

Abgabetermin: 1.12.2021

<input type="checkbox"/> DES3UEG1: Niklas	Name	Angelis Angelos	Aufwand in h	5
<input checked="" type="checkbox"/> DES3UEG2: Niklas				
<input type="checkbox"/> DES3UEG3: Traxler	Punkte		Kurzzeichen Tutor	

Ziel dieser Übung ist die Vertiefung des Trigger-Konzepts und die praktische Anwendung der unterschiedlichen Typen.

1. Trigger (Human Resources) (10 Punkte – 3+2+1+1+3)

Die Zeilen in der Tabelle JOBS speichern ein Höchstgehalt und ein Mindestgehalt, die für verschiedene JOB_ID-Werte zulässig sind. Erstellen Sie in PL/SQL ein Programm, das sicherstellt, dass bei INSERT- und UPDATE-Operationen eines Angestellten, Gehalt und die Provision (commission_pct) das Mindestgehalt nicht unterschreiten. Beachten Sie NULL-Werte!

1. Erstellen Sie eine Prozedur mit dem Namen CHECK_SALARY mit folgenden Parametern:

- die Job-Kennung eines Angestellten
- Gehalt (inkl. Provision).

Verwenden Sie die Job-Kennung, um das Mindestgehalt für den angegebenen Job zu finden. Wenn der Gehaltsparameter unter dem Mindestgehalt liegt, soll eine benutzerdefinierte Exception mit folgender Mitteilung ausgelöst werden:

„Invalid salary <sal>. Salary too low for job <jobid>.“

Ersetzen Sie die Elemente in der Fehlermeldung durch die entsprechenden Werte. Ist das gegebene Gehalt größer als das Höchstgehalt, wird das Höchstgehalt angepasst. Testen Sie die Prozedur, indem Sie für die Job-Kennung ‚ST_CLERK‘ überprüfen, ob ein Gehalt von 1.900,- möglich ist.

2. Erstellen Sie für die Tabelle EMPLOYEES einen Trigger mit dem Namen CHECK_SALARY_TRG, der vor einer INSERT- oder UPDATE-Operation in einer Zeile ausgelöst wird. Der Trigger soll die Prozedur CHECK_SALARY aufrufen, um die in der Aufgabe 1 definierte Regel auszuführen. Der Trigger soll die (neue) Job-Kennung und das Gehalt (inkl. Provision) an die Prozedur übergeben.
3. Testen Sie CHECK_SAL_TRG anhand der folgenden Fälle: Aktualisieren Sie das Gehalt von employee_id = 100 auf 50 000 bzw. 10 000. Was passiert und warum?
4. Welches Problem tritt auf, wenn Sie den Trigger in Aufgabe 1.2 auch überprüfen lassen, ob das (neue) Gehalt eines Mitarbeiters das durchschnittliche Abteilungsgehalt nicht um 50% übersteigt? Was wäre eine mögliche Lösung? Antworten Sie kurz in Textform.
5. Erweitern Sie die Tabelle EMPLOYEES und fügen Sie zwei Logging-Felder hinzu (date_modified, user_modified). Erstellen Sie einen Trigger LOG_EMPLOYEES, der diese Felder aktuell hält. Testen Sie den Trigger ausführlich (führen Sie nach den Tests ein ROLLBACK aus, um Ihre Datenbasis nicht zu verändern). Anschließend entfernen Sie die zwei Spalten wieder aus Ihrer EMPLOYEES-Tabelle.

2. INSTEAD OF-Trigger

(9 Punkte – 3+5+1)

INSTEAD OF-Trigger werden ausschließlich für Sichten eingesetzt, um Daten zu ändern, bei denen eine DML-Anweisung für eine Sicht abgesetzt wird, die implizit nicht aktualisierbar ist. Diese Trigger werden INSTEAD OF-Trigger genannt, da der Datenbankserver im Gegensatz zu anderen Trigger-Typen nicht die auslösende Anweisung ausführt, sondern den Trigger auslöst. Mit diesem Trigger werden INSERT-, UPDATE- oder DELETE-Operationen direkt für die zu Grunde liegenden Basistabellen durchgeführt. Sie können INSERT-, UPDATE- oder DELETE-Anweisungen für eine Sicht erstellen, und der INSTEAD OF-Trigger arbeitet unsichtbar im Hintergrund und sorgt für die Ausführung der gewünschten Aktionen.

