

<input checked="" type="checkbox"/> <b>DES31UE Glock</b>	<b>Name</b> <u>Elias Leonhardsberger</u>	<b>Aufwand in h</b> <u>3.5</u>
<input type="checkbox"/> <b>DES32UE Werth</b>	<b>Punkte</b> _____	<b>Kurzzeichen Tutor</b> _____

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).

- 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.
- 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 ...
```

- 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.

# DES3UE Übung 4

Elias Leonhardsberger

12. Dezember 2024, Hagenberg

## Inhaltsverzeichnis

<b>1 Datenbankpakete (PL/SQL-Packages)</b>	<b>5</b>
1.1 SQL . . . . .	5
1.2 Ergebnisse . . . . .	7
<b>2 PL/SQL Prozeduren</b>	<b>8</b>
2.1 SQL . . . . .	8
2.2 Ergebnisse . . . . .	9
<b>3 Cursor mit FOR-UPDATE</b>	<b>9</b>
3.1 SQL . . . . .	9
3.2 Ergebnisse . . . . .	12
3.2.1 Ohne NOWAIT und COMMIT . . . . .	12
3.2.2 Ohne NOWAIT, mit COMMIT . . . . .	12
3.2.3 Mit NOWAIT . . . . .	13

# 1 Datenbankpakete (PL/SQL-Packages)

## 1.1 SQL

```
1  -- 2
2  DROP TYPE top_store_tab;
3  DROP TYPE top_store_row;
4
5  -- 2
6  CREATE TYPE top_store_row AS OBJECT
7  (
8      storeId    NUMBER,
9      storeCity  VARCHAR2(100),
10     revenue    NUMBER
11 );
12
13 -- 2
14 CREATE TYPE top_store_tab IS TABLE OF top_store_row;
15
16 CREATE OR REPLACE PACKAGE get_sales_volume_pkg AS
17     -- 1
18     FUNCTION GetRevenue(store IN NUMBER)
19         RETURN NUMBER;
20     -- 2
21     FUNCTION GetTop3Stores
22         RETURN top_store_tab PIPELINED;
23     -- 3
24     FUNCTION GetTopNStores(n IN NUMBER := 3)
25         RETURN top_store_tab PIPELINED;
26 END;
27
28 CREATE OR REPLACE PACKAGE BODY get_sales_volume_pkg AS
29     -- 1
30     FUNCTION GetRevenue(store IN NUMBER)
31         RETURN NUMBER
32     IS
33         sum_payments NUMBER;
34     BEGIN
35         SELECT SUM(amount)
36         INTO sum_payments
37         FROM payment p
38         INNER JOIN staff s
39             USING (staff_id)
40         WHERE s.store_id = store;
41         RETURN sum_payments;
42     EXCEPTION
43         WHEN NO_DATA_FOUND THEN
44             RETURN 0;
```

```

45     END;
46
47     -- 2
48     FUNCTION GetTop3Stores
49         RETURN top_store_tab PIPELINED
50         IS
51         CURSOR storeCursor IS SELECT s.STORE_ID, C.CITY,
52             ↳ COALESCE((GetRevenue(s.STORE_ID)), 0) AS revenue
53         FROM STORE s
54         INNER JOIN ADDRESS a
55             ON s.ADDRESS_ID = a.ADDRESS_ID
56         INNER JOIN CITY C
57             ON a.CITY_ID = C.CITY_ID
58         ORDER BY revenue DESC
59         FETCH FIRST 3 ROWS ONLY; -- you could also use with ties to
60             ↳ include 2 third places for example
61
62     BEGIN
63         FOR storeRec IN storeCursor
64             LOOP
65                 PIPE ROW (top_store_row(storeRec.STORE_ID, storeRec.CITY,
66                     ↳ storeRec.revenue));
67             END LOOP;
68     END;
69
70     -- 3
71     FUNCTION GetTopNStores(n IN NUMBER := 3)
72         RETURN top_store_tab PIPELINED
73         IS
74         CURSOR storeCursor IS SELECT s.STORE_ID, C.CITY,
75             ↳ COALESCE((GetRevenue(s.STORE_ID)), 0) AS revenue
76         FROM STORE s
77         INNER JOIN ADDRESS a
78             ON s.ADDRESS_ID = a.ADDRESS_ID
79         INNER JOIN CITY C
80             ON a.CITY_ID = C.CITY_ID
81         ORDER BY revenue DESC
82         FETCH FIRST n ROWS ONLY; -- you could also use with ties to
83             ↳ include 2 third places for example
84
85     BEGIN
86         FOR storeRec IN storeCursor
87             LOOP
88                 PIPE ROW (top_store_row(storeRec.STORE_ID, storeRec.CITY,
89                     ↳ storeRec.revenue));
90             END LOOP;
91     END;
92 END;
93
94
95

```

