Universiteit Gent - TELIN - DDCM
2023-2024

Workshop: Introductie PostgreSQL

1 Introductie

Vooraleer je goed en wel van start kan gaan met het implementeren en gebruiken van een databank, heb je wat algemene basiskennis nodig rond de installatie van en communicatie met databanksystemen. De bedoeling van deze workshop is dan ook om jullie een eerste keer kennis te laten maken met een databankbeheersysteem (dbms).
Belangrijk om op te merken is dat we, in deze workshop (en ook de volgende workshops), regelmatig gebruik zullen maken van de commandolijn. Als je geen (of weinig) ervaring hebt met de commandolijn, kan je best, vooraleer je aan deze workshop begint, het document dat de commandolijn kort introduceert eens doornemen. Dit document kan je terugvinden in de map van deze workshop op Ufora.
In dit document wordt er eerst en vooral dieper ingegaan op het databankbeheersysteem dat we doorheen dit semester continu gaan gebruiken, namelijk PostgreSQL (zie Sectie 2). We leggen uit wat PostgreSQL precies is en hoe je dit systeem kan installeren. Vervolgens leren we in Sectie 3 twee manieren aan om met een PostgreSQL databank te connecteren en te communiceren: via een commandolijnapplicatie (`psql`) en via een webapplicatie (pgAdmin 4). Tot slot tonen we je, in Sectie 4, hoe je een backup kan maken van een PostgreSQL databank en hoe je deze backup terug in kan laden (restore).

2 PostgreSQL

In dit vak zullen we voornamelijk werken met het relationeel databankmodel. Aangezien dit model enorm populair is, bestaan er zeer veel verschillende commerciële

implementaties van relationele databankbeheersystemen. Dergelijke programma's voorzien alle concepten van het relationele model (zoals tabellen, primaire sleutels, vreemde sleutels. . .) en laten dus toe om fysieke databanken volledig te implementeren. Wij hebben gekozen voor een van de meest gekende, en bovendien een open-source en gratis te downloaden relationeel databankbeheersysteem: PostgreSQL. Hoewel dit slechts 1 van de vele relationele databankbeheersystemen is, kan je veel van de aangeleerde concepten ook rechtstreeks toepassen op andere courante systemen zoals MySQL, Oracle, SQL Server. . .
Om een relationele databank aan te maken, moeten we eerst en vooral het PostgreSQL databankbeheersysteem downloaden en installeren. Hoe je dit doet, staat uitgebreid uitgelegd in een aparte installatiehandleiding die te vinden is in de map van deze workshop op Ufora. Volg nauwgezet de stappen die in deze handleiding beschreven staan om PostgreSQL, versie 16 (en het programma pgAdmin 4, waarmee je eenvoudig kan communiceren met PostgreSQL databanken) te installeren, vooraleer verder te gaan met deze workshop.

Installeer PostgreSQL, versie 16 en pgAdmin 4 op je eigen computer door de stappen in de installatiehandleiding te volgen.

3 Communiceren met een databank

Wanneer je de handleiding hebt doorlopen, zou PostgreSQL op een correcte manier geïnstalleerd moeten zijn en zou er bovendien een lokale PostgreSQL cluster moeten draaien op jouw PC. Bij het opstarten van PostgreSQL opent er geen grafisch gebruikersscherm waarin je allerlei dingen kan doen, zoals je misschien gewoon bent van een programma. PostgreSQL is namelijk een programma dat in de achtergrond draait op jouw PC. Bij de installatie van PostgreSQL wordt er standaard 1 databank, met naam `postgres`, aangemaakt. Natuurlijk willen we ook andere databanken (bv. `voetbal`) kunnen aanmaken en vervolgens kunnen gebruiken. Daartoe moeten we kunnen verbinden met de PostgreSQL cluster waarin we de databank willen aanmaken. In dit geval is dat de lokale cluster die op jouw eigen PC draait, maar het is belangrijk om te weten dat je vanop jouw PC ook kan verbinden met PostgreSQL clusters die op andere computers (of servers) draaien, als je hier toegang tot hebt.
Eens je bent verbonden met een cluster, kan je hier eenvoudig mee communiceren door instructies op te stellen die het dbms moet uitvoeren. Een van de grote voordelen van het relationele model is dat er een gestandaardiseerde taal bestaat om met relationele databankbeheersystemen te communiceren. Deze taal draagt de naam *Structured Query Language* of kortweg SQL. De diverse commerciële implementaties gebruiken elk een eigen SQL-dialect, maar de communicatieconcepten

