# DES3UE Datenbanksysteme WS 2024 Übung 4

X			Elias Leonhardsberger	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
);
/
```

Beispielausgabe:

∯ Sī	TOREID   STORECITY		
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';
```

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('......');
```

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.

# DES3UE Übung 4

# Elias Leonhardsberger

# 9. Dezember 2024, Hagenberg

# Inhaltsverzeichnis

1	Datenbankpakete (PL/SQL-Packages)					
	1.1 SQL	5				
	1.2 Ergebnisse	7				
2	PL/SQL Prozeduren	8				
	2.1 SQL	8				
	2.2 Ergebnisse	9				
3	Cursor mit FOR-UPDATE	9				
	3.1 SQL	9				
	3.2 Ergebnisse					

### 1 Datenbankpakete (PL/SQL-Packages)

#### 1.1 **SQL**

```
DROP TYPE top store tab;
   DROP TYPE top_store_row;
   -- 2
   CREATE TYPE top store row AS OBJECT
6
       storeId
                  NUMBER,
       storeCity VARCHAR2(100),
9
       revenue
                  NUMBER
10
   );
11
12
13
   CREATE TYPE top store tab IS TABLE OF top store row;
14
15
   CREATE OR REPLACE PACKAGE get_sales_volume_pkg AS
16
       FUNCTION GetRevenue(store IN NUMBER)
            RETURN NUMBER;
19
20
       FUNCTION GetTop3Stores
21
            RETURN top_store_tab PIPELINED;
       FUNCTION GetTopNStores(n IN NUMBER := 3)
            RETURN top store tab PIPELINED;
   END;
26
27
   CREATE OR REPLACE PACKAGE BODY get sales volume pkg AS
28
29
       FUNCTION GetRevenue(store IN NUMBER)
            RETURN NUMBER
            IS
32
            sum payments NUMBER;
33
       BEGIN
34
            SELECT SUM(amount)
35
            INTO sum payments
36
            FROM payment p
            INNER JOIN staff s
                USING (staff id)
39
            WHERE s.store id = store;
40
            RETURN sum_payments;
41
       EXCEPTION
42
            WHEN NO DATA FOUND THEN
43
                RETURN 0;
44
```

```
END;
45
46
       -- 2
       FUNCTION GetTop3Stores
48
           RETURN top_store_tab PIPELINED
49
50
           CURSOR storeCursor IS SELECT s.STORE ID, C.CITY,
51

→ COALESCE((GetRevenue(s.STORE_ID)), 0) AS revenue

           FROM STORE s
52
           INNER JOIN ADDRESS a
53
                ON s.ADDRESS ID = a.ADDRESS ID
54
           INNER JOIN CITY C
55
                ON a.CITY_ID = C.CITY_ID
56
           ORDER BY revenue DESC
57
                FETCH FIRST 3 ROWS ONLY; -- you could also use with ties to
                    include 2 third places for example
       BEGIN
59
           FOR storeRec IN storeCursor
60
                LOOP
61
                    PIPE ROW (top_store_row(storeRec.STORE_ID, storeRec.CITY,
62
                        storeRec.revenue));
                END LOOP;
       END;
64
65
       -- 3
66
       FUNCTION GetTopNStores(n IN NUMBER := 3)
67
           RETURN top_store_tab PIPELINED
           IS
69
           CURSOR storeCursor IS SELECT s.STORE ID, C.CITY,
               COALESCE((GetRevenue(s.STORE ID)), 0) AS revenue
           FROM STORE s
71
           INNER JOIN ADDRESS a
72
                ON s.ADDRESS ID = a.ADDRESS ID
73
           INNER JOIN CITY C
                ON a.CITY_ID = C.CITY_ID
75
           ORDER BY revenue DESC
                FETCH FIRST n ROWS ONLY; -- you could also use with ties to
                    include 2 third places for example
       BEGIN
78
           FOR storeRec IN storeCursor
79
                LOOP
80
                    PIPE ROW (top_store_row(storeRec.STORE_ID, storeRec.CITY,

    storeRec.revenue));
                END LOOP;
82
       END;
83
   END;
84
```

85