```

86  -- 1
87  SELECT get_sales_volume_pkg.GetRevenue(2)
88  FROM dual;
89
90  -- 2
91  SELECT *
92  FROM TABLE (get_sales_volume_pkg.GetTop3Stores());
93
94  -- 3
95  SELECT *
96  FROM TABLE (get_sales_volume_pkg.GetTopNStores());
97
98  -- 3
99  SELECT *
100 FROM TABLE (get_sales_volume_pkg.GetTopNStores(5));
101

```

## 1.2 Ergebnisse

	GET_SALES_VOLUME_PKG.GETREVENUE(2)
1	44568.15

Abbildung 1: Ergebnis Aufgabe 1.1

	STOREID	STORECITY	REVENUE
1	1	Lethbridge	44694.83
2	2	Woodridge	44568.15
3	5	Tsuyama	13117.39

Abbildung 2: Ergebnis Aufgabe 1.2

	STOREID	STORECITY	REVENUE
1	1	Lethbridge	44694.83
2	2	Woodridge	44568.15
3	5	Tsuyama	13117.39

Abbildung 3: Ergebnis 1 Aufgabe 1.3

	☐ STOREID ▼	÷	☐ STORECITY ▼	÷	☐ REVENUE ▼	÷
1	1		Lethbridge		44694.83	
2	2		Woodridge		44568.15	
3	5		Tsuyama		13117.39	

Abbildung 4: Ergebnis 2 Aufgabe 1.3

## 2 PL/SQL Prozeduren

### 2.1 SQL

```

1 DROP TABLE top_salaries;
2
3 -- 1
4 CREATE TABLE top_salaries
5 (
6     salary      NUMBER(8, 2),
7     emp_cnt     NUMBER(3) DEFAULT NULL,
8     createdBy  VARCHAR2(100),
9     dateCreated TIMESTAMP,
10    modifiedBy VARCHAR2(100),
11    dateModified TIMESTAMP
12 );
13
14 ALTER TABLE top_salaries
15     ADD CONSTRAINT top_salaries_pk PRIMARY KEY (salary);
16 ALTER TABLE top_salaries
17     ADD CONSTRAINT top_salaries_emp_cnt_gt0 CHECK (emp_cnt > 0);
18
19 TRUNCATE TABLE top_salaries;
20
21 -- 2
22 CREATE OR REPLACE PROCEDURE InsertTopSalaries(
23     pSalary IN NUMBER,
24     pEmp_cnt IN NUMBER)
25 IS
26 BEGIN
27     INSERT INTO top_salaries (salary, emp_cnt, createdBy, dateCreated,
28     ↪ modifiedBy, dateModified)
29     VALUES (pSalary, pEmp_cnt, USER, CURRENT_TIMESTAMP, USER,
30     ↪ CURRENT_TIMESTAMP);
31 END;
32
33 -- 3
34 DECLARE
35     num NUMBER(3) := 3;
36     sal employees.salary%TYPE;

```



