Universiteit Gent - TELIN - DDCM
2019-2020

Workshop: Datamanipulatie

1 Introductie

In de workshop 'Introductie PostgreSQL & Fysiek ontwerp' hebben we voornamelijk onderzocht hoe je met een relationele databank kan communiceren en de basiscomponenten van een relationele databank fysiek kan implementeren in PostgreSQL. In de workshop 'Importeren van data' hebben we het toevoegen van data in een databank bekeken. In deze workshop gaan we dieper in op twee operaties die vaak noodzakelijk zijn in een databank, namelijk het aanpassen en verwijderen van data. Net als in de vorige workshops zullen we aan de hand van de 'Rolls Robin' voorbeeldopgave kijken hoe we in PostgreSQL data aanpassen en verwijderen. Voor de volledigheid is het relationele databankschema van de 'Rolls Robin' databank terug te vinden in de Appendix. Ook vinden jullie op Ufora (in de map van de vorige workshops) SQL-scripts die alle statements bevatten voor de fysieke implementatie van deze databank en het toevoegen van data aan de databank.
Voor we beginnen willen we terug even vermelden dat we tijdens deze workshop SQL-statements zullen introduceren die vallen onder de DML-familie. De afkorting 'DML' staat voor 'Data Manipulation Language' omwille van het feit dat deze instructies data zullen manipuleren (toevoegen, aanpassen, verwijderen). Opnieuw kiezen we voor het open-source PostgreSQL databankmanagementsysteem.

2 Eerste stappen

Vooraleer we effectief bestaande data gaan aanpassen of verwijderen in de `rollsrobin` databank, komen we eerst nog even kort terug op een aantal zaken uit de vorige workshops.

Ten eerste willen we jullie er terug aan herinneren dat alle info over het downloaden en installeren van PostgreSQL en alle gerelateerde tools op jullie eigen PC is samengevat in een handleiding op Ufora. Daarnaast is er ook een handleiding te vinden omtrent het gebruik van PostgreSQL op de labo-PC's. Vooraleer jullie starten met deze workshop is het aangeraden om het document dat voor jullie van toepassing is grondig door te nemen.
Daarnaast wordt er verwacht dat jullie de vorige workshops 'Introductie PostgreSQL & Fysiek ontwerp' en 'Datamanipulatie' volledig hebben afgewerkt. Hieronder wordt er per voorgaande workshop beschreven wat noodzakelijk is om de workshop 'Datamanipulatie' tot een goed einde te kunnen brengen.

2.1 Workshop 'Introductie PostgreSQL & Fysiek ontwerp'

Idealiter hadden jullie op het einde van de workshop 'Introductie PostgreSQL & Fysiek ontwerp' een werkende `rollsrobin` databank en/of een backup in de vorm van een SQL-script van deze databank. Mocht dit niet het geval zijn, kunnen jullie een volledig backup-script terugvinden in de overeenkomstige map op Ufora. Dit script kan je gebruiken als voorbeeldoplossing van het fysieke ontwerp voor de `rollsrobin` databank. Indien jullie dit script wensen te gebruiken is het aangeraden om de oplossing in detail door te nemen vooraleer verder te gaan, zodat je zeker bent dat je alle concepten uit deze workshop goed begrijpt.
Een fysieke implementatie van de `rollsrobin` databank bekomen jullie door deze backup in te laden (restore). Dit kan je doen via de commandolijn-applicatie `psql` met volgend commando:

```
psql
    --dbname rollsrobin
    --username postgres
    --file rollsrobin_physical.sql
```

Let wel op dat er reeds een lege `rollsrobin` databank moet bestaan op je lokale PostgreSQL-cluster. Deze dien je dus eerst zelf aan te maken.

