Praktikum 3

Lernziel

- Einblick in die serverseitige Programmierung.
- Kennenlernen der Nützlichkeit von Triggern in einem typischen Anwendungsgebiet.

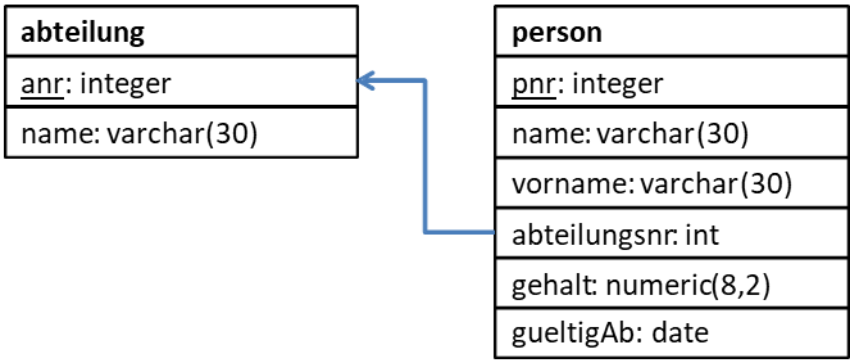
Vorbereitung

Informieren Sie sich über folgende Punkte:

- Was ist ein Trigger und auf welche Weise wird er in PostgreSQL angelegt ([1], insbesondere die Abschnitte 38 und 42, sowie [2])?
- Wie programmiert man in PL/pgSQL Stored Procedures, die von Triggern angestoßen werden (Quellen s.o.)?
- Was sind Views und wie legt man sie an [3]?

Aufgabe 1

Das nachfolgende Datenbankschema soll um eine Tabelle `gehaltshistorie` erweitert werden, in welcher Änderungen am Gehalt einer `person` dokumentiert werden.



Für jeden Änderungsvorgang auf der Tabelle `person` sollen mit Hilfe eines Triggers in einer neuen Tabelle `gehaltshistorie` jeweils die `pnr` der Person, das alte Gehalt, das alte `gueltig`Ab-Datum sowie der Änderungszeitpunkt und der User-Name des Datenbank-Benutzers, der das Gehalt geändert hat, festgehalten werden

Für das Gehalt einer Person sollen folgende Geschäftsregeln gelten:

Regel 1: Das Gehalt einer neu eingefügten oder geänderten Person darf um maximal 50% höher sein als der bisherige Durchschnitt in der Abteilung der Person. (Dies gilt nur, wenn es weitere Personen in der gleichen Abteilung gibt).

Regel 2: Das Gehalt einer Person darf nicht reduziert werden.

Regel 3: Das neue Gehalt einer Person darf höchstens 120% ihres bisherigen Gehalts betragen. Sollte das neue Gehalt höher sein, wird es automatisch auf 120% des bisherigen Gehalts gekappt.

Regel 4: Es dürfen keine rückwirkenden Gehaltsänderungen vorgenommen werden, d.h. das neue gueltigAb-Datum darf nicht zeitlich früher liegen als das bisherige gueltigAb-Datum.

Insert- und Update-Vorgänge, die die obigen Regeln verletzen, sind unter Angabe einer entsprechenden Meldung zurückzuweisen (Regel 1, 2 und 4) bzw. zu korrigieren (Regel 3).

Test

Testen Sie Ihren Trigger mit der folgenden Sequenz von Aufrufen:

```
INSERT INTO Abteilung VALUES (1, 'Vertrieb'), (2,'Entwicklung');
```

```
INSERT INTO Person VALUES
  (1, 'Flasche', 'Frank', 1, 2000, to_date('01.01.2019','DD.MM.YYYY'));
```

OK

```
INSERT INTO Person VALUES
  (2, 'Buechse', 'Bernd', 2, 3200, to_date('01.01.2019','DD.MM.YYYY'));
```

OK

```
INSERT INTO Person VALUES
  (3, 'Dose', 'Doris', 1, 3200, to_date('01.01.2019','DD.MM.YYYY'));
```

Wird zurückgewiesen wegen Regel 1

```
INSERT INTO Person VALUES
  (4, 'Schachtel', 'Susi', 1, 3000, to_date('01.01.2019','DD.MM.YYYY'));
```