```

35     vEmp_cnt top_salaries.emp_cnt%TYPE;
36     CURSOR emp_cursor IS
37         SELECT salary, COUNT(*)
38         FROM employees
39         GROUP BY salary
40         ORDER BY salary DESC
41         FETCH FIRST num ROWS ONLY;
42 BEGIN
43     OPEN emp_cursor;
44     FETCH emp_cursor INTO sal, vEmp_cnt;
45     WHILE emp_cursor%FOUND
46     LOOP
47         InsertTopSalaries(sal, vEmp_cnt);
48         FETCH emp_cursor INTO sal, vEmp_cnt;
49     END LOOP;
50     CLOSE emp_cursor;
51 END;
52
53 ALTER SESSION SET NLS_DATE_FORMAT = 'dd.mm.yyyy hh24:mi:ss';
54 SELECT *
55 FROM top_salaries;
56
57

```

## 2.2 Ergebnisse

	SALARY ▾	EMP_CNT ▾	CREATEDBY ▾	DATECREATED ▾	MODIFIEDBY ▾	DATEMODIFIED ▾
1	24000.00		1 S2310307019	2024-12-09 11:09:22.144390	S2310307019	2024-12-09 11:09:22.144390
2	17000.00		2 S2310307019	2024-12-09 11:09:22.145107	S2310307019	2024-12-09 11:09:22.145107
3	13000.00		1 S2310307019	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

```

1 CREATE OR REPLACE PACKAGE top_salary_pkg AS
2     PROCEDURE GetTopSalaries(pLowSalary IN NUMBER, pHighSalary IN
3         ⇨ NUMBER);
4     PROCEDURE DoubleTopSalaries(pLowSalary IN NUMBER, pHighSalary IN
5         ⇨ NUMBER);
6     PROCEDURE FillTopSalaries(pNum IN NUMBER);
7 END;
8
9 CREATE OR REPLACE PACKAGE BODY top_salary_pkg AS
10     PROCEDURE GetTopSalaries(pLowSalary IN NUMBER, pHighSalary IN NUMBER)

```

```

9      IS
10     CURSOR cTopSalaries IS
11         SELECT 'Salary:' || TO_CHAR(SALARY) || '. EmpCnt:' ||
12             ↳ TO_CHAR(EMP_CNT) ||
13             ↳ '. Erstellt:' || TO_CHAR(CREATEDBY) || '/' ||
14             ↳ TO_CHAR(DECEMBER) ||
15             ↳ '. Geändert:' || TO_CHAR(MODIFIEDBY) || '/' ||
16             ↳ TO_CHAR(DECEMBER) || '.' text
17     FROM top_salaries
18     WHERE SALARY BETWEEN pLowSalary AND pHighSalary
19     ORDER BY SALARY DESC;
20
21 BEGIN
22     FOR vTopSalaries IN cTopSalaries
23     LOOP
24         DBMS_OUTPUT.PUT_LINE(vTopSalaries.text);
25     END LOOP;
26
27 END;
28
29 PROCEDURE DoubleTopSalaries(
30     pLowSalary IN NUMBER,
31     pHighSalary IN NUMBER)
32 IS
33     CURSOR cTopSalaries IS
34         SELECT salary, MODIFIEDBY, DATEMODIFIED
35         FROM top_salaries
36         WHERE salary BETWEEN pLowSalary AND pHighSalary
37         FOR UPDATE OF salary, MODIFIEDBY, DATEMODIFIED NOWAIT;
38
39 BEGIN
40     FOR vTopSalaries IN cTopSalaries
41     LOOP
42         UPDATE top_salaries
43         SET salary = salary * 2,
44             MODIFIEDBY = USER,
45             DATEMODIFIED = CURRENT_TIMESTAMP
46         WHERE CURRENT OF cTopSalaries;
47     END LOOP;
48
49 END;
50
51 PROCEDURE FillTopSalaries(pNum IN NUMBER)
52 IS
53     sal employees.salary%TYPE;
54     vEmp_cnt top_salaries.emp_cnt%TYPE;
55     CURSOR emp_cursor IS
56         SELECT salary, COUNT(*)
57         FROM employees
58         GROUP BY salary
59         ORDER BY salary DESC
60         FETCH FIRST pNum ROWS ONLY;
61
62 BEGIN

```

