DES3UE Datenbanksysteme WS 2024 Übung 4

Abgabetermin: siehe elearning

☐ **DES31UE Glock** **Name** _____ **Aufwand in h** _____
☐ **DES32UE Werth** **Punkte** _____ **Kurzzeichen Tutor** _____

Ziel dieser Übung ist die Erstellung von gespeicherten Prozeduren in der Datenbank, sowie die Vertiefung von PL/SQL in Paketen, dem Cursor-Konzept und Behandlung von Exceptions.

1. Datenbankpakete (PL/SQL-Packages) **(10 Punkte –** 1. 2 Pkt, 2.-3. je 3 Pkt**)**

Mit Hilfe von PL/SQL-Packages können Sie zusammengehörige PL/SQL-Typen, Variablen, Datenstrukturen, Exceptions und Unterprogramme in einer Bibliothek zusammenfassen. Packages bestehen normalerweise aus zwei Komponenten (Spezifikation und Body), die separat in der Datenbank gespeichert werden. Das Package selbst kann nicht aufgerufen, parametrisiert oder verschachtelt werden.

1. Erstellen Sie ein Datenbankpaket get_sales_volume_pkg mit der *gegebenen* Funktion GetRevenue GetRevenue ermittelt aus der Sakila Datenbank das Umsatzvolumen (= Gesamtsumme aller getätigten Bestellungen) eines Standorts (= store). Speichern Sie die Package Spezifikation und den Package Body gemeinsam ab. Erstellen Sie einen **Testaufruf** (zB für store_id = 2), der das Ergebnis ausgibt (z.B: „Store with id 2: *n*$).

```
FUNCTION GetRevenue (store IN NUMBER)
RETURN NUMBER
IS
  sum_payments NUMBER;
BEGIN
   SELECT SUM(amount) INTO sum_payments
   FROM payment p
         INNER JOIN staff s USING (staff_id)
  WHERE s.store_id = store;
  RETURN sum_payments;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
    RETURN 0;
END;
```

2. Implementieren Sie im Package get_sales_volume_pkg eine zusätzliche Funktion GetTop3Stores, die eine Top-3 Liste der 3 besten Standorte als PIPELINED TABLE zurückgibt, d.h. der Standorte, die am meisten Umsatz machen. Verwenden Sie dazu eine CURSOR FOR LOOP (und die Prozedur DBMS_OUTPUT.PUT_LINE zu Debug-Zwecken). Verwenden Sie zur Ermittlung des Umsatzes die bereits vorhandene Funktion. Geben Sie die Package Spezifikation, den Package Rumpf sowie den **Testaufruf** an. Sortieren Sie nach dem Umsatz absteigend. Nutzen Sie außerdem die bereits gegebenen Datentypen

```
CREATE TYPE top_store_row AS OBJECT (
storeid      NUMBER,
storecity     VARCHAR2(100),
revenue       NUMBER
);
/
```

```
CREATE TYPE top_store_tab IS TABLE OF top_store_row;
```

Beispielausgabe:

	STOREID	STORECITY	REVENUE
1	2	Woodridge	6207

(erste von 3 Zeilen)

3. Verändern Sie Ihre bereits erstellte Funktion GetTop3Stores so, dass sie allgemein gültig ist. Bezeichnen Sie die veränderte Funktion mit GetTopNStores. Diese Prozedur soll die besten N Standorte ausgeben. Erweitern Sie dazu die GetTop3Stores um einen Eingangsparameter Anzahl n_count. Fügen Sie dem Parameter einen Default-Wert hinzu (zB n_count = 3), damit die Prozedur weiterhin ohne Parameter aufgerufen werden kann. Geben Sie die Package Spezifikation, den Package Rumpf sowie den **Testaufruf** (mit und ohne Parameter) an.

Beispielausgabe

```
...
The top 4 stores are:
Der Store Nr 2 in Woodridge erwirtschaftet: 6207$
…
```

2. PL/SQL Prozeduren **(6 Punkte –** 1. 1 Pkt, 2.-3. Je 2 Pkt**)**

Anm: Für dieses Beispiel benötigen Sie die erweiterte Tabelle top_salaries aus dem Skript `scriptTopSalaries.sql`).

1. Für die Datensätze in der Tabelle top_salaries werden Logging-Daten von der Erstellung sowie von der letzten Änderung benötigt. Erweitern Sie dazu die Tabelle top_salaries um die Felder createdBy, dateCreated, modifiedBy und dateModified. Bei der Anlage eines Datensatzes sind die Created- und Modified-Felder ident. Speichern Sie das DDL-Skript ab.