die je doorheen de komende workshops zal leren en specifiek zijn voor PostgreSQL zijn dus ook grotendeels bruikbaar in het geval van andere implementaties. Er bestaan heel wat verschillende manieren om SQL-instructies door te geven aan PostgreSQL. In deze sectie zullen we twee manieren aan jullie voorstellen, zijnde `psql` (een commandolijn-applicatie) en pgAdmin, versie 4 (een webapplicatie met grafische interface). We zullen tonen hoe we een databank kunnen aanmaken en nadien opnieuw kunnen verwijderen door middel van deze twee communicatietools.

3.1 Commandolijn

Een eerste manier om met een PostgreSQL databank te communiceren is door te werken met het `psql`-programma, dat samen met PostgreSQL wordt geïnstalleerd. Je kan met dit programma werken in de commandolijn. Zoals je ondertussen wel al weet, kan je in de commandolijn allerlei programma's uitvoeren die op jouw PC geïnstalleerd staan, waaronder dus ook `psql`. Voer om te beginnen het volgende commando uit in de commandolijn[1].

```
psql --host=127.0.0.1 --port=5432 --username=postgres --dbname=postgres
```

Om correct te kunnen connecteren met een PostgreSQL cluster dien je steeds een aantal vereiste connectieparameters op te geven. Dit zijn

- een IP-adres/hostname: de host waarop de PostgreSQL cluster draait (de lokale PostgreSQL cluster draait op IP-adres 127.0.0.1, alias localhost),
- een poort: de poort van de host waarop je de PostgreSQL cluster kan bereiken,
- een gebruikersnaam: de naam van de gebruiker waarmee je jezelf wil aanmelden op de PostgreSQL cluster,
- een wachtwoord: het wachtwoord van de gebruiker waarmee je jezelf wil aanmelden op de PostgreSQL cluster.

Door het bovenstaande commando uit te voeren, verbind je dus via het `psql`-programma met de lokaal draaiende PostgreSQL cluster (op IP-adres 127.0.0.1, alias localhost, aangegeven met de `host`-optie, poort 5432, aangegeven met de `port`-optie). De daaropvolgende `username`-optie zorgt ervoor dat je jezelf aanmeldt als de gebruiker met gebruikersnaam 'postgres' (de standaard gebruiker op PostgreSQL). Tot slot kan je ook nog de naam van de databank waarmee je wenst te connecteren meegegeven na de `dbname`-optie. In het bovenstaande voorbeeld is dit de databank die

[1] Indien je een melding krijgt dat dit programma niet gevonden wordt, controleer dan of je PATH-variabele correct is ingesteld. Als dit niet blijkt te werken, kan je nog steeds in de commandolijn naar de locatie navigeren waar `psql` is geïnstalleerd en `psql` daar uitvoeren. Op Windows is dit typisch de map `C:\Program Files\PostgreSQL\16\bin`. Typ, daarnaast, het gegeven commando over en kopieer dit niet rechtstreeks vanuit dit document.

standaard wordt aangemaakt wanneer je PostgreSQL installeert en die als naam `postgres` heeft. Tot slot wordt er, bij uitvoering van dit commando, gevraagd om een wachtwoord. Dit is het wachtwoord van de gebruiker met naam 'postgres' dat je hebt ingesteld bij de installatie.
Eens je succesvol geconnecteerd bent, kan je door uitvoering van het commando `\l` (waarbij de 'l' voor 'list' staat) een overzicht bekomen van alle databanken die bestaan op jouw lokale PostgreSQL cluster. We willen nu echter een nieuwe databank aanmaken, met naam `voetbal`. Dit doen we door dit als een SQL-instructie door te geven aan het databankbeheersysteem. De PostgreSQL-instructie om een databank aan te maken is de volgende[2].

```
CREATE DATABASE voetbal;
```