```

53      OPEN emp_cursor;
54      FETCH emp_cursor INTO sal, vEmp_cnt;
55      WHILE emp_cursor%FOUND
56          LOOP
57          InsertTopSalaries(sal, vEmp_cnt);
58          FETCH emp_cursor INTO sal, vEmp_cnt;
59      END LOOP;
60      CLOSE emp_cursor;
61  END;
62 END;
63
64 -- Tests und Initialisierung für Aufgabe 3 und 4
65 TRUNCATE TABLE top_salaries;
66 ALTER SESSION SET nls_date_format = 'dd.mm.yyyy hh24:mi:ss';
67 BEGIN
68     top_salary_pkg.FillTopSalaries(3);
69     dbms_output.put_line('.....');
70     top_salary_pkg.GetTopSalaries(0, 100000);
71     dbms_output.put_line('.....');
72     top_salary_pkg.DoubleTopSalaries(11000, 13500);
73     dbms_output.put_line('.....');
74     top_salary_pkg.GetTopSalaries(0, 100000);
75     dbms_output.put_line('.....');
76     COMMIT ;
77 END;
78
79 BEGIN
80     top_salary_pkg.GetTopSalaries(0, 100000);
81 END;

```

Für Aufgabe 3.3 und 3.4 wurde folgendes SQL-Statement verwendet:

```

1  -- test in different sessions
2  BEGIN
3      top_salary_pkg.GetTopSalaries(0, 100000);
4      top_salary_pkg.DoubleTopSalaries(0, 100000);
5      top_salary_pkg.GetTopSalaries(0, 100000);
6
7      -- only execute this in 1 session
8      COMMIT;
9  END;

```

## 3.2 Ergebnisse

```
.....
Salary:24000. EmpCnt:1. Erstellt:S2310307019/09-DEC-24 11.16.00.565822 AM. Geändert:S2310307019/09-DEC-24 11.16.00.565822 AM.
Salary:17000. EmpCnt:2. Erstellt:S2310307019/09-DEC-24 11.16.00.566651 AM. Geändert:S2310307019/09-DEC-24 11.16.00.566651 AM.
Salary:13000. EmpCnt:1. Erstellt:S2310307019/09-DEC-24 11.16.00.566684 AM. Geändert:S2310307019/09-DEC-24 11.16.00.566684 AM.
.....
Salary:26000. EmpCnt:1. Erstellt:S2310307019/09-DEC-24 11.16.00.566684 AM. Geändert:S2310307019/09-DEC-24 11.16.00.579697 AM.
Salary:24000. EmpCnt:1. Erstellt:S2310307019/09-DEC-24 11.16.00.565822 AM. Geändert:S2310307019/09-DEC-24 11.16.00.565822 AM.
Salary:17000. EmpCnt:2. Erstellt:S2310307019/09-DEC-24 11.16.00.566651 AM. Geändert:S2310307019/09-DEC-24 11.16.00.566651 AM.
.....
S2310307019> BEGIN
                top_salary_pkg.GetTopSalaries(0, 100000);
            END;
[2024-12-09 12:16:01] completed in 52 ms
Salary:26000. EmpCnt:1. Erstellt:S2310307019/09-DEC-24 11.16.00.566684 AM. Geändert:S2310307019/09-DEC-24 11.16.00.579697 AM.
Salary:24000. EmpCnt:1. Erstellt:S2310307019/09-DEC-24 11.16.00.565822 AM. Geändert:S2310307019/09-DEC-24 11.16.00.565822 AM.
Salary:17000. EmpCnt:2. Erstellt:S2310307019/09-DEC-24 11.16.00.566651 AM. Geändert:S2310307019/09-DEC-24 11.16.00.566651 AM.
```

Abbildung 6: Ergebnis Aufgabe 3.1-3.2

### 3.2.1 Ohne NOWAIT und COMMIT

Die erste Session läuft durch, die anderen warten, bis die Transaktion der ersten Session beendet ist, also in dem Beispiel für immer.