Führen Sie folgendes Skript aus, um die Basistabellen für die Aufgabe zu erstellen.

```
CREATE TABLE new_emps AS
  SELECT employee_id, last_name, salary, department_id
  FROM employees;

CREATE TABLE new_locs AS
  SELECT l.location_id, l.city, l.country_id
  FROM locations l;

CREATE TABLE new_depts AS
  SELECT d.department_id, d.department_name,
         AVG(e.salary) AS dept_sal, d.location_id
  FROM employees e INNER JOIN departments d
                   ON (e.department_id = d.department_id)
  GROUP BY d.department_id, d.department_name, d.location_id;

CREATE TABLE new_countries AS
  SELECT c.country_id, c.country_name, COUNT(e.employee_id) AS c_emps
  FROM countries c INNER JOIN locations l ON (l.country_id = c.country_id)
  INNER JOIN departments d ON (d.location_id = l.location_id)
  INNER JOIN employees e ON (e.department_id = d.department_id)
  GROUP BY c.country_id, c.country_name;
```

Skript UE08_02_01.sql

1. Erstellen Sie eine Sicht emp_details basierend auf den im Skript UE08_02_01.sql erstellten Tabellen NEW_EMPS, NEW_LOCS, NEW_DEPTS und NEW_COUNTRIES mit folgenden Spalten: employee_id, last_name, salary, department_id, department_name, dept_sal, location_id, city, country_name und c_emps und verknüpfen Sie die Tabellen über entsprechenden IDs. Stellen Sie mit Hilfe des DataDictionary (USER_UPDATABLE_COLUMNS) fest, auf welchen Spalten der Sicht die Operationen UPDATE, INSERT oder DELETE erlaubt sind.
2. Legen Sie nun für die Sicht emp_details einen INSTEAD OF-Trigger an, um DML-Operationen auf dieser Sicht zu erlauben. Folgende Funktionalitäten sind gefordert:
 - a. Bei einem DELETE wird der Satz aus new_emps gelöscht und das dept_sal angepasst. Passen Sie die Anzahl der Mitarbeiter im entsprechenden Land an.
 - b. Bei einem INSERT wird ein neuer Mitarbeiter in new_emps angelegt und dept_sal ebenfalls angepasst. Passen Sie die Anzahl der Mitarbeiter entsprechend an.
 - c. Bei einem UPDATE auf salary aktualisieren Sie neben den salary des Mitarbeiters auch den Abteilungsdurchschnitt dept_sal.
 - d. Bei einem UPDATE auf die department_id aktualisieren Sie neben der Abteilungsnummer des Mitarbeiters auch den Abteilungsdurchschnitt dept_sal und passen Sie die Anzahl der Mitarbeiter in den betroffenen Ländern an.
3. Überprüfen Sie die implementierte Funktionalität des INSTEAD OF-Triggers mit mind. einem INSERT-, einem UPDATE und einem DELETE-Statement.

3. Trigger für Systemereignisse

(5 Punkte – 1+3+1)

1. Erstellen Sie eine Tabelle USER_LOGGING mit den Feldern session_id, login_time, db_user, os_user, ip und host_name. Wählen Sie geeignete Datentypen.
2. Erstellen Sie für das Schema einen Systemtrigger, der pro Session beim Anmelden einen Datensatz einfügt. Verwenden Sie die Funktion SYS_CONTEXT (sh. Oracle-Dokumentation) mit den entsprechenden Parametern um die Session-ID, die IP-Adresse, den Betriebssystem-User und den Host-Namen (Bezeichnung des verbundenen Rechners) zu ermitteln. Schreiben Sie diese Werte gemeinsam mit dem angemeldeten Datenbank-User und eines aktuellen Zeitstempels in die Tabelle.
3. Melden Sie sich bei der Datenbank ab und wieder an. Wiederholen Sie diese Schritte (wenn möglich) auch von einem anderen Rechner aus. Zeigen Sie den Inhalt Ihrer Tabelle an.

