See (see.sql von Veranstaltungswebseite) und dokumentieren Sie die evtl. überraschenden Ergebnisse.

a) Stellen Sie auf einem Rechner zwei Verbindungen zur Datenbank mit unterschiedlichen Nutzern her (z. B. SQL-Developer zweimal starten).
b) Geben Sie sich gegenseitig die Tabelle See zum Ändern frei (genauer GRANT ALL ON See TO ...) und richten dafür ein Synonym ein.
c) Prüfen Sie, ob ihre Tabellen See die gleichen Einträge haben, wie die Tabelle des anderen Nutzers (mit einer SQL-Anfrage, überlegen Sie sich, wie man „Mengengleichheit“ in SQL prüfen kann).
d) Fügen Sie bei einem Nutzer in die Tabelle See einen Baggersee mit der Tiefe 1 ein, was ist bei beiden Nutzern in der Tabelle See sichtbar? Versuchen Sie vom anderen Nutzer aus die gleichen Daten in die gleiche Tabelle über das Synonym einzutragen. Wie verhält sich das Worksheet? Was passiert dann, wenn der erste Nutzer COMMIT eingibt?
e) Erhöhen Sie bei einem Nutzer die Tiefe des Baggersees um 1, versuchen Sie die Erhöhung in der gleichen Tabelle vom anderen Nutzer aus, danach soll der erste Nutzer COMMIT eingeben. Wie hat sich See bei beiden Nutzern verändert? Was passiert mit den Tabelleninhalten, wenn auch der andere Nutzer COMMIT eingibt?

f) Der eine Nutzer soll einen View auf die Namen in See erstellen, diesen für den anderen Nutzer für alle Operationen freigeben und ihm die Rechte an der ursprünglichen Tabelle See wegnehmen. Kann der zweite Nutzer über den View die Tabelle lesen/verändern?

Aufgabe 3 – Anomalien [5 Punkte]

Stellen Sie dar, ob und wenn ja welche ANOMALIEN (lost update, dirty read, unrepeatable read, ghost update) bei Ausführung der folgenden Schedules produziert werden. Begründen Sie die von Ihnen gemachte Feststellung.

(a) $r_1(x), r_2(x), w_1(x), r_2(y), r_1(y), w_2(y), w_2(x), c_1, c_2$

(b) $r_1(x), r_2(x), w_1(x), r_2(x), a_1, a_2$

(c) $r_1(x), r_2(x), r_2(y), w_2(y), w_2(x), r_1(x), w_1(z), c_1, c_2$

(d) $r_1(z), r_1(x), r_1(z), r_2(y), w_1(z), w_1(y), c_2, c_1$

(e) $r_1(x), r_1(y), w_1(x), r_2(x), w_2(y), w_2(z), a_1, c_2$

Hierbei steht $r_i(x)$ für einen lesenden Zugriff der Transaktion i auf das Objekt x und $w_i(x)$ für einen schreibenden Zugriff der Transaktion i auf das Objekt x. Weiterhin zeigen a_i bzw. c_i das „Resultat" einer Transaktion i an (a=abort und c=commit).

Aufgabe 4 – Serialisierbarkeit [5 Punkte]

Klassifizieren Sie die folgenden Schedules als *view-serialisierbar / nicht view-serialisierbar* und als *konflikt-serialisierbar / nicht konflikt-serialisierbar*. Erläutern Sie die von Ihnen jeweils getroffene Entscheidung und illustrieren Sie Ihre Überlegungen durch die Angabe einer möglichen seriellen Ausführung der Schedules und/oder jeweils vollständige Konfliktgraphen (Präzedenzgraphen).

(a) $r_1(x), w_1(x), r_2(x), w_1(x), w_3(x), w_4(x)$

(b) $r_2(z), r_1(x), w_1(x), r_3(y), r_2(x), w_1(y), r_1(x), w_3(z), r_2(y), r_2(z)$

(c) $r_2(x), w_1(y), w_3(y), r_5(z), r_1(z), w_4(x), r_2(z), w_4(t), w_1(t), r_3(y)$

(d) $w_2(x), w_1(t), r_4(t), r_1(x), r_3(t), w_1(y), r_5(y), w_3(t), w_5(y), r_5(t), r_1(z)$

(e) $r_1(y), w_3(x), r_2(y), r_4(x), w_1(y), w_4(z), r_4(y), w_2(y), w_2(x)$

Anm : Alle Transaktionen werden hier immer mit Commit beendet.

Rechte erteilen
Um einen Benutzer alle Rechte auf eine Tabelle zu geben nutzt man den Befehl

Grant ALL ON <object> TO <username>

gibt dem Benutzer <username> alle Rechte (aktualisieren, löschen, … auf die Tabelle/View. … <object>

Bsp: (Ausgeführt als Benutzer test)

GRANT ALL ON country TO tapken

räumt dem Benutzer tapken alle Rechte auf die Tabelle country des Benutzers test ein.

Der Zugriff auf die Tabelle erfolgt dann nach dem Prinzip der Punktnotation

<schemaname>.<tabellenname>.

In Oracle entsprechen die Schemata den Nutzernamen. Aus diesem Grund greift nun der Nutzer Tapken wie folgt auf die Tabelle zu:

*SELECT * FROM test.country;*

Dies ist nicht sonderlich komfortabel.

Synonyme:
Durch die Einführung von Synonymen kann man auf die Punktnotation verzichten.

CREATE [PUBLIC] SYNONYM <synonym>
FOR <schema>.<object>;

schafft die Möglichkeit, über <synonym> auf das Objekt (Tabelle, View, …) <schema>.<object> zuzugreifen. Beispiel (als Benutzer tapken ausgeführt):

CREATE SYNONYM country1
FOR test.country

ermöglicht jetzt folgende Anfragen (ausgeführt wieder als Nutzer Tapken)

SELECT * FROM country1
(ist äquivalent zu SELECT * FROM test.country

Löschen von Synonymen geschieht mit:

Drop <synonym> löscht diese Kurzschreibweise wieder.

Löschen von Rechten:
Mit

REVOKE ALL ON COUNTRY FROM tapken(ausgeführt als Benutzer test)

entzieht der Benutzer test das Recht dem Benuter tapken wieder, allgemein

REVOKE ALL ON <object> FROM <username>

Nähere Details finden Sie in dem Foliensatz Rechte