```
-- only execute this in 1 session
--COMMIT;

END;
[2024-12-12 18:34:41] completed in 56 ms
Salary:26000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.34.29.755809 PM. Geändert:S2310307019/12-DEC-24 05.34.29.758639 PM.
Salary:24000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.34.29.755259 PM. Geändert:S2310307019/12-DEC-24 05.34.29.755259 PM.
Salary:17000. EmpCnt:2. Erstellt:S2310307019/12-DEC-24 05.34.29.755766 PM. Geändert:S2310307019/12-DEC-24 05.34.29.755766 PM.
Salary:52000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.34.29.755809 PM. Geändert:S2310307019/12-DEC-24 05.34.41.956631 PM.
Salary:48000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.34.29.755259 PM. Geändert:S2310307019/12-DEC-24 05.34.41.956607 PM.
Salary:34000. EmpCnt:2. Erstellt:S2310307019/12-DEC-24 05.34.29.755766 PM. Geändert:S2310307019/12-DEC-24 05.34.41.956509 PM.
```

Abbildung 7: Ergebnis Aufgabe 3.3 ohne COMMIT

### 3.2.2 Ohne NOWAIT, mit COMMIT

Da die Transaktion der ersten Session beendet ist, kann die zweite Session die Transaktion durchführen.

```
-- only execute this in 1 session
COMMIT;

END;
[2024-12-12 18:36:16] completed in 91 ms
Salary:26000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.36.10.625915 PM. Geändert:S2310307019/12-DEC-24 05.36.10.628462 PM.
Salary:24000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.36.10.625368 PM. Geändert:S2310307019/12-DEC-24 05.36.10.625368 PM.
Salary:17000. EmpCnt:2. Erstellt:S2310307019/12-DEC-24 05.36.10.625875 PM. Geändert:S2310307019/12-DEC-24 05.36.10.625875 PM.
Salary:52000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.36.10.625915 PM. Geändert:S2310307019/12-DEC-24 05.36.10.623226 PM.
Salary:48000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.36.10.625368 PM. Geändert:S2310307019/12-DEC-24 05.36.10.623205 PM.
Salary:34000. EmpCnt:2. Erstellt:S2310307019/12-DEC-24 05.36.10.625875 PM. Geändert:S2310307019/12-DEC-24 05.36.10.623121 PM.

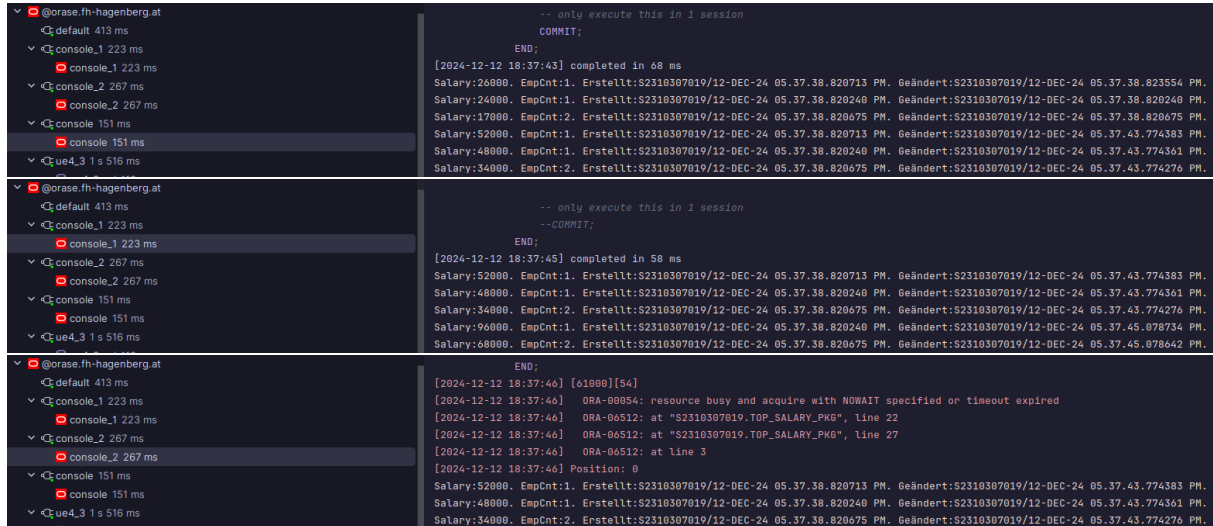
-- only execute this in 1 session
--COMMIT;

