JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

VIEWS IN POSTGRES/ MYSQL

Informationstechnische Systeme
4AHITM
2015/16

Matthias Mischek & Benedikt Berger

Version 0.1
Begonnen am 25.11.2015
Beendet am 9.12.2015

Note:

Inhaltsverzeichnis

1. Aufgabenstellung

Ziele

Das Ziel ist die Gegenüberstellung von Views in MySQL und Postgresql zu erarbeiten. Welche Unterschiede gibt es in den beiden DBMS in der Verwendung von Views? Am Beispiel einer bestehenden Datenbank soll dann auch die Anwendung von Views trainiert werden. Es soll dabei auch die Rechtevergabe getestet werden.

Voraussetzungen

Grundlegendes Verständnis von Views und deren Einsatz.
Create View: http://dev.mysql.com/doc/refman/5.7/en/create-view.html | http://www.postgresql.org/docs/current/static/sql-createview.html

Alter View: http://dev.mysql.com/doc/refman/5.7/en/alter-view.html | http://www.postgresql.org/docs/current/static/sql-alterview.html

Drop View: http://dev.mysql.com/doc/refman/5.7/en/drop-view.html | http://www.postgresql.org/docs/current/static/sql-dropview.html

Aufgaben zu Views auf die Schokofabrik

- Mitarbeiter, die die Maschinen mit der höchsten Produktionszahl haben.
- Liste der meist gelagerten Produkte.
- Liste der Kunden, die die meisten Produkte nach Jahren abgenommen haben.

Erstellen Sie auch einen Benutzer, der nur auf die generierten Views Zugriff hat und überprüfen Sie diese Rechte! Das Protokoll ist als Gruppenarbeit zu realisieren, wobei der Zugrfif auf die Datenbank auch über das Netzwerk zu überprüfen ist.

2. Was sind Views?

Eine View ist eine logische Relation (auch virtuelle Relation oder virtuelle Tabelle) in einem Datenbanksystem. Diese logische Relation wird über eine im Datenbankmanagementsystem (DBMS) gespeicherte Abfrage definiert. Der Datenbankbenutzer kann eine Sicht wie eine normale Tabelle abfragen. Wann immer eine Abfrage diese Sicht benutzt, wird diese zuvor durch das Datenbankmanagementsystem berechnet. Eine Sicht (View) stellt im Wesentlichen einen Alias für eine Abfrage dar.

3. Unterschied zwischen MySQL und PostgreSQL

In **PostgreSQL** handelt es sich bei Views tatsächlich um eine Alias des SELECT Statements, da es jedes Mal aufgerufen wird, wenn die View referenziert wird. Zudem sind in PostgreSQL sogenannte „Materialized Views“ vorhanden, welche tatsächlich eine neue Tabelle erstellen. [PSQL2], [PSQL1]

In **MySQL** haben Änderungen der ursprünglichen Tabellen keine Auswirkung auf den damit verbundenen View. [MSQL1]

3.1. Update View

In MySQL und PostgreSQL kann beim Erstellen einer View mit der Option WITH CHECK OPTION, gewährleistet werden, dass auch später Daten erscheinen welche über die View eingefügt oder geändert wurden. In PostgreSQL sind Views automatically updateable, falls diese einfach sind. Komplizierte hingegen können standardmäßig nur gelesen werden. Eine updatealbe View kann manuell erzeugt werden. In MySQL ist eine View updateable, wenn die Reihen in der Quelltablle und der View eine 1 zu 1 Beziehung hat.

3.2. Alter Statement

Mithilfe des **Alter** Statements können Eigenschaften (wie Name, Besitzer …) einer View verändert werden, allerdings nicht das SELECT Statement. Zusätzlich kann in MySQL mit dem Alter Statement auch das SELECT Statement verändert werden. [PSQL3],[MSQL2]

3.3. Drop Statement

Das Drop Statement löscht eine View oder falls notwendig können auch zusätzlich damit referenziert Views gelöscht werden, was Fehler vermeiden kann. [MSQL3], [PSQL4]