Aufgabe 1)

```

CREATE OR REPLACE PROCEDURE CHECK_SALARY (jId Jobs.Job_id%TYPE, sal
Jobs.Min_Salary%TYPE)
IS
miSal Jobs.Min_Salary%TYPE;
BEGIN
SELECT Min_Salary
INTO miSal
FROM Jobs
WHERE Job_ID = UPPER(jId);
IF sal < miSal THEN
    RAISE_APPLICATION_ERROR( -20001, 'Invalid salary ' || sal || '. Salary
too low for job ' || jId || '.' );
ELSE
    UPDATE Jobs
    SET Min_Salary = sal
    WHERE Job_ID = UPPER(jId);
END IF;
END;
/
CALL CHECK_SALARY('ST_CLERK', 19000);
ROLLBACK ;

```

```

CREATE OR REPLACE TRIGGER CHECK_SALARY_TRG
BEFORE INSERT OR UPDATE ON Employees
FOR EACH ROW
BEGIN
CHECK_SALARY(:NEW.Job_ID, :NEW.Salary);
END;
/

```

```

UPDATE EMPLOYEES
SET SALARY = 10000
WHERE EMPLOYEE ID = 100;

```

```

[2021-12-01 12:36:03] [72000][20001]
[2021-12-01 12:36:03]   ORA-20001: Invalid salary 10000. Salary too low for job AD_PRES.
[2021-12-01 12:36:03]   ORA-06512: in "S2010307048.CHECK_SALARY", Zeile 10
[2021-12-01 12:36:03]   ORA-06512: in "S2010307048.CHECK_SALARY_TRG", Zeile 2
[2021-12-01 12:36:03]   ORA-04088: Fehler bei der Ausführung von Trigger 'S2010307048.CHECK_SALARY_TRG'
[2021-12-01 12:36:03] Position: 7

```

-- 100 ist king 10000 ist zu wenig für ceo 50000 geht wieder.

```

ALTER TABLE EMPLOYEES ADD
(dateModified DATE,
user_modified VARCHAR2(30));

```

```

CREATE OR REPLACE TRIGGER LOG_EMPLOYEES
BEFORE INSERT OR UPDATE ON Employees
FOR EACH ROW
BEGIN
:new.user_modified := USER;
:new.dateModified := SYSDATE;
END;
/

```

```
INSERT INTO Employees (EMPLOYEE_ID, SALARY, JOB_ID, LAST_NAME, EMAIL, HIRE_DATE)
VALUES (300, 40000, 'SA_REP', 'MUSTERMANN', 'MUSTERMAIL', SYSDATE);
```

```
SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID = 300;
```

ONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	DATE_MODIFIED	USER_MODIFIED
1 >	2021-12-01 12:37:18	SA_REP	40000.00	<null>	<null>	<null>	2021-12-01 12:37:18	S2010307048

```
ROLLBACK;
```

```
ALTER TABLE EMPLOYEES
DROP COLUMN dateModified;
ALTER TABLE EMPLOYEES
DROP COLUMN user_modified;
```