END;
[2024-12-12 18:36:18] completed in 26 ms
Salary:52000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.36.10.625915 PM. Geändert:S2310307019/12-DEC-24 05.36.10.623226 PM.
Salary:48000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.36.10.625368 PM. Geändert:S2310307019/12-DEC-24 05.36.10.623205 PM.
Salary:34000. EmpCnt:2. Erstellt:S2310307019/12-DEC-24 05.36.10.625875 PM. Geändert:S2310307019/12-DEC-24 05.36.10.623121 PM.
Salary:96000. EmpCnt:1. Erstellt:S2310307019/12-DEC-24 05.36.10.625368 PM. Geändert:S2310307019/12-DEC-24 05.36.10.159449 PM.
Salary:68000. EmpCnt:2. Erstellt:S2310307019/12-DEC-24 05.36.10.625875 PM. Geändert:S2310307019/12-DEC-24 05.36.10.159443 PM.
```

Abbildung 8: Ergebnis Aufgabe 3.3 mit COMMIT

### 3.2.3 Mit NOWAIT

Die erste Session führt einen Commit aus, dadurch ist die Transaktion beendet und die zweite Session läuft weiterhin problemlos durch. Die dritte Session wartet aber jetzt nicht mehr, sondern gibt eine Fehlermeldung aus.



```
-- only execute this in 1 session
COMMIT;

END;
[2024-12-12 18:37:43] completed in 68 ms
Salary:26000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820713 PM, Geändert:S2310307019/12-DEC-24 05.37.38.823554 PM.
Salary:24000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820240 PM, Geändert:S2310307019/12-DEC-24 05.37.38.820240 PM.
Salary:17000, EmpCnt:2, Erstellt:S2310307019/12-DEC-24 05.37.38.820675 PM, Geändert:S2310307019/12-DEC-24 05.37.38.820675 PM.
Salary:52000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820713 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774383 PM.
Salary:48000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820240 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774361 PM.
Salary:34000, EmpCnt:2, Erstellt:S2310307019/12-DEC-24 05.37.38.820675 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774276 PM.

-- only execute this in 1 session
--COMMIT;

END;
[2024-12-12 18:37:45] completed in 58 ms
Salary:52000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820713 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774383 PM.
Salary:48000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820240 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774361 PM.
Salary:34000, EmpCnt:2, Erstellt:S2310307019/12-DEC-24 05.37.38.820675 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774276 PM.
Salary:96000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820240 PM, Geändert:S2310307019/12-DEC-24 05.37.45.078734 PM.
Salary:68000, EmpCnt:2, Erstellt:S2310307019/12-DEC-24 05.37.38.820675 PM, Geändert:S2310307019/12-DEC-24 05.37.45.078642 PM.

END;
[2024-12-12 18:37:46] [61000][54]
[2024-12-12 18:37:46] ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired
[2024-12-12 18:37:46] ORA-06512: at "S2310307019.TOP_SALARY_PKG", line 22
[2024-12-12 18:37:46] ORA-06512: at "S2310307019.TOP_SALARY_PKG", line 27
[2024-12-12 18:37:46] ORA-06512: at line 3
[2024-12-12 18:37:46] Position: 0
Salary:52000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820713 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774383 PM.
Salary:48000, EmpCnt:1, Erstellt:S2310307019/12-DEC-24 05.37.38.820240 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774361 PM.
Salary:34000, EmpCnt:2, Erstellt:S2310307019/12-DEC-24 05.37.38.820675 PM, Geändert:S2310307019/12-DEC-24 05.37.43.774276 PM.
```

Abbildung 9: Ergebnis Aufgabe 3.4