```
SELECT get_sales_volume_pkg.GetRevenue(2)
   FROM dual;
89
    -- 2
90
   SELECT *
   FROM TABLE (get sales volume pkg.GetTop3Stores());
92
   -- 3
   SELECT *
95
   FROM TABLE (get sales volume pkg.GetTopNStores());
96
97
   -- 3
98
   SELECT *
99
   FROM TABLE (get_sales_volume_pkg.GetTopNStores(5));
100
101
```

### 1.2 Ergebnisse

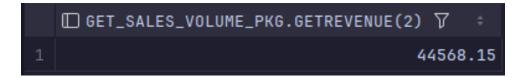


Abbildung 1: Ergebnis Aufgabe 1.1

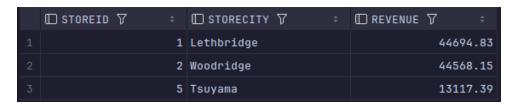


Abbildung 2: Ergebnis Aufgabe 1.2

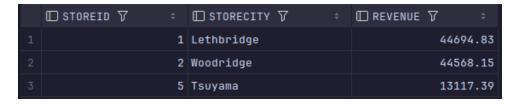


Abbildung 3: Ergebnis 1 Aufgabe 1.3

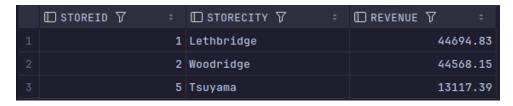


Abbildung 4: Ergebnis 2 Aufgabe 1.3

### 2 PL/SQL Prozeduren

#### 2.1 SQL

```
DROP TABLE top_salaries;
   CREATE TABLE top salaries
5
       salary
                   NUMBER(8, 2),
6
       emp cnt
                   NUMBER(3) DEFAULT NULL,
       createdBy VARCHAR2(100),
       dateCreated TIMESTAMP,
       modifiedBy VARCHAR2(100),
10
       dateModified TIMESTAMP
11
   );
12
13
   ALTER TABLE top_salaries
14
       ADD CONSTRAINT top_salaries_pk PRIMARY KEY (salary);
15
   ALTER TABLE top salaries
       ADD CONSTRAINT top_salaries_emp_cnt_gt0 CHECK (emp_cnt > 0);
18
   TRUNCATE TABLE top salaries;
19
20
21
   CREATE OR REPLACE PROCEDURE InsertTopSalaries(
22
       pSalary IN NUMBER,
       pEmp_cnt IN NUMBER)
       IS
25
   BEGIN
26
       INSERT INTO top_salaries (salary, emp_cnt, createdBy, dateCreated,
27

→ modifiedBy, dateModified)

       VALUES (pSalary, pEmp_cnt, USER, CURRENT_TIMESTAMP, USER,
           CURRENT_TIMESTAMP);
   END;
29
30
   -- 3
31
   DECLARE
32
       num NUMBER(3) := 3;
33
       sal employees.salary%TYPE;
```

```
vEmp_cnt top_salaries.emp_cnt%TYPE;
35
       CURSOR emp_cursor IS
            SELECT salary, COUNT(*)
37
            FROM employees
38
            GROUP BY salary
39
            ORDER BY salary DESC
40
            FETCH FIRST num ROWS ONLY;
41
   BEGIN
42
       OPEN emp_cursor;
43
       FETCH emp_cursor INTO sal, vEmp_cnt;
       WHILE emp_cursor%FOUND
45
            LOOP
46
                InsertTopSalaries(sal, vEmp_cnt);
47
                FETCH emp_cursor INTO sal, vEmp_cnt;
48
            END LOOP;
       CLOSE emp_cursor;
   END;
51
52
   ALTER SESSION SET NLS_DATE_FORMAT = 'dd.mm.yyyy hh24:mi:ss';
53
   SELECT *
   FROM top_salaries;
```

#### 2.2 Ergebnisse

	SALARY ♥ ÷	□ EMP_CNT 🎖 🗼	☐ CREATEDBY 🎖 💠	□ DATECREATED ♥ ÷	□ MODIFIEDBY ♡ ÷	□ DATEMODIFIED ♥ ÷
1	24000.00		\$2310307019	2024-12-09 11:09:22.144390	S2310307019	2024-12-09 11:09:22.144390
2	17000.00		\$2310307019	2024-12-09 11:09:22.145107	\$2310307019	2024-12-09 11:09:22.145107
3	13000.00		\$2310307019	2024-12-09 11:09:22.145153	S2310307019	2024-12-09 11:09:22.145153

Abbildung 5: Ergebnis Aufgabe 2

#### 3 Cursor mit FOR-UPDATE

#### 3.1 SQL