2.2 Workshop 'Importeren van data'

Op het einde van de workshop 'Importeren van data' hebben jullie normaal gezien een `rollsrobin` databank waar alle data uit `rollsrobindata.csv` gestructureerd inzitten. Indien je werkte op een labo-PC, diende je een back-up te nemen van de volledige databank - inclusief de data zelf. Deze back-up kan je dan op identieke manier als hierboven beschreven opnieuw restoren.
Indien jullie deze workshop nog niet geheel hebben afgewerkt, kunnen jullie altijd de twee voorbeeldoplossingen (in de vorm van een SQL-script) gebruiken, die te vinden zijn in de overeenkomstige map op Ufora. Het eerste script, `super_rollsrobin.sql`,

definieert een super-relatie die bestaat uit alle attributen die terug te vinden zijn in `rollsrobindata.csv`. Dit script kan je inladen in een query tool gekoppeld aan de databank `rollsrobin` van pgAdmin 4 en uitvoeren. Daarna kunnen jullie de data uit dit .csv-bestand importeren in deze super-relatie via de 'Import/Export...' functie van pgAdmin 4. Tot slot kunnen jullie het script `rollsrobin_dataimport.sql` uitvoeren (opnieuw via de query tool) om data uit de super-relatie te kopiëren naar de tabellen aangemaakt in de allereerste workshop. Bekijk beide scripts in detail zodat je zeker alles begrijpt uit de vorige workshop.

Zorg vooraleer verder te gaan dat je een volledig geïmplementeerde en gevulde `rollsrobin` databank hebt aangemaakt op je lokale PostgreSQL-cluster. Check of alle tabellen het gewenste aantal rijen bevat (zie workshop 'Importeren van data').

3 Aanpassen en verwijderen

Eens data aan de database zijn toegevoegd, is het mogelijk om te gaan werken met deze data. Ten eerste zullen jullie tijdens de SQL-lessen zien hoe het mogelijk is om specifieke antwoorden op vragen over deze data te geven of diepgaande analyses te doen met behulp van `SELECT`-instructies. Daarnaast is het ook mogelijk om data die in de databank zitten aan te passen (update) of te verwijderen (delete). Dit bespreken we hieronder.

3.1 Data aanpassen

Het aanpassen van data is zeer eenvoudig. Je hoeft enkel de data te selecteren die je wil aanpassen en de aanpassingen te definiëren. Dit kan met volgende instructie[1].

```
UPDATE relatie SET attribuutnaam = waarde WHERE conditie;
```

Met deze instructie is het mogelijk om een nieuwe waarde te geven aan het attribuut met de opgegeven naam in de gegeven relatie voor alle tuples die voldoen aan de conditie.
Het aanpassen van de achternaam van alle personen (in de relatie `persoon`) met voornaam 'Toni' naar 'Bronselaer' kan door middel van volgende instructie.

```
UPDATE persoon SET achternaam = 'Bronselaer'
    WHERE voornaam = 'Toni';
```

[1] `https://www.postgresql.org/docs/current/sql-update.html`

Voer bovenstaande `UPDATE`-instructie uit. Check of de aanpassing is doorgevoerd door de data in de tabel `persoon` te bekijken.

3.2 Data verwijderen

Ook het verwijderen van data kan met een eenvoudige instructie. Opnieuw moet je enkel de relatie en de data die je wilt verwijderen definiëren. Dit kan met volgende instructie[2].

```
DELETE FROM relatie WHERE conditie;
```

Met deze instructie verwijder je de data die voldoen aan de conditie uit de gegeven relatie.
Als voorbeeld het statement waarbij we personen verwijderen met voornaam 'Toni'.

```
DELETE FROM persoon WHERE voornaam = 'Toni';
```

Voer bovenstaande `DELETE`-instructie uit. Lukt dit? Waarom wel/niet?

3.3 Beperkingen

Wanneer jullie bovenstaande `DELETE`-instructie proberen uit te voeren, zal het systeem een foutmelding genereren. Deze foutmelding meldt dat het niet mogelijk is om de persoon/personen met voornaam 'Toni' uit de tabel `persoon` te verwijderen omdat er in de tabel `registratieformulier` nog een referentie zit naar deze persoon/personen. Dit is een probleem bij zowel het aanpassen als het verwijderen van data waar nog naar gerefereerd wordt door middel van een vreemde sleutelbeperking. Er bestaan verschillende strategieën om met zo een `DELETE` om te gaan. De meest gebruikte strategieën zijn

- `NO ACTION`: Een rij mag niet verwijderd (aangepast) worden zolang er nog rijen naar verwijzen - indien men dit toch probeert, genereert de database een foutmelding. Dit is de default optie.
- `CASCADE`: Als een rij verwijderd (aangepast) wordt, worden alle rijen die ernaar verwijzen ook verwijderd (aangepast).