OK

```
INSERT INTO Person VALUES
  (3, 'Dose', 'Doris', 1, 3200, to_date('01.01.2019','DD.MM.YYYY'));
```

Jetzt ok wegen veränderten Durchschnitts

```
UPDATE Person
SET gehalt = 3000, gueltigAb = to_date('01.01.2020','DD.MM.YYYY')
  WHERE pnr = 3;
```

Wird zurückgewiesen wegen Regel 2

```
UPDATE Person
SET gehalt = 3400, gueltigAb = to_date('01.01.2018','DD.MM.YYYY')
  WHERE pnr = 3;
```

Wird zurückgewiesen wegen Regel 4

```
UPDATE Person
SET gehalt = 3400, gueltigAb = to_date('01.01.2020','DD.MM.YYYY')
  WHERE pnr = 3;
```

OK → Update und neuer Eintrag in Gehaltshistorie

```
UPDATE Person
SET gehalt = 4500, gueltigAb = to_date('01.01.2021','DD.MM.YYYY')
  WHERE pnr = 3;
```

Wird zurückgewiesen wegen Regel 1

```
UPDATE Person
SET gehalt = 4200, gueltigAb = to_date('01.01.2021','DD.MM.YYYY')
  WHERE pnr = 3;
```

Gehalt wird auf 4080 gekappt wegen Regel 3,
OK → Update und neuer Eintrag in Gehaltshistorie

Hinweise

- Als Primary Key Ihrer Tabelle `gehaltshistorie` sollten Sie ein künstliches Schlüsselattribut vom Typ `SERIAL` nehmen. Dann wird für jeden neuen Eintrag in dieser Tabelle automatisch ein eindeutiger Schlüsselwert generiert.
- Das aktuelle Datum und den Namen des aktuellen Datenbank-Benutzers erhalten Sie über die eingebauten Funktionen `current_date` und `current_user` (benötigen Sie, wenn Sie einen Eintrag in die Tabelle `gehaltshistorie` einfügen);
- Beachten Sie, dass in PL/pgSQL der Body der kompletten Trigger-Funktion in einfache Hochkommata eingeschlossen wird. Wenn Sie daher innerhalb der Trigger-Funktion wieder Hochkommata verwenden wollen (z.B. für Strings), müssen Sie diese escapen (d.h. verdoppeln).
  Beispiel:
  `RAISE NOTICE ''Hier müssen jeweils zwei Hochkommata stehen'';`
- Innerhalb einer Trigger-Funktion können Sie u.a. auf folgende eingebaute Variablen zurückgreifen:
  - `NEW`: enthält Datensatz mit Werten nach Durchführung der Änderung bzw. Einfügeoperation;
  - `OLD`: enthält Datensatz mit den alten Werten vor der Durchführung der Änderung;
  - `TG_OP`: Art der Operation, die den Trigger ausgelöst hat (`INSERT, UPDATE, DELETE`)

Aufgabe 2

Legen Sie SQL-Statements vor, mit dem im Rahmen Ihres Lösungsansatzes folgende Anfragen beantwortet werden können:

- Geben Sie pro Abteilung die Abteilungsnummer (anr), den Namen der Abteilung und die Gesamtanzahl der Gehaltsänderungen in dieser Abteilung aus.
- Definieren Sie einen View `alleGehaelter` mit den Attributen (`pnr, gehalt, gueltigAb`), der die aktuellen und früheren Gehälter aller Personen mit den jeweiligen `gueltigAb`-Daten enthält (Hinweis: UNION).
- Ermitteln Sie unter Verwendung des zuvor definierten Views das Gehalt einer gegebenen Person (`pnr`) zu einem bestimmten Zeitpunkt.

Literatur

[1] PostgreSQL 13 Documentation, Part V. Server Programming. Online verfügbar unter: https://www.postgresql.org/docs/13/server-programming.html

[2] Kapitel 5 der DBS-Vorlesung. Online auf moodle verfügbar unter: https://moodle.hsnr.de/pluginfile.php/588122/course/section/90328/DBS-WS2021-Kap5-ServerProg-m.pdf

[3] Kapitel 6 der DBS-Vorlesung. Online auf moodle verfügbar unter: https://moodle.hsnr.de/pluginfile.php/588122/course/section/90328/DBS-WS2021-Kap6-WeitereDBObjekte-m.pdf