DES3UE Datenbanksysteme WS 2018 Übung 8

Abgabetermin: 28.11.2018, 13:30 Uhr
Abgabeform elektronisch

☐	**DES31UE Niklas**	**Name** _____**Niklas Vest**_____	**Aufwand in h** __**4**____
☐	**DES32UE Niklas**		
☒	**DES33UE Traxler**	**Punkte** _____	**Kurzzeichen Tutor** _____

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

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

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

1. Erstellen Sie eine Prozedur mit dem Namen CHECK_SALARY mit folgenden Parametern: einen Parameter für die Job-Kennung eines Angestellten und einen Parameter für Gehalt (inkl. Provision). Die Prozedur verwendet die Job-Kennung, um das Höchstgehalt für den angegebenen Job zu finden. Wenn der Gehaltsparameter nicht innerhalb dieses Kriteriums liegt (Höchstgehalt), soll eine benutzerdefinierte Exception mit einer entsprechenden Fehlermeldung ausgelöst werden: „Invalid salary <sal>. Salary too high for job <jobid>." Ersetzen Sie die verschiedenen Elemente in der Fehlermeldung durch die entsprechenden Werte. Ist das gegebene Gehalt kleiner als das Mindestgehalt, wird das Mindestgehalt angepasst. Testen Sie die Prozedur, indem Sie für die Job-Kennung ‚AD_PRES' überprüfen, ob Gehalt inkl. Provision 50.000,- nicht übersteigen.

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 an die Prozedur übergeben. Speichern Sie den Trigger in einer Datei.

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 Gehalt eines Mitarbeiters das Gehalt seines Managers nicht 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).

2. INSTEAD OF-Trigger (3+5+1 = 9 Punkte)

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 (1+3+1 = 5 Punkte)

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 (zB Labor-Rechnern FH). Zeigen Sie den Inhalt Ihrer Tabelle an.