Aufgabe2)

```
CREATE TABLE new_emps AS
SELECT employee_id, last_name, salary, department_id
FROM employees;
```

```
CREATE TABLE new_locs AS
SELECT l.location_id, l.city, l.country_id
FROM locations l;
```

```
CREATE TABLE new_depts AS
SELECT d.department_id, d.department_name,
       AVG(e.salary) AS dept_sal, d.location_id
FROM employees e INNER JOIN departments d
ON (e.department_id = d.department_id)
GROUP BY d.department_id, d.department_name, d.location_id;
```

```
CREATE TABLE new_countries AS
SELECT c.country_id, c.country_name, COUNT(e.employee_id) AS c_emps
FROM countries c INNER JOIN locations l ON (l.country_id = c.country_id)
INNER JOIN departments d ON (d.location_id = l.location_id)
INNER JOIN employees e ON (e.department_id = d.department_id)
GROUP BY c.country_id, c.country_name;
```

```
CREATE VIEW emp_details AS
SELECT employee_id, last_name, salary, department_id, department_name,
dept_sal, location_id, city, country_name, c_emps
FROM new_emps
JOIN new_depts USING (department_id)
JOIN new_locs USING (location_id)
JOIN new_countries USING (country_id);
```

```
SELECT * FROM emp_details;
```

```
SELECT column_name, updatable, insertable, deletable
FROM USER_UPDATABLE_COLUMNS
WHERE table_name = upper('emp_details');
```

```

CREATE OR REPLACE TRIGGER emp_details_trigger
  INSTEAD OF DELETE OR INSERT OR UPDATE ON emp_details
  FOR EACH ROW
BEGIN
  IF DELETING THEN
    DELETE FROM new_emps
    WHERE employee_id = :OLD.employee_id;

    UPDATE new_depts
    SET dept_sal = (
      SELECT AVG(salary)
      FROM employees JOIN departments USING (department_id)
      WHERE department_id = :OLD.department_id
    )
    WHERE department_id = :OLD.department_id;

    UPDATE new_countries
    SET c_emps = :OLD.c_emps - 1
    WHERE country_id = (SELECT country_id
                        FROM new_depts
                        JOIN new_locs USING (location_id)
                        WHERE department_id = :OLD.department_id);

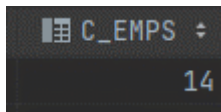
  ELSIF INSERTING THEN
    INSERT INTO new_emps
    VALUES (:NEW.employee_id, :NEW.last_name, :NEW.salary,
:NEW.department_id);

    UPDATE new_depts
    SET dept_sal = dept_sal + :NEW.salary
    WHERE department_id = :NEW.department_id;

    UPDATE new_countries
    SET c_emps = :OLD.c_emps + 1
    WHERE country_id = (SELECT country_id
                        FROM new_depts
                        JOIN new_locs USING (location_id)
                        WHERE department_id = :OLD.department_id);

  END IF;
END;
/

```

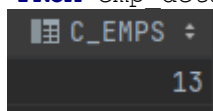


C_EMPS
14

Bevor:

```
DELETE FROM emp_details WHERE EMPLOYEE_ID = 104;
```

```
SELECT * FROM emp_details;
```



C_EMPS
13

Dannach:

```
ROLLBACK;
```

Aufgabe3)

```

CREATE TABLE user_logging (
    sessionid NUMBER PRIMARY KEY,
    login_time DATE,
    db_user VARCHAR2(128),
    os_user varchar2(128),
    ip varchar2(256),
    host_name varchar2(256)
);

SELECT SYS_CONTEXT('userenv', 'ip_address') FROM dual;

CREATE OR REPLACE TRIGGER user_logging_trg
    AFTER LOGON ON SCHEMA
BEGIN
    INSERT INTO user_logging (sessionid, login_time, db_user, os_user, ip,
    host_name)
    VALUES (SYS_CONTEXT('userenv', 'sid'), SYSDATE, USER,
    SYS_CONTEXT('userenv', 'os_user'), SYS_CONTEXT('userenv', 'ip_address'),
    SYS_CONTEXT('userenv', 'host'));
END;
/
SELECT * from user_logging;

```

	SESSIONID	LOGIN_TIME	DB_USER	OS_USER	IP	HOST_NAME
1	365	2021-12-01 11:25:49	S2010307048	angel	193.170.132.230	DESKTOP-RJ13KU0
2	435	2021-12-01 11:26:41	S2010307048	angel	193.170.132.230	DESKTOP-RJ13KU0