Datenbanken 1 Praktikum 12

Aufgabe 1 – Statische und dynamische Integritätsregeln

Die Datenbank beinhaltet die folgenden 8 Integritätsregeln.
Diese dynamischen Integritätsregeln sollen durch Trigger umgesetzt werden. Erstellen Sie eine tabellarische Übersicht durch welche Art von Trigger auf welchen Tabellen die Integritätsregeln implementiert werden können. Ergänzen Sie auch die Events durch welche die Trigger ausgelöst werden. Dies sind dann die Änderungsoperationen, mit denen Sie nach der Implementierung die Trigger testen können. Die erste Integritätsregel ist bereits in der Tabelle als Beispiel eingetragen.

Nr.	Methode	Event	Tabelle	Events zum Testen
1	AFTER	INSERT	Kunde	INSERT INTO Kunde VALUES (…)
2	…	…	…	…

Integritätsregeln:

1) Ein neuer Kunde erhält ein Willkommenspräsent in den Warenkorb.
2) Bestellungen dürfen nicht gelöscht werden.
3) Wird eine Bestellung durch einen Kunden getätigt, so erhält diese automatisch den Status „bestellt“ und das aktuelle Tagesdatum als Wunschtermin.
4) Wenn der Mindestbestellwert von 25 Euro nicht erreicht wird, dann wird die Bestellung abgebrochen.
5) Ist eine Bestellung an den Lieferer übergeben worden, so wird durch den Lagermitarbeiter der Bestellstatus auf „geliefert“ geändert.
6) Der Bestellstatus einer Bestellung kann nur folgendermaßen geändert werden:
   „bestellt“ → „bestaetigt“ → „geliefert“
   Andere Bestellstatusübergänge sind nicht erlaubt und führen zu einem Abbruch der jeweiligen Änderungsoperation.
7) Innerhalb des Monats, in dem ein Kunde Geburtstag hat, erhält dieser einen Geburtstagsrabatt von 10%.
8) Wenn eine neue Bestellung eingefügt wurde, dann wird automatisch der Warenkorbinhalt des jeweiligen Kundens in die Bestellung übernommen und der Warenkorbinhalt des Kunden gelöscht.

Nr.	Methode	Event	Tabelle	Events zum Testen
1	AFTER	INSERT	Kunde	INSERT INTO Kunde VALUES (…)
2	BEFORE	DELETE	Bestellung	DELETE FROM Bestellung WHERE bestellnummer = 1
3	BEFORE	INSERT	Bestellung	INSERT INTO Bestellung (bestellnummer, kundennummer, Bestelldatum, bestellstatus, versandstatus) VALUES (…, …, …, sysdate, ‚bestellt',)
4	BEFORE	INSERT	Bestellung	Kunde 1 hat einen leeren Warenkorb
5	AFTER	UPDATE	Bestellung	UPDATE Bestellung SET bestellstatus = 'versandt' WHERE bestellnummer = 65
6	BEFORE	UPDATE	Bestellung	UPDATE Bestellung SET bestellstatus = '…' WHERE bestellnummer = …
7	BEFORE	INSERT	Bestell-position	INSERT INTO Bestellposition (Position, Bestellnummer, artikelnummer, anzahl, preis, reduktion) VALUES (…)
8	AFTER	INSERT	Bestellung	INSERT INTO Bestellung (bestellnummer, kundennummer, Bestelldatum) VALUES (…, …, now())

Aufgabe 2 - Views

Im Lager wird die Bestellung auf der Basis eines ausgedruckten Lieferscheines zum Versand vorbereitet. Zur Überprüfung müssen dem Lagermitarbeiter sowohl die Kundenadresse, das Rechnungsdatum, den gesamten Rechnungsbetrag und der Bestellstatus der jeweiligen Bestellung einsehbar sein, jedoch nicht die jeweiligen Artikelpreise und der Bestellstatus. Wird die Bestellung versendet, so wird der Status der Bestellung durch einen Lagermitarbeiter auf ‚versandt' gesetzt.

