LANDESBERUFSSCHULE 4 SALZBURG

Informatik

VIEW - SICHT

Datenbank MySQL

LBS 4

25.09.2020

Inhalt

Lernziele 3

Einleitung 3

VIEW - Sicht 3

Aufbau einer einfachen VIEW 3

UPDATES mit VIEWS 4

Beispiel 1: 4

Beispiel 2: 4

VIEW – WITH CHECK OPTION 5

Warum soll eine VIEW updatebar sein 5

Lernziele

- Ich kenne den Aufbau einer VIEW
- Ich kann das Konzept einer VIEW erklären
- Ich kann die Verwendung/Anwendung von VIEWS erläutern

Einleitung

Eine VIEW oder auch Sicht genannt, entspricht einer logischen Relation in einer Datenbank. VIEWS sind meistens komplexe Statements, die mit einem Namen in der Datenbank abgespeichert sind. Die Sichten können in normalen SELECT-Anweisungen wiederverwendet werden. Die VIEW wird zur Verwendungszeit berechnet, dadurch werden die Ergebnisse immer auf aktuelle Tabellen angewendet. Sichten enthalten nur Informationen, die auch in den Basistabellen verfügbar sind, die Struktur kann aber anders sein.

VIEW - Sicht

Bei einer View wird eine Abfrage und nicht das Ergebnis gespeichert. Eine VIEW kann grundsätzlich alle Elemente einer SELECT-Anweisung enthalten. Die Aufgabe besteht darin, dem Benutzer einerseits den Zugriff auf die Datenbank zu erleichtern und andererseits den Zugriff auf Tabellen zu beschränken. Genauso wie bei Tabellen, können Benutzer Rechte auf eine VIEW bekommen.

Die Rechteverwaltung soll aber in der Benutzerverwaltung administriert werden!

Aufbau einer einfachen VIEW

Der Aufbau einer VIEW ist ähnlich wie das SELECT-Statement, es muss nur das Schlüsselwort CREATE VIEW verwendet werden.

```
CREATE VIEW VPersonal AS
SELECT Vorname, Nachname, GebDatum AS Geburtsdatum
FROM personal;
```

CREATE VIEW erzeugt eine neue Sicht mit dem Namen ‚Vpersonal' und das Schlüsselwort AS weist das SELECT-Statement der VIEW zu.

Diese VIEW kann wie eine normale Tabelle verwendet werden.

```
SELECT * FROM VPersonal;
```

UPDATES mit VIEWS

Allgemein können VIEWS lesend, sowie schreibend zum Aktualisieren von Tabellen verwendet werden. Möchte man Schreibzugriff auf Tabellen mit einer VIEW erreichen, müssen die nachstehende Bedingungen berücksichtigt werden.

Als Grundvoraussetzung gilt, dass es eine eindeutige Übereinstimmung zwischen der VIEW und der zu ändernden Tabellen gibt, andernfalls kann eine Anomalie erfolgen.

Beispiel 1:

```
UPDATE VPersonal
SET Vorname = 'FRANZ'
WHERE Vorname like 'Heinz';
```

Wenn nur ein Vorname mit Heinz vorliegt, ist alles gut. Aber was passiert, wenn es mehrere Benutzer mit dem Vornamen Heinz gibt?

In einer SELECTIONS-VIEW dürfen Daten nicht aus dem sichtbaren Bereich verschwinden.

Beispiel 2:

```
UPDATE VPersonal
SET BerufsBez = 'Chef'
WHERE BerufsBez like 'Meister';
```

Dieses Update wird scheitern, weil in der erstellten VIEW nur die Attribute Vorname, Nachname und Geburtsdatum vorhanden ist.

In einer PROJEKTS-VIEW müssen alle Felder vorhanden sein, auch die in der originalen Relation z.B. als NOT NULL definiert sind (es fehlt die ‚id').

Wenn Aggregationen oder Verbünde verwendet werden, können die Daten nicht immer sicher zugeordnet werden.

```
CREATE VIEW VLand AS (SELECT Vorname, Nachname, GebDatum AS Geburtsdatum,
adresse_plz
FROM personal p JOIN adresse a on a.plz = p.adresse_plz)
```

Bei einem Insert auf diese View wird eine Fehlermeldung erzeugt.

```
INSERT INTO VLand (Vorname, Nachname, Geburtsdatum, adresse_plz, land)
VALUES ('Josef', 'Gruber','28.01.2020',2323,'AUT');
```

Achtung: Es kann immer nur eine Tabelle aktualisiert werden.

VIEW – WITH CHECK OPTION

Eine VIEW kann man also genauso für das Einfügen von Daten in eine Tabelle verwenden. Grundlage des nächsten Beispiels ist die VIEW VGeburtsdatum. Hier wurde eine einfache VIEW erstellt, die einige Spalten der Personaltabelle beinhaltet.

```
CREATE VIEW VGeburtsdatum AS
SELECT Persnr, Vorname, Nachname, GebDatum AS Geburtsdatum ,adresse_plz
FROM personal
WHERE GebDatum IS NOT NULL;
```

Wenn mit dieser VIEW ein Datensatz eingefügt wird, wo das Geburtsdatum als Null vorhanden ist, gibt es keine Probleme. Allerdings kann die VIEW diesen Eintrag nicht mehr anzeigen (Geburtstag IS NULL)

```
INSERT INTO Geburtsdatum (PersNr, Vorname, Nachname, Geburtsdatum,
adresse_plz)
VALUES (12,'Josef', 'Markler', NULL, 5400);
```

Dieses Insert wird also erfolgreich eingetragen, der Eintrag von Josef Markler scheint bei einem SELECT auf die VIEW nicht auf. Also wurde mit einer VIEW ein Eintrag gemacht, der von der VIEW nicht angezeigt werden kann.

Damit das nicht passiert, gibt es die WITH CHECK OPTION, die den Eintrag in die Tabelle verhindert. Nachstehende VIEW verhindert die Ausführung.

```
ALTER VIEW VGeburtsdatum AS
SELECT Persnr, Vorname, Nachname, GebDatum AS Geburtsdatum, adresse_plz
FROM personal
WHERE GebDatum IS NOT NULL
WITH CHECK OPTION;
```

Achtung! Die Tabelle „personal“ kann aber nach wie vor mit NULL-Werten befüllt werden.

Warum soll eine VIEW updatebar sein

Da Datenbankbenutzer nur auf einen Teil der Datenbank (Tabellen) Zugriff haben sollen, kann mit einer VIEW der benötigte Datenbestand zur Verfügung gestellt werden. Weil aber Benutzer nicht nur Abfragen durchführen, sondern auch Daten in die Datenbank einspielen, kann mittels einer VIEW dieses Recht dem Benutzer gewährt werden. Der Benutzer muss in diesem Fall keine Zugriffsrechte auf die Basistabelle haben. Damit diese Zugriffe auch funktionieren, müssen die VIEWS

dementsprechend aufgebaut und mit den Kriterien aus dem Punkt UPDATES mit VIEWS übereinstimmen.