Ausarbeitung UE06

PL/SQL Grundlagen

Anmerkung: Dieses Dokument ist grauenvoll formattiert. Ich bin vor kurzem auf Linux umgestiegen und bin noch nicht so vertraut mit den Werkzeugen und lerne noch bzw. suche nach Alternativen. Ich bitte um Entschuldigung.

1.3 Tests

	salary
1	24000.00
2	17000.00
3	13000.00
4	12000.00
5	11000.00
6	10500.00
7	9000.00
8	8600.00
9	8300.00
10	7000.00

Illustration 2: p_num = 10 führt zu 10 Tupeln.

salary

Illustration 1: p_num = 0 und p_num = -1 führen zu leerer Ergebnisrelation

	salary
1	24000.00
2	17000.00
3	13000.00
4	12000.00
5	11000.00
6	10500.00
7	9000.00
8	8600.00
9	8300.00
10	7000.00
11	6000.00
12	5800.00
13	4400.00
14	4200.00
15	3500.00
16	3100.00
17	2600.00
18	2500.00

Illustration 3: p_num = 30 führt zur Übernahme der der ganzen "Ergebnisrelation des Cursors".

1.4

```
DROP TABLE top_salaries;
ALTER TABLE top_salaries
    ADD (emp_cnt NUMBER DEFAULT 1 NOT NULL)
    ADD CONSTRAINT emp_cnt_gt_0 CHECK(emp_cnt > 0)
    ADD CONSTRAINT top_salaries_pk PRIMARY KEY(salary);
```

1.5

```
DECLARE
  -- employee
  num NUMBER(3) := &p_num;
  top_sal top_salaries%ROWTYPE;
  -- employee cursor, highest salary first
  CURSOR emp_cursor IS
    SELECT salary, COUNT(*)
    FROM employees
    GROUP BY salary
    ORDER BY salary DESC;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO top_sal;
  -- for as long as more rows should be read
  -- and while there are still more rows to process
  WHILE emp_cursor%ROWCOUNT <= num AND emp_cursor%FOUND LOOP
    -- add dat salary to the top_salaries table
    INSERT INTO top_salaries (salary, emp_cnt)
    VALUES (top_sal.salary, top_sal.emp_cnt);
    -- fetch next tuple
    FETCH emp_cursor INTO top_sal;
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

	salary	emp_cnt
1	24000.00	1
2	17000.00	2
3	13000.00	1
4	12000.00	1
5	11000.00	1

Illustration 4: p_num = 5

PL/SQL Prozeduren

2.1

```
DELETE FROM top_salaries;
ALTER TABLE top_salaries
  ADD (created_by    VARCHAR2(50)  DEFAULT ''      NOT NULL)
  ADD (date_created  DATE           DEFAULT SYSDATE NOT NULL)
  ADD (modified_by   VARCHAR2(50)  DEFAULT ''      NOT NULL)
  ADD (date_modified DATE           DEFAULT SYSDATE NOT NULL);
DECLARE
  -- employee
  num NUMBER(3) := &p_num;
  -- salary
  sal employees.salary%TYPE;
  cnt top_salaries.emp_cnt%TYPE;
  -- employee cursor, highest salary first
```

```
  CURSOR emp_cursor IS
    SELECT salary, COUNT(*)
    FROM employees
    GROUP BY salary
    ORDER BY salary DESC;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO sal, cnt;
  -- for as long as more rows should be read
  -- and while there are still more rows to process
  WHILE emp_cursor%ROWCOUNT <= num AND emp_cursor%FOUND LOOP
    -- add dat salary to the top_salaries table
    INSERT INTO top_salaries (salary, emp_cnt, date_created, created_by, date_modified,
modified_by)
    VALUES (sal, cnt, SYSDATE, USER, SYSDATE, USER);
    -- fetch next tuple
    FETCH emp_cursor INTO sal, cnt;
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

	salary	emp_cnt	created_by	date_created	modified_by	date_modified
1	24000.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
2	17000.00	2	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
3	13000.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
4	12000.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
5	11000.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
6	10500.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
7	9000.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
8	8600.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
9	8300.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41
10	7000.00	1	S1710307099	2018-11-13 21:59:41	S1710307099	2018-11-13 21:59:41