a) Implementieren Sie die zugehörende Benutzersicht und testen Sie dessen Funktion. Ist diese Benutzersicht änderbar?

```
CREATE OR REPLACE VIEW Versand_View
AS
SELECT bestellnummer, kundennummer, nachname, ort, bestelldatum, bestellstatus, SUM(anzahl*preis) AS bestellsumme
FROM kunde NATURAL JOIN bestellung NATURAL JOIN bestellposition NATURAL JOIN artikel
GROUP BY bestellnummer, kundennummer, nachname, ort, datum, bestellstatus;
//Aufgrund der Gruppierung ist diese Benutzersicht nicht updateable.
```

b) Implementieren Sie einen Instead-Of-Trigger, um die Änderungen auf der Benutzersicht ausführen zu können. Überprüfen Sie, ob die Änderungen korrekt auf dieser Benutzersicht ausgeführt werden können.

```
CREATE OR REPLACE TRIGGER Update_Versand_View
INSTEAD OF UPDATE ON VERSAND_VIEW
FOR EACH ROW
BEGIN
        UPDATE bestellung
        SET bestellstatus = :new.bestellstatus
        WHERE idbestellung = :new.idbestellung;
END;
```

Austesten:

```
UPDATE Versand_View
SET bestellstatus = ‚bestaetigt'
WHERE Bestellnummer = 1;
```

Aufgabe 3 – Zugriff auf dieselbe Tabelle aus einem Trigger

Implementieren Sie im Schema in Oracle die Integritätsbedingung, dass ein Artikel an maximal zwei Lagerstandorten gleichzeitig gelagert werden darf.

a) Erstellen Sie einen Trigger, der diese Integritätsregel für das Einfügen von Lagerplätzen sicherstellt. Testen Sie den Trigger mit einer geeigneten Änderungsoperation.

```
CREATE OR REPLACE TRIGGER InsertLagerplatz
BEFORE INSERT ON Lager
FOR EACH ROW
DECLARE     anzahl INTEGER DEFAULT 0;
            nurZweiLagerplaetze EXCEPTION;
BEGIN
    IF :new.anummer IS NOT NULL
    THEN
            SELECT COUNT(*) INTO anzahl
            FROM Lager
            WHERE anummer= :new.anummer;
            IF anzahl >= 2 THEN
                    RAISE nurZweiLagerplaetze;
            END IF;
    END IF;
EXCEPTION
    WHEN nurZweiLagerplaetze
    THEN RAISE_APPLICATION_ERROR (-20500, 'Es gibt bereits zwei Lagerplaetze');
END;
```

Test des Triggers:

```
INSERT INTO Lager (Lagernummer, Standort, ANummer, Lagerbestand)
VALUES (1, ‘INF‘, 1, 3);
INSERT INTO Lager (Lagernummer, Standort, ANummer, Lagerbestand)
VALUES (2, ‘INF‘, 1, 3);
INSERT INTO Lager (Lagernummer, Standort, ANummer, Lagerbestand)
VALUES (1, ‘INF‘, 1, 3);
INSERT INTO Lager (Lagernummer, Standort, ANummer, Lagerbestand)
VALUES (3, ‘INF‘, 1, 3);
```

```
Fehlerbericht -
ORA-20500: Es gibt bereits zwei Lagerplaetze
ORA-06512: in "C##FBPOOL251.INSERTLAGERPLATZ", Zeile 15
ORA-04088: Fehler bei der Ausführung von Trigger
'C##FBPOOL251.INSERTLAGERPLATZ'
```

b) Ändern Sie den Trigger dahingehend, dass der Trigger aus Teilaufgabe a) auch bei Änderungsoperationen ausgeführt wird. Ändern Sie einen Lagerplatz so ab, dass ein Artikel zusätzlich an diesem (dritten) Lagerplatz gelagert wird. Wie ist das Verhalten des DBMS zu erklären?