2. Erstellen Sie eine Datenbank-Prozedur InsertTopSalaries, die einen Datensatz in der Tabelle top_salaries anlegt und die Logging-Felder befüllt. Für die Logging-Felder verwenden Sie die Systemfunktionen USER und SYSDATE. Die Systemfunktion USER liefert den Namen des angemeldeten Benutzers. Die Systemfunktion SYSDATE liefert das aktuelle Systemdatum. Das Skript für die Erstellung der Prozedur speichern Sie ab. Die Prozedur soll folgende Spezifikation aufweisen:

```
CREATE OR REPLACE PROCEDURE InsertTopSalaries (
    pSalary     IN NUMBER,
    pEmp_cnt    IN NUMBER)
IS  ...
```

3. Ersetzen Sie die INSERT-Anweisung im Skript durch die in der vorherigen Aufgabe erstellte Prozedur InsertTopSalaries und überprüfen Sie das Ergebnis. Speichern Sie das Skript ab. Hinweis: Um auch die Uhrzeit zu sehen, können Sie das Datumsformat mit folgendem Kommando festlegen:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'dd.mm.yyyy hh24:mi:ss';
```

3. Cursor mit FOR-UPDATE **(8 Punkte –** 1.-3 je 2 Pkt; 4. 1 Pkt**)**

Wenn mehrere Sessions für eine einzelne Datenbank vorhanden sind, besteht die Möglichkeit, dass die Zeilen einer bestimmten Tabelle aktualisiert wurden, nachdem Sie den Cursor geöffnet haben. Sie sehen die aktualisierten Daten nur, wenn Sie den Cursor erneut öffnen. Es ist daher günstiger, die Zeilen zu sperren, bevor Sie Zeilen aktualisieren oder löschen. Sie können zum Sperren der Zeilen die FOR-UPDATE-Klausel in der Cursor-Abfrage verwenden.

Verwenden Sie für die folgenden Aufgaben die Tabelle top_salaries aus dem letzten Beispiel.

1. Erweitern Sie im Datenbankpaket top_salary_pkg (vgl. `top_salary_pkg.sql`) die Prozedur GetTopSalaries um die beiden Eingangsparameter LowSalary und HighSalary, um die Auswahl von Datensätze auf den Bereich LowSalary <= salary <= HighSalary einschränken zu können. Geben Sie die Package Spezifikation, den Package Rumpf sowie den Testaufruf an.

2. Erweitern Sie das Datenbankpaket top_salary_pkg um die Prozedur DoubleTopSalaries, die die Gehälter der mit pLowSalary und pHighSalary selektierten Sätze verdoppelt. Aktualisieren Sie auch die Logging-Felder modifiedby und dateModified. Die Prozedur soll folgende Spezifikation aufweisen:

```
PROCEDURE DoubleTopSalaries (
      pLowSalary  IN NUMBER,
      pHighSalary IN NUMBER)
IS . . .
```

Verwenden Sie zum Testen z.B. folgende Testaufrufe:

```
SHOW ERR

SET SERVEROUTPUT ON

ALTER SESSION SET nls_date_format = 'dd.mm.yyyy hh24:mi:ss';

BEGIN
  top_salary_pkg.FillTopSalaries(3);
  dbms_output.put_line('.................');
  top_salary_pkg.GetTopSalaries(0, 100000);
  dbms_output.put_line('.................');
  top_salary_pkg.DoubleTopSalaries(11000, 13500);
  dbms_output.put_line('.................');
  top_salary_pkg.GetTopSalaries(0, 100000);
  dbms_output.put_line('.................');

END;
/
```

Hinweise: Verwenden Sie beim Cursor die FOR-UPDATE und beim UPDATE die WHERE CURRENT OF-Klausel. Geben Sie die Package Spezifikation, den Package Body sowie den Testaufruf an.

3. Öffnen Sie eine zweite Datenbank-Session und führen Sie die Prozedur DoubleTopSalaries mit den gleichen Parametern in jeder Session aus, ohne ein COMMIT auszuführen. Wählen Sie die Parameter so, damit zumindest ein Satz aus TopSalaries selektiert wird. Was passiert in der zweiten Session? Führen Sie in der ersten ein COMMIT aus und beschreiben Sie die Auswirkungen.

4. Erweitern Sie nun den Cursor in der Prozedur DoubleTopSalaries um die NOWAIT-Klausel. Wiederholen Sie den Test und erläutern Sie den Unterschied. Geben Sie die Package Spezifikation, den Package Rumpf sowie den Testaufruf an.