Views in SQL

Bernhard C. Wyss

Institut für Mobile und Verteilte Systeme
Hochschule für Technik
Fachhochschule Nordwestschweiz

Inhalt

Was sind Views?

- benannte Abfrage
- im Datenbanksystem hinterlegt
- Verwendung wie Basistabellen
- Datenmanipulation durch Views sind eingeschränkt

Wozu dienen Views?

- Zugriff auf Daten einschränken
- Vereinfachung komplexer Abfragen
- dient der Datenunabhängigkeit
- ermöglicht verschiedene Sichten auf dieselben Daten

Einfache und komplexe Views

Einfache Views

- beziehen Daten aus einer Tabelle
- enthalten keine Funktionen oder Gruppen
- Datenmanipulation ist möglich

Komplexe Views

- beziehen Daten aus mehreren Tabellen
- enthalten Funktionen oder Gruppen
- Datenmanipulation ist nur eingeschränkt möglich

Erzeugen von Views

Eine Subquery wird innerhalb eines **CREATE VIEW** Befehls eingebettet

Syntax

Meint auch im Fe

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW vname
    [(alias[, alias] ...)]
  AS subquery
 [WITH CHECK OPTION [CONSTRAINT cname]]
 [WITH READ ONLY [CONSTRAINT cname]];
```

Erzeugen von Views

View zeigt nur eine Auswahl von Spalten aus mitglieder

```
CREATE OR REPLACE VIEW mitglieder_info
  AS SELECT name, wohnort
     FROM mitglieder;
```

Die View enthält alle Spalten der SELECT Klausel mit den Datentypen aus der Tabellendefinition.

Erzeugen von Views

View zeigt nur eine Auswahl von Spalten aus mitglieder

```
CREATE OR REPLACE VIEW mitglieder_info
  AS SELECT substr(name,4) AS nachname, wohnort AS ort
     FROM mitglieder;
```

Mit Alias Namen in der SELECT Klausel werden die Spaltennamen der View festgelegt.

Erzeugen von Views

View zeigt nur eine Auswahl von Spalten aus mitglieder

```
CREATE OR REPLACE VIEW mitglied_info
  (nachname, ort)
  AS SELECT substr(name,4), wohnort
     FROM mitglieder;
```

Mit der eigenen Spaltenliste vor der Subquery werden die Spaltennamen der View festgelegt.
Dies ist die bevorzugte Variante!

Erzeugen von Views

View zeigt nur Mitglieder aus Basel

```
CREATE OR REPLACE VIEW basler_mitglieder
  (mnr, name, wohnort, gdatum) <-Ist optional
  AS SELECT mnr, name, wohnort, gdatum
     FROM mitglieder
     WHERE wohnort = 'Basel';
```

Abfragen mit Views

Daten aus Views können abgefragt werden, wie wir das mit Basistabellen gewohnt sind

Query mit View

```
SELECT *
FROM basler_mitglieder;
```

Es sind nur die Spalten und Zeilen sichtbar, welche durch die View definiert sind.

Manipulation mit Views

Daten in Views können geändert werden

```
UPDATE mitglied_info
SET ort = 'Luzern'
WHERE nachname='Buser';
```

Die View verhält sich wie eine Tabe

aber

Manipulation eingeschränkt

```
INSERT INTO mitglied_info
(nachname, ort)
VALUES ('Schwarz', 'Zürich');
```

Probelm: substr() funktion. Und d

Regeln für DELETE

Man kann keine Zeilen aus einer View entfernen, falls sie Folgendes enthält:

- Gruppenfunktionen
- GROUP BY Klausel
- das Schlüsselwort DISTINCT
- die Pseudospalte ROWNUM

Regeln für UPDATE

Man kann keine Zeilen in einer View ändern, falls sie Folgendes enthält:

- Gruppenfunktionen
- GROUP BY Klausel
- das Schlüsselwort DISTINCT
- die Pseudospalte ROWNUM
- durch Ausdrücke definierte Spalten also z.B eine funktion

Regeln für INSERT

Man kann keine Zeilen zu einer View hinzufügen, falls sie Folgendes enthält:

- Gruppenfunktionen
- GROUP BY Klausel
- das Schlüsselwort DISTINCT
- die Pseudospalte ROWNUM
- durch Ausdrücke definierte Spalten
- NOT NULL Spalten der Basistabelle, die nicht durch die View ausgewählt werden

WITH CHECK OPTION

Änderungen sind möglich, welche durch die Views nicht gesehen werden

```
UPDATE basler_mitglieder
SET wohnort = 'Bern'
WHERE mnr = 'M005';
```

Dieses Mitglied ist nun natürlich nicht mehrsic

Diese Änderung ist möglich, ist aber in der View nicht sichtbar.

WITH CHECK OPTION

Mit der WITH CHECK OPTION Änderungen einschränken

```
CREATE OR REPLACE VIEW basler_mitglieder
  (mnr, name, wohnort, gdatum)
  AS SELECT mnr, name, wohnort, gdatum
     FROM mitglieder
     WHERE wohnort = 'Basel'
  WITH CHECK OPTION;
```

Der vorherige UPDATE Befehl wird nun zurückgewiesen, da die Zeile in der View nicht sichtbar ist.

Views mit GROUP BY Klausel

Gesamte Ausleihdauer jeder DVDkopie

```
CREATE OR REPLACE VIEW ausleih_dauer
  (dvdnr, dauer)
  AS SELECT dvdnr, SUM(NVL(rueckgabe,SYSDATE)-datum)
     FROM ausleihen
     GROUP BY dvdnr;
```

Diese View ist nicht manipulierbar.

Joins in Views

Alle Angaben über Filialen und deren DVDkopien

```
CREATE OR REPLACE VIEW filialen_kopien
  (fnr, ort, plz, dvdnr, katalognr)
  AS SELECT fnr, ort, plz, dvdnr, katalognr
     FROM filialen JOIN dvdkopien USING (fnr);
```

Ist diese View manipulierbar?.

Wir können dort manipulieren wo wir

Regeln für Join Views

key-preserved table

Eine Tabelle ist *key-preserved*, wenn jeder Schlüssel der Tabelle auch ein Schlüssel im Resultat des Joins ist. Die Schlüssel dieser Tabelle bleiben also durch den Join erhalten.

Allgemeine Regel

Jede INSERT, UPDATE oder DELETE Operation auf einer Join View kann gleichzeitig nur eine Basistabelle ändern.

Dies gilt auch wenn eine (theoretische)

Regeln für Join Views

UPDATE Regel

Alle änderbaren Spalten einer Join View müssen auf die Spalten einer sog. key-preserved table abgebildet werden.

DELETE Regel

Zeilen einer Join View können gelöscht werden, solange es genau eine key-preserved table im Join gib.

INSERT Regel

Der INSERT Befehl darf sich nicht auf Spalten einer non-key-preserved table beziehen.

Änderbare Spalten in Join Views

updatable_columns

```
SELECT *
FROM user_updatable_columns
WHERE table_name = 'FILIALEN_KOPIEN';
```

Das Resultat zeigt an, welche Spalten potentiell änderbar sind.