4. Praxis

4.1. Punkt 1

Mitarbeiter, die die Maschinen mit der höchsten Produktionszahl haben.

```
CREATE VIEW hoeste_produktionszahlen AS
SELECT maschnummer, beschreibung, count(*) AS "Anzahl Produkte"
FROM erzeugt
INNER JOIN maschine ON maschnummer=nummer
GROUP BY maschnummer, beschreibung
ORDER BY count(*) DESC
LIMIT 1;

SELECT person.nummer, vorname, nachname
FROM bedient
INNER JOIN mitarbeiter ON mitnummer = nummer
INNER JOIN person ON person.nummer = mitarbeiter.nummer
WHERE bedient.maschnummer = (SELECT maschnummer FROM
hoeste_produktionszahlen);
```

```
schokofabrik=> CREATE VIEW hoeste_produktionszahlen AS
schokofabrik-> SELECT maschnummer, beschreibung, count(*) AS "Anzahl Produkte"
schokofabrik-> FROM erzeugt
schokofabrik-> INNER JOIN maschine ON maschnummer=nummer
schokofabrik-> GROUP BY maschnummer, beschreibung
schokofabrik-> ORDER BY count(*) DESC
schokofabrik-> LIMIT 1;
CREATE VIEW
schokofabrik=>
schokofabrik=> SELECT person.nummer, vorname, nachname
schokofabrik-> FROM bedient
schokofabrik-> INNER JOIN mitarbeiter ON mitnummer = nummer
schokofabrik-> INNER JOIN person ON person.nummer = mitarbeiter.nummer
schokofabrik-> WHERE bedient.maschnummer = (SELECT maschnummer FROM hoeste_produktionszahlen);
 nummer |  vorname   | nachname
--------+------------+-----------
      4 | Justus     | Goller
      6 | Konstantin | Seger
     35 | Reini      | Mandelson
     37 | Karl       | Umser
(4 Zeilen)
```

Ausgegeben wird eine Tabelle mit den beiden benötigten Spalten „vorname“ und „nachname“ welche alle Mitarbeiter enthält, welche die Maschinen mit der höchsten Produktionszahl betreiben.

4.2. Punkt 2

Liste der meist gelagerten Produkte.

```
CREATE VIEW maxgelagert_produkte AS
SELECT pnummer, SUM(menge) AS "Anzahl"
FROM lagert
GROUP BY pnummer
ORDER BY pnummer ASC;

SELECT nummer, bezeichnung, "Anzahl"
FROM produkt
INNER JOIN maxgelagert_produkte ON nummer = pnummer
ORDER BY "Anzahl" DESC
LIMIT 4;
```

```
schokofabrik=> CREATE VIEW maxgelagert_produkte AS
schokofabrik-> SELECT pnummer, SUM(menge) AS "Anzahl"
schokofabrik-> FROM lagert
schokofabrik-> GROUP BY pnummer
schokofabrik-> ORDER BY pnummer ASC;
CREATE VIEW
schokofabrik=>
schokofabrik=> SELECT nummer, bezeichnung, "Anzahl"
schokofabrik-> FROM produkt
schokofabrik-> INNER JOIN maxgelagert_produkte ON nummer = pnummer
schokofabrik-> ORDER BY "Anzahl" DESC
schokofabrik-> LIMIT 4;
 nummer |          bezeichnung          | Anzahl
--------+-------------------------------+--------
      3 | Tafel Weisse Schokolade klein |  16000
      1 | Tafel Milchschokolade klein   |  16000
      5 | Tafel Dunkle Schokolade klein |  11500
      6 | Tafel Dunkle Schokolade gross |   9000
(4 Zeilen)
```

Es wird eine Tabelle mit den am meisten gelagerten Produkten ausgegeben, sowie zusätzlich die Anzahl der gelagerten Produkten.