Voer deze instructie uit en controleer opnieuw welke databanken er bestaan op je lokale PostgreSQL cluster. Als alles goed is gegaan, zou de `voetbal` databank nu in deze lijst moeten voorkomen. Nu de databank is aangemaakt, zouden we kunnen beginnen met het implementeren van verschillende componenten in deze databank. Hiervoor moet je eerst connectie maken met de `voetbal` databank die je juist hebt aangemaakt. Dit kan eenvoudigweg door middel van volgende instructie.

```
\c voetbal
```

waarbij 'c' staat voor 'connect'. We tonen echter eerst hoe je een databank kan verwijderen, en zullen vervolgens met pgAdmin 4 de databank opnieuw aanmaken. Een databank kan met de volgende SQL-instructie worden verwijderd.

```
DROP DATABASE voetbal;
```

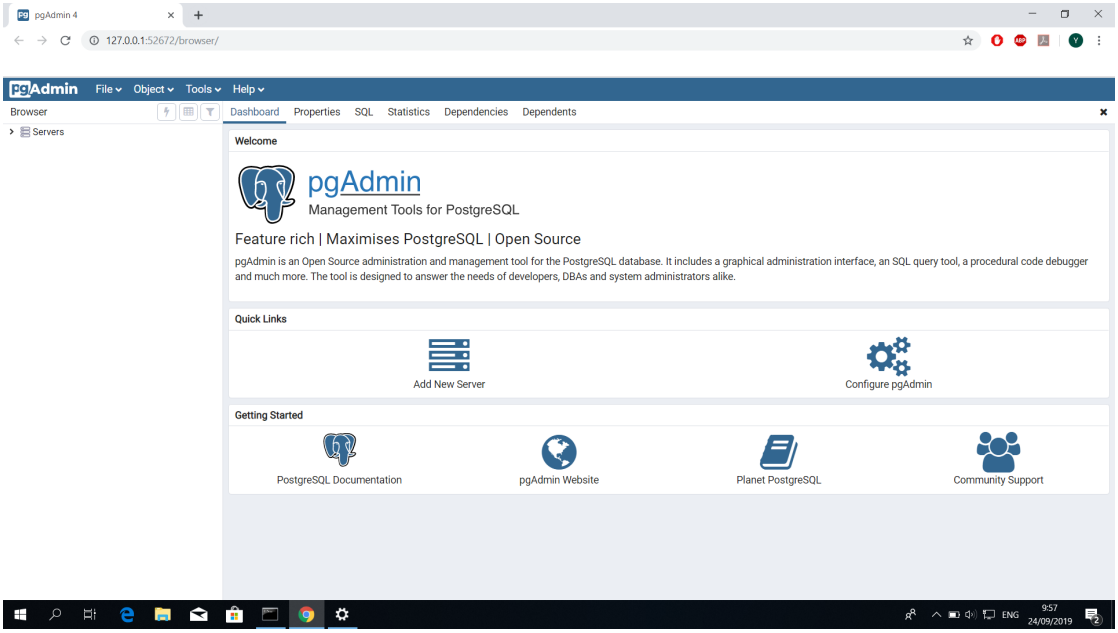
Voer dit commando uit vooraleer verder te gaan met de volgende opdrachten. Let wel op, je kan geen databank verwijderen indien er nog connecties bestaan naar deze databank. Connecteer dus eerst terug met de `postgres` databank en verwijder alle openstaande connecties naar de `voetbal` databank. Voor de volledige documentatie van `psql` kan je steeds terecht op `https://www.postgresql.org/docs/current/app-psql.html`.

Voer bovenstaande opdrachten uit en experimenteer wat met `psql` vooraleer verder te gaan.

[2] Sluit steeds al je PostgreSQL-specifieke commando's af met een puntkomma.

3.2 pgAdmin 4

Zoals we eerder al vermeld hebben is pgAdmin 4 een van de tools die gebruikt kan worden om te communiceren met een PostgreSQL databank. Net zoals in het geval van `psql` kan je door middel van pgAdmin 4 verbinden en communiceren met PostgreSQL databanken die op jouw lokale machine bestaan en ook met PostgreSQL databanken die op een andere machine te vinden zijn. Bovendien heeft de tool, in tegenstelling tot `psql`, een grafische interface waardoor je heel intuïtief met de gewenste databank kan werken. Daarnaast, echter, biedt pgAdmin 4, net als `psql`, ook een interface aan om rechtstreeks SQL-instructies (bv. `CREATE DATABASE`) naar een PostgreSQL cluster te sturen. In pgAdmin 4 bestaan er met andere woorden twee manieren om met een PostgreSQL databank te interageren. We tonen beide manieren hieronder.
Na het opstarten van pgAdmin 4 opent er een browservenster waarin de applicatie je begroet met een beginscherm dat eruit ziet zoals het scherm dat wordt weergegeven in Figuur 1. Dit scherm is opgedeeld in 2 delen: een browservenster links en een detailgedeelte rechts. Het detailgedeelte toont steeds informatie over het deel van de PostgreSQL cluster dat links geselecteerd is. Ook kan je in dit detailgedeelte acties uitvoeren die inwerken op het geselecteerde deel van de PostgreSQL cluster.