[2] `https://www.postgresql.org/docs/current/sql-delete.html`

- `SET NULL`: Als een rij verwijderd (aangepast) wordt, worden de waarden van de foreign keys van alle rijen die ernaar verwijzen op `NULL` gezet. In dit geval zou je dus bijvoorbeeld pas een persoon kunnen verwijderen wanneer je eerst alle referenties naar deze persoon hebt verwijderd (namelijk alle contracten, werknemers en registratieformulieren die naar deze persoon verwijzen).

Voor het kiezen van één van de strategieën keren we even terug naar het fysieke databankontwerp. Als je een strategie wilt kiezen kan je dit aangeven bij het aanmaken van de vreemde sleutel met `ON UPDATE` (als de rij waarnaar verwezen wordt aangepast wordt) en `ON DELETE` (als de rij waarnaar verwezen wordt verwijderd wordt) gevolgd door de gewenste strategie.
We hernemen even het `CREATE TABLE`-statement voor de aanmaak van de `constructeur`-relatie.

```
CREATE TABLE constructeur
(
    merk varchar,
    model varchar,
    type varchar,

    CONSTRAINT constructeur_pkey PRIMARY KEY (merk, model),

    CONSTRAINT constructeur_type_fkey FOREIGN KEY (type)
        REFERENCES type (naam)
);
```

Daarnaast wensen we dat wanneer de naam van een wagentype wordt aangepast alle rijen die ernaar verwijzen in de `constructeur`-relatie ook worden aangepast en dat indien een wagentype wordt verwijderd, de waarde onder het `type`-attribuut in de `constructeur`-relatie op `NULL` wordt gezet. De bovenstaande instructie moet in dit geval aangepast worden naar

```
CREATE TABLE constructeur
(
    merk varchar,
    model varchar,
    type varchar,

    CONSTRAINT constructeur_pkey PRIMARY KEY (merk, model),

    CONSTRAINT constructeur_type_fkey FOREIGN KEY (type)
        REFERENCES type (naam)
        ON UPDATE CASCADE ON DELETE SET NULL
);
```

Indien je een nieuwe strategie wil toevoegen aan een al bestaande vreemde sleutel-beperking, ben je helaas genoodzaakt om deze eerst te verwijderen, en daarna terug te definiëren mét de gewenste strategie. Voor de `constructeur_type_fkey` vreemde sleutel voer je volgende commando's uit.