Illustration 5: p_num = 10

2.2

```
CREATE OR REPLACE PROCEDURE InsertTopSalaries (pSalary IN NUMBER, pEmp_cnt IN NUMBER)
  IS
  BEGIN
    INSERT INTO top_salaries (salary, emp_cnt, date_created, created_by, date_modified,
modified_by)
    VALUES (psalary, pEmp_cnt, SYSDATE, USER, SYSDATE, USER);
  END;
  /
```

2.3

```
DECLARE
  -- employee
  num NUMBER(3) := &p_num;
  -- salary
  sal employees.salary%TYPE;
  cnt top_salaries.emp_cnt%TYPE;
  -- employee cursor, highest salary first
  CURSOR emp_cursor IS
    SELECT salary, COUNT(*)
    FROM employees
    GROUP BY salary
    ORDER BY salary DESC;
BEGIN
  OPEN emp_cursor;
```

```
  FETCH emp_cursor INTO sal, cnt;
  -- for as long as more rows should be read
  -- and while there are still more rows to process
  WHILE emp_cursor%ROWCOUNT <= num AND emp_cursor%FOUND LOOP
    -- add dat salary to the top_salaries table
    InsertTopSalaries(sal, cnt);
    -- fetch next tuple
    FETCH emp_cursor INTO sal, cnt;
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

	salary	emp_cnt	created_by	date_created	modified_by	date_modified
1	24000.00	1	S1710307099	2018-11-13 22:05:27	S1710307099	2018-11-13 22:05:27
2	17000.00	2	S1710307099	2018-11-13 22:05:27	S1710307099	2018-11-13 22:05:27
3	13000.00	1	S1710307099	2018-11-13 22:05:27	S1710307099	2018-11-13 22:05:27

Illustration 6: p_num = 3

3. Performance-Optimierung

```
DECLARE
  starttime NUMBER;
  total NUMBER;
BEGIN
  starttime := DBMS_UTILITY.GET_TIME();
  UPDATE my_payment mp
  SET penalty = amount * 1.15
  WHERE EXISTS (SELECT r.rental_id
          FROM rental r
                 INNER JOIN inventory i ON r.inventory_id = i.inventory_id
                 INNER JOIN film f ON i.film_id = f.film_id
          WHERE r.rental_id = mp.rental_id AND
            CEIL(return_date - rental_date) > f.rental_duration);
  total := DBMS_UTILITY.GET_TIME() - starttime;
  DBMS_OUTPUT.PUT_LINE('PL/SQL WITHOUT LOOP: ' || total / 100 || ' seconds');
END;
/
```

```
[2018-11-13 22:08:05] completed in 280 ms
[2018-11-13 22:08:05] PL/SQL WITHOUT LOOP: .26 seconds
```

Illustration 7: Optimierte Laufzeit

4. Multiple Choice

1. PL/SQL eignet sich gut um...

B: SQL-Anweisungen in Verbindung mit Schleifen und Bedingungen auszuführen.

C: wiederkehrende Aufgaben auszuführen.

2. In PL/SQL...

C: sind SQL-Funktionen (zB Datum) ebenfalls verfügbar.

3. Wenn SQL-Anweisungen in einem PL/SQL-Block verwendet werden…

B: sind spezielle Schlüsselwörter (INTO, …) für die Speicherung eines Ergebnisses notwendig

D: können diese mit anderen PL/SQL-Konstrukten gemischt werden.

4. Welche Aussagen sind wahr?

A: Liefert ein SQL-Statement mehrere Ergebniszeilen, ist ein Cursor notwendig.

B: Eine Variable kann auch als „NOT NULL“ deklariert werden.

E: Mit PL/SQL soll möglichst viele Business-Logik in die Datenbank gebracht werden. (Stimmt großteils; Zentralisierung der Logik hilft um Applikationen auf verschiedenen Platformen eine gemeinsame Schnittstelle zu bieten.)

F: PL/SQL kann auch Java-Code ausführen.

G: IN- und OUT-Parameter einer Prozedur können einen Default-Wert besitzen.