```
CREATE OR REPLACE PACKAGE top_salary_pkg AS

PROCEDURE GetTopSalaries(pLowSalary IN NUMBER, pHighSalary IN

NUMBER);

PROCEDURE DoubleTopSalaries(pLowSalary IN NUMBER, pHighSalary IN

NUMBER);

PROCEDURE FillTopSalaries(pNum IN NUMBER);

END;

CREATE OR REPLACE PACKAGE BODY top_salary_pkg AS

PROCEDURE GetTopSalaries(pLowSalary IN NUMBER, pHighSalary IN NUMBER)
```

```
IS
9
           CURSOR cTopSalaries IS
10
                SELECT 'Salary: ' | TO CHAR(SALARY) | ' . EmpCnt: ' | |
11
                    TO CHAR(EMP CNT) ||
                        '. Erstellt: | | TO_CHAR(CREATEDBY) || '/' ||
12
                        → TO_CHAR(DATECREATED) | |
                        '. Geändert: ' || TO CHAR(MODIFIEDBY) || '/' ||
13
                            TO_CHAR(DATEMODIFIED) || '.' text
                FROM top_salaries
                WHERE SALARY BETWEEN pLowSalary AND pHighSalary
15
                ORDER BY SALARY DESC;
16
       BEGIN
17
           FOR vTopSalaries IN cTopSalaries
                LOOP
19
                    DBMS OUTPUT.PUT LINE(vTopSalaries.text);
                END LOOP;
       END;
       PROCEDURE DoubleTopSalaries(
23
           pLowSalary IN NUMBER,
24
           pHighSalary IN NUMBER)
25
           IS
26
           CURSOR cTopSalaries IS
                SELECT salary, MODIFIEDBY, DATEMODIFIED
                FROM top_salaries
29
                WHERE salary BETWEEN pLowSalary AND pHighSalary
30
                FOR UPDATE OF salary, MODIFIEDBY, DATEMODIFIED NOWAIT;
31
       BEGIN
           FOR vTopSalaries IN cTopSalaries
33
                LOOP
                    UPDATE top salaries
                    SET salary
                                      = salary * 2,
36
                        MODIFIEDBY
                                     = USER,
37
                        DATEMODIFIED = CURRENT_TIMESTAMP
38
                    WHERE CURRENT OF cTopSalaries;
                END LOOP;
40
       END;
       PROCEDURE FillTopSalaries(pNum IN NUMBER)
           IS
43
            sal employees.salary%TYPE;
44
           vEmp_cnt top_salaries.emp_cnt%TYPE;
45
           CURSOR emp_cursor IS
                SELECT salary, COUNT(*)
                FROM employees
48
                GROUP BY salary
49
                ORDER BY salary DESC
50
                FETCH FIRST pNum ROWS ONLY;
51
       BEGIN
52
```

```
OPEN emp cursor;
53
          FETCH emp_cursor INTO sal, vEmp_cnt;
          WHILE emp cursor%FOUND
55
              LOOP
56
                  InsertTopSalaries(sal, vEmp_cnt);
57
                  FETCH emp_cursor INTO sal, vEmp_cnt;
58
              END LOOP;
59
           CLOSE emp_cursor;
       END;
   END;
62
63
   TRUNCATE TABLE top_salaries;
64
   ALTER SESSION SET nls_date_format = 'dd.mm.yyyy hh24:mi:ss';
65
   BEGIN
66
       top_salary_pkg.FillTopSalaries(3);
67
       dbms_output.put_line('....');
68
       top salary pkg.GetTopSalaries(0, 100000);
69
       dbms output.put line('....');
70
       top_salary_pkg.DoubleTopSalaries(11000, 13500);
71
       dbms_output.put_line('....');
72
       top salary pkg.GetTopSalaries(0, 100000);
       dbms_output.put_line('....');
   END;
75
76
   BEGIN
77
       top salary pkg.GetTopSalaries(0, 100000);
78
  END;
79
```

### 3.2 Ergebnisse

Abbildung 6: Ergebnis Aufgabe 3.1-3.2