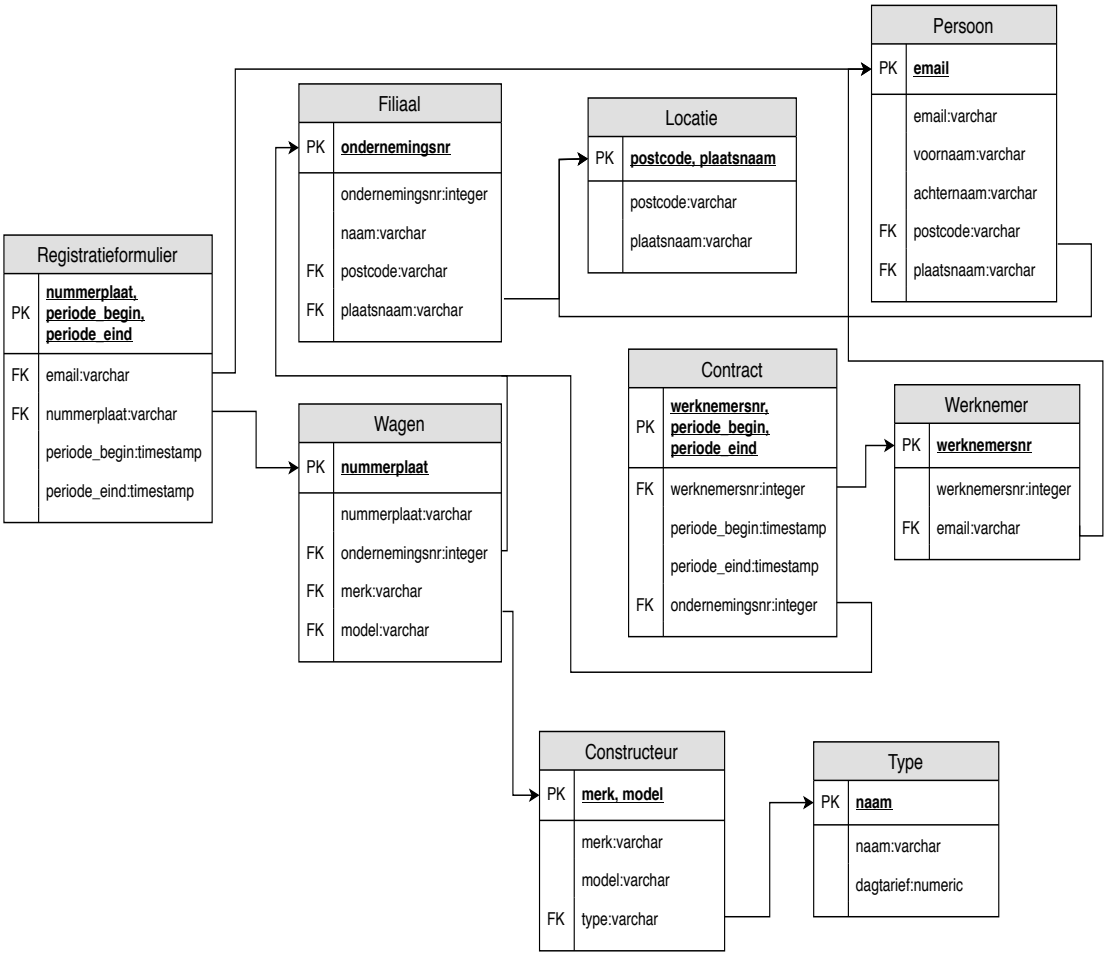
```
ALTER TABLE constructeur
    DROP CONSTRAINT constructeur_type_fkey;

ALTER TABLE constructeur
    ADD CONSTRAINT constructeur_type_fkey FOREIGN KEY (type)
    REFERENCES type (naam)
    ON UPDATE CASCADE ON DELETE SET NULL;
```

Pas alle vreemde sleutel-beperkingen aan zodat aanpassingen aan waarden onder een gerefereerd attribuut worden doorgevoerd in alle refererende relaties en het verwijderen van data resulteert in het verwijderen van de refererende rijen. Denk dus goed na welke strategie je gaat toepassen! Test of dit effectief werkt door de eerder gevraagde `DELETE`-instructie uit te voeren.

Appendix: Logisch databankontwerp Rolls Robin

In onderstaande figuur vind je een schematische weergave van een mogelijk logisch ontwerp voor de Rolls Robin databank. Hierbij wordt iedere basisrelatie weergegeven door een rechthoek, die bovendien een oplijsting van alle attributen met bijhorend datatype bevat. Daarnaast worden de attributen die behoren tot de primaire sleutel bovenaan weergegeven, en worden vreemde sleutels voorgesteld door een pijl tussen de betreffende attribuutverzamelingen. Alle extra beperkingen die niet kunnen weergegeven worden in dit schema worden onderaan opgelijst.



Extra beperkingen

- Type:
  1. CHECK: dagtarief ≥ 0
- Registratieformulier:

1. CHECK: periode_begin ≤ periode_eind
2. Insert: controleer bij toevoegen van registratieformulier dat periode niet overlapt met periode van ander registratieformulier voor dezelfde nummerplaat.

- Contract:
  1. CHECK: periode_begin ≤ periode_eind
  2. Insert: controleer bij toevoegen van nieuw contract dat periode van contract niet overlapt met periode van ander contract voor hetzelfde werknemersnummer.