Figuur 1: pgAdmin 4 beginscherm.

Net zoals in het geval van `psql` zullen we eerst een connectie moeten maken met de PostgreSQL cluster die lokaal draait op jouw PC (hostname localhost, poort 5432), vooraleer we een nieuwe databank met naam `voetbal` kunnen aanmaken. In het

linkervenster wordt er een overzicht weergegeven van alle bestaande connecties. Een connectie maken met jouw lokale PostgreSQL cluster kan je doen op de manier die beschreven staat in onderstaande opdracht.

Maak een connectie met de 'localhost' cluster door in het browservenster met de rechtermuisknop te klikken op Servers - Register - Server... Geef de aangemaakte connectie de naam 'local'. Standaard draait een lokale PostgreSQL cluster op IP-adres 127.0.0.1 (alias localhost), poort 5432 en is de standaard gebruiker met gebruikersnaam 'postgres' aangemaakt zonder wachtwoord, tenzij je dit bij de installatie hebt ingesteld. Deze connectieparameters kan je invoeren onder het 'Connection' tabblad.

Je kan de verbinding 'local' nu openklikken. Onder 'Databases' vind je alle databanken die op deze connectie (host + poort) bestaan. Aangezien we daarnet in `psql` de aangemaakte `voetbal` databank opnieuw verwijderd hebben, zie je deze databank niet meer in het overzicht staan.

Maak de `voetbal` databank aan via de grafische interface van pgAdmin 4 door rechts te klikken op Databases - Create - Database... en vervolgens de naam van de databank in te vullen.

Normaal gezien staat de databank nu zichtbaar tussen de andere databanken die op jouw lokale PostgreSQL cluster bestaan[3]. We zullen, echter, de databank ook eens aanmaken door rechtstreeks SQL-instructies door te geven in pgAdmin 4.

Verwijder eerst de databank `voetbal`, door er rechts op te klikken en Delete/Drop te selecteren.

In pgAdmin 4 kan je dus ook rechtstreeks SQL-instructies doorgeven aan het dbms, net zoals we dit eerder deden in `psql`. Dit kan via de 'query tool' die je opent door rechts te klikken op de naam van een (willekeurige) databank in het browservenster, en dan 'Query Tool' te selecteren.

[3] Mogelijks moet je de pagina eens verversen vooraleer nieuw aangemaakte componenten worden weergegeven in het browservenster. Dit kan je doen door met de muis rechts op een component in het browservenster te klikken en 'Refresh...' te selecteren.

Maak de `voetbal` databank aan door middel van de `CREATE DATABASE` SQL-instructie en klik vervolgens op het uitvoeringssymbool bovenaan in het venster of druk op F5 om de instructie uit te voeren.

We hebben nu op drie verschillende manieren de `voetbal` databank aangemaakt. In het vervolg zullen we gebruik maken van pgAdmin 4 vanwege de grafische ondersteuning, maar we zullen wel met SQL-instructies blijven werken. Het is immers van cruciaal belang om SQL goed onder de knie te krijgen, zodat je niet afhankelijk bent van de grafische interface van pgAdmin 4 om met een (PostgreSQL) databankbeheersysteem te communiceren. Dit zorgt er dan ook voor dat je met verschillende tools een PostgreSQL databank kan beheren.