4.1. Punkt 3

Liste der Kunden, die die meisten Produkte nach Jahren abgenommen haben.

```
CREATE VIEW kunde_maxprodukte AS SELECT q1.firmenname, q1.anzahl, q3.jahr
FROM (SELECT SUM(menge) AS anzahl,EXTRACT(YEAR FROM datum) AS
jahr,firmenname FROM auftrag NATURAL JOIN enthaelt GROUP BY
firmenname,jahr) AS q1

INNER JOIN (SELECT MAX(q2.anzahl) AS anzahl,jahr FROM
(SELECT SUM(menge) AS anzahl, EXTRACT(YEAR FROM datum) AS jahr,firmenname
FROM auftrag NATURAL JOIN enthaelt GROUP BY firmenname,jahr)
AS q2 GROUP BY jahr) AS q3 ON q1.anzahl=q3.anzahl;

SELECT * from kunde_maxprodukte;
```

```
schokofabrik=> CREATE VIEW kunde_maxprodukte AS SELECT q1.firmenname, q1.anzahl, q3.jahr FROM (SELECT SUM(menge) AS an
zahl,EXTRACT(YEAR FROM datum) AS jahr,firmenname FROM auftrag NATURAL JOIN enthaelt GROUP BY firmenname,jahr) AS q1
schokofabrik-> INNER JOIN (SELECT MAX(q2.anzahl) AS anzahl,jahr
schokofabrik(> FROM
schokofabrik(> (SELECT SUM(menge) AS anzahl, EXTRACT(YEAR FROM datum) AS jahr,firmenname
schokofabrik(> FROM auftrag NATURAL JOIN enthaelt GROUP BY firmenname,jahr)
schokofabrik(> AS q2 GROUP BY jahr) AS q3 ON q1.anzahl=q3.anzahl;
CREATE VIEW
schokofabrik=> SELECT * from kunde_maxprodukte;
 firmenname | anzahl | jahr
------------+--------+------
 Megamarkt  | 171505 | 2007
 Megamarkt  | 242700 | 2008
(2 Zeilen)
```

Die erzeugte Tabelle gibt die Firmen an, welche nach Jahren die meisten Produkte abgenommen haben und zusätzlich die Anzahl der Produkte.

4.1. Punkt 4

Erstellen Sie auch einen Benutzer, der nur auf die generierten Views Zugriff hat und überprüfen Sie diese Rechte!

```
CREATE USER benutzer1 WITH login password 'passwort1';
GRANT ALL PRIVILEGES ON hoeste_produktionszahlen TO benutzer1;
GRANT ALL PRIVILEGES ON maxgelagert_produkte TO benutzer1;
GRANT ALL PRIVILEGES ON kunde_maxprodukte TO benutzer1;
```

5. Zeitaufzeichnung

Arbeitsaufteilung	durchgeführt von	geschätzter Aufwand
Praxis	Benedikt Berger	3 Stunden
Theorie	Matthias Mischek	3 Stunden
	Gesamt	**6 Stunden**

Arbeitsaufteilung	durchgeführt von	tatsächlicher Aufwand
Praxis	Benedikt Berger	~ 4 Stunden
Theorie	Matthias Mischek	~ 4 Stunden
	Gesamt	**~ 8 Stunden**

6. Quellenangaben

[PSQL1] PostgreSQL - CREATE VIEW [Online]. Available at: http://www.postgresql.org/docs/current/static/sql-createview.html [abgerufen am 26.01.2016]

[PSQL2] PostgreSQL - MATERIALIZED VIEW [Online]. Available at: http://www.postgresql.org/docs/current/static/sql-creatematerializedview.html [abgerufen am 26.01.2016]

[PSQL3] PostgreSQL - ALTER VIEW [Online]. Available at: http://www.postgresql.org/docs/current/static/sql-alterview.html [abgerufen am 26.01.2016]

[PSQL4] PostgreSQL - DROP VIEW [Online]. Available at: http://www.postgresql.org/docs/current/static/sql-dropview.html [abgerufen am 26.01.2016]

[MSQL1] MySQL - CREATE VIEW [Online]. Available at: http://dev.mysql.com/doc/refman/5.7/en/create-view.html [abgerufen am 26.01.2016]

[MSQL2] MySQL - ALTER VIEW [Online]. Available at: http://dev.mysql.com/doc/refman/5.7/en/alter-view.html [abgerufen am 26.01.2016]

[MSQL3] MySQL - DROP VIEW [Online]. Available at: http://dev.mysql.com/doc/refman/5.7/en/drop-view.html [abgerufen am 26.01.2016]