Änderung der 2. Zeile in:

```
CREATE OR REPLACE TRIGGER InsertLagerplatz
BEFORE INSERT OR UPDATE ON Lager
```

Bei der Ausführung des Updates erfolgt eine Fehlermeldung:

```
Fehlerbericht -
ORA-04091: Tabelle C##FBPOOL251.LAGER wird gerade geändert,
Trigger/Funktion sieht dies möglicherweise nicht
ORA-06512: in "C##FBPOOL251.INSERTLAGERPLATZ", Zeile 6
ORA-04088: Fehler bei der Ausführung von Trigger
'C##FBPOOL251.INSERTLAGERPLATZ'
```

Diese Fehlermeldung ist damit zu erklären, dass im Trigger ein Zugriff auf die Tabelle erfolgt, die den Trigger auslöst.

c) Implementieren Sie eine Lösung, so dass durch eine Hilfstabelle und zwei Trigger die Integritätsregel bei einem Update überprüft wird.

1. Schritt: Änderungen in eine Hilfstabelle schreiben

Erstellen einer temporären Hilfstabelle:

```
CREATE GLOBAL TEMPORARY TABLE TempArtikelnummer(Artikelnummer Integer)
ON COMMIT DELETE ROWS;
```

BEFORE-Trigger überträgt die zu ändernden Daten aus der Tabelle Lager in die Hilfstabelle:

```
CREATE OR REPLACE TRIGGER TRIGGER_Teil1
BEFORE INSERT OR UPDATE ON Lager
FOR EACH ROW
BEGIN
  IF (:new.ANummer IS NOT NULL)
    THEN
      INSERT INTO TempArtikelnummer VALUES (:new.ANummer);
    END IF;
  END;
```

2. Schritt: Die Hilfstabelle mit befehlsorientiertem AFTER-Trigger auswerten
Der Trigger zählt die Anzahl der Lagerplätze zu den in der Hilfstabelle eingefügten Artikelnummern. Die Verarbeitung wird abgebrochen, falls ein Artikel aus der Hilfstabelle mehr als zwei Lagerplätze besitzt.

```
CREATE OR REPLACE TRIGGER Trigger_Teil2
AFTER INSERT OR UPDATE ON Lager
DECLARE
        anzahl INTEGER := 0;
        regel_nur_zwei_Lagerplaetze EXCEPTION;
BEGIN
        SELECT COUNT(*) INTO anzahl
        FROM Tempartikelnummer hilfstabelle
        WHERE
                ( SELECT COUNT(*) FROM Lager l
                WHERE hilfstabelle.Artikelnummer = l.ANummer) > 2;
        IF anzahl > 0 THEN
                RAISE regel_nur_zwei_Lagerplaetze;
        END IF;
EXCEPTION
        WHEN regel_nur_zwei_Lagerplaetze
        THEN raise_application_error (-20500, 'Bereits zwei Lagerplaetze vorhanden');
END;
```

d) Überprüfen Sie, ob die Integritätsregel auch bei Updates eingehalten wird.
Testen des Triggers:

```
SELECT ANummer, Count(*)
FROM Lager
GROUP BY ANummer;
```

	ANUMMER	COUNT(*)
1	1	2
2	6	1
3	2	1
4	4	1
5	5	1
6	8	1
7	3	1
8	7	1

```
UPDATE Lager
SET ANummer=1, Lagerbestand=1
WHERE Lagernummer= 27595;
```

```
Fehlerbericht -
ORA-20500: Bereits zwei Lagerplaetze vorhanden
ORA-06512: in "C##FBPOOL251.TRIGGER_TEIL2", Zeile 16
ORA-04088: Fehler bei der Ausführung von Trigger        'C##FBPOOL251.TRIGGER_TEIL2'
```

Die Fehlermeldung wird durch den Trigger korrekt ausgegeben.