4 Backup en restore

De databanken die je hebt aangemaakt en de data die je in deze databanken kan laden worden opgeslagen in het permanente geheugen (de harde schijf) van jouw PC. Dit zorgt ervoor dat de betrouwbaarheid van de opslag rechtstreeks afhankelijk is van de hardware. Als je niet de juiste voorzorgsmaatregelen neemt, bestaat de kans dat op een dag jouw harde schijf crasht en je alle gegevens kwijt bent. Om dergelijke scenario's te vermijden en om de inwisselbaarheid van databanken over verschillende machines te vergroten, voorziet PostgreSQL een backup-mechanisme. Dit mechanisme vertaalt een databank naar een lange reeks SQL-instructies die nodig en voldoende zijn om een databank later volledig te kunnen reconstrueren. Dit kan niet alleen handig zijn uit veiligheidsoverwegingen, maar het kan ook gebruikt worden om een databank van één machine naar een andere machine te kopiëren, zonder dat je daarvoor elke stap uit het fysiek ontwerp op de nieuwe machine moet herhalen en je vervolgens alle data in de nieuwe databank opnieuw moet inladen. In het vervolg zullen we tonen hoe je een backup maakt van de `voetbal` databank en hoe je deze backup ook weer kunt inladen via de commandolijn.

4.1 Backup

Met de commandolijn-applicatie[4] `pg_dump` kan je een backup (een dump) nemen van een databank. Dit commando start, net als `psql`, een nieuw programma op dat uitgevoerd kan worden in de commandolijn. Let dus goed op dat je dit commando *niet* uitvoert binnen de `psql` omgeving. `pg_dump` schrijft alle instructies die nodig zijn om de PostgreSQL databank waarvan je een backup wenst te nemen weg naar een

[4] `https://www.postgresql.org/docs/current/app-pgdump.html`

.sql-bestand. Het backup-bestand bevat dus niets anders dan een sequentie SQL-instructies die men zou moeten uitvoeren om, vertrekkende van een lege toestand, te eindigen met een databank zoals die is op het moment van de backup. Hieronder wordt een voorbeeld gegeven van hoe je het `pg_dump` commando kan oproepen.

```
pg_dump
    --clean
    --create
    --if-exists
    --host=127.0.0.1
    --port=5432
    --dbname=voetbal
    --schema-only
    --username=postgres
    --file=voetbal.sql
```

`pg_dump` heeft heel wat verschillende opties die je kan meegeven wanneer je het programma wil uitvoeren. Aan de optie `dbname` geef je de naam van de databank mee waarvan je een backup wil maken. Deze databank dient te draaien op de server met het IP-adres (of hostname) meegegeven aan de `host` optie en op de poort meegegeven aan de `port` optie. De opties `clean` en `if-exists` zorgen er voor dat de `DROP DATABASE IF EXISTS` instructie wordt toegevoegd aan het .sql-backupbestand. In dit geval wordt, indien er reeds een databank met de opgegeven naam bestaat, deze databank eerst verwijderd bij het inladen van het backup-bestand. De optie `create` zorgt ervoor dat in de backup ook de `CREATE DATABASE` instructie wordt opgenomen. Met deze instructie wordt een nieuwe, lege databank met de gespecificeerde naam aangemaakt. Met de `schema-only` optie geef je aan dat je enkel de definitie van de databank wil backuppen, en niet de data zelf. Met de optie `username` geef je aan via welke gebruiker je een backup wil maken. Na de `file` optie geef je de padnaam op van het backup-bestand. Let op dat je het volledige commando op 1 lijn typt met spaties om de verschillende opties te scheiden (dus niet zoals wordt weergegeven in bovenstaand voorbeeld, met ‘newlines’ of ‘enters’ tussen de verschillende opties).

Maak een *schema*-backup van de `voetbal` databank waarin de databank *niet* opnieuw wordt aangemaakt. Je moet dus zelf nadenken welke opties er meegegeven moeten worden aan het `pg_dump` commando.

4.2 Restore

Via de commandolijn-applicatie `psql` is het mogelijk om een backup van een databank terug in te laden (restore). Dit kan je doen aan de hand van het volgende commando.

```
psql
    --host=127.0.0.1
    --port=5432
    --dbname=voetbal_backup
    --username=postgres
    --file=voetbal.sql
```

Dit commando is gelijkaardig aan de `pg_dump` en `psql` commando's die reeds uitgebreid toegelicht zijn. We gaan hier dus niet verder in op de betekenis van de verschillende opties. Let op, de `dbname` optie kan je enkel gebruiken indien je backupscript geen databankaanmaak-instructies bevat en je reeds een (lege) databank met corresponderende naam hebt aangemaakt op de gegeven PostgreSQL server.

Maak via `psql` een tweede databank aan met naam `voetbal_backup`. Verlaat vervolgens het programma `psql` (via Ctrl + C) en restore de databank naar `voetbal_backup` waarvan je een backup hebt gemaakt in Sectie 4.1

.