Kapitel 8

Views und Triggers

Stefan Keller

Dank an Dr. Andreas Neumann

Warum Views?

- Abfragen vereinfachen (De-Normalisierung)
- Daten Datenbank-seitig prozessieren
- Datenschutz (Benutzerrechte)

«Real applications tend to use lots and lots (and lots and lots!) of views»
(Zitat Jennifer Widom, Stanford)

View Eigenschaften

- Verhalten (lesenderweise) ähnlich normale Tabelle
- Verhalten bei INSERT/UPDATE/DELETE anders
- Definition über normale SQL Query

```
CREATE VIEW myview AS
  SELECT *
  FROM mytab;
```

Komplexere Views sind Read-Only

- Views können komplex sein und für die Datenbank wäre es unklar welche Daten in welche Tabelle geschrieben werden müssten
- Manche Datenbanken erlauben manipulierbare (updatable) Views, aber nur bei Views die aus einfachen Abfragen bestehen → sehr einschränkend
- PostgreSQL Updatable Views mit folgenden Einschränkungen: http://www.postgresql.org/docs/current/static/sql-createview.html#SQL-CREATEVIEW-UPDATABLE-VIEWS

- Lösung:
  - Instead-Of-Trigger
  - müssen manuell erstellt werden

Beispiel View-Definition

```
CREATE OR REPLACE VIEW test.strassenachsen AS
 SELECT
  s.ogc_fid,
  s._tid,
  s.the_geom,
  l.adressfunktion,
  s.ordnung,
  replace(n.text,'_',' ')::character varying AS text,
  l.lokalisationnummer
 FROM av.gebaeudeadressen__strassenstueck s
 LEFT JOIN av.gebaeudeadressen__lokalisation l
  ON s.strassenstueck_von = l._tid
 LEFT JOIN av.gebaeudeadressen__lokalisationsname n
  ON l._tid = n.benannte;
```

Trigger und Trigger-Funktionen

- Trigger werden durch ein Event ausgelöst:
  - INSERT
  - UPDATE
  - DELETE

- Trigger-Funktionen sind der eigentliche Code (geschrieben in SQL, PL/pgSQL, Perl, Python, C, Java), der einen Record manipuliert oder nach einem Statement etwas ausführt

- Im Falle von mehreren ausgelösten Triggern werden diese in alphabetischer Reihenfolge ausgeführt

Trigger-Funktionen

- … sind:
  - ROW-Level oder STATEMENT-Level
    - Typischerweise ROW-Level
  - BEFORE oder AFTER
    - BEFORE-Trigger manipulieren eingehende Records und geben sie weiter
- Die Trigger-Ausführung
  - kann mit WHERE-Bedingungen eingeschränkt werden (bessere Performance!)
- INSTEAD OF-Trigger
  - Geben keine Records weiter:
    RETURN NULL statt RETURN NEW

Beispiel 1 Trigger-Funktion

```
create table test (id int primary key, name text);

create or replace function test_trigger_function ()
returns trigger as $$
begin
  if tg_op = 'INSERT' then
    raise notice 'Ein Datensatz wurde in test eingefügt.';
    return new;
  elsif tg_op = 'UPDATE' then
    raise notice 'Ein Datensatz in test wurde aktualisiert.';
    return new;
  elsif tg_op = 'DELETE' then
    raise notice 'Ein Datensatz in test wurde gelöscht.';
    return old;
  end if;
end;
$$ language plpgsql;

create or replace trigger test_trigger
after insert or update or delete on test
for each row execute function test_trigger_function();

truncate test; -- Würde eine Ergänzung der Funktion benötigen und ein Statement-Trigger
insert into test (id, name) values (1, 'Hallo'), (2, 'bbb'), (3, 'ccc');
update test set name = 'Hallo updated' where id = 1;
delete from test where id = 2;
select * from test order by 1;
```

Beispiel 2 Trigger-Funktion (1 / 3)

```
CREATE OR REPLACE FUNCTION test.test_trigger_function()
  RETURNS trigger AS
$BODY$
  BEGIN
    NEW.flaeche := ST_Area(NEW.the_geom);
    NEW.umfang := ST_Perimeter(NEW.the_geom);
    RETURN NEW;
  END;
$BODY$
LANGUAGE plpgsql VOLATILE COST 100;
```

Beispiel 2 Trigger-Funktion (2 / 3)

```
CREATE OR REPLACE FUNCTION test.test_trigger_function()
  RETURNS trigger AS
$BODY$
  DECLARE
    myrec RECORD;
  BEGIN
    NEW.flaeche := ST_Area(NEW.the_geom);
    NEW.umfang := ST_Perimeter(NEW.the_geom);
    SELECT array_to_string(array_agg(nz.zonenbez_gemeinde),'; ')
        AS zonen into myrec
        FROM raumplanung.nutzungszonen nz
        WHERE ST_Intersects(NEW.the_geom,nz.the_geom);
    NEW.zonen := myrec.zonen;
    RETURN NEW;
  END;
$BODY$
LANGUAGE plpgsql VOLATILE COST 100;
```

Beispiel 2 Trigger-Funktion (3 / 3)

```
CREATE TRIGGER trigger_update_mytab
  BEFORE INSERT OR UPDATE
  ON mytab
  FOR EACH ROW
  EXECUTE PROCEDURE test_trigger_function()
```