HET ETL PROCES

INLEIDING

In dit tweede deel van het project vertrekken we van het sterschema dat je in deel 1 hebt ontworpen en geïmplementeerd. Het doel van dit deel is een ETL-proces te implementeren dat je in staat stelt om gegevens in de snapshots stelselmatig te importeren in het warehouse.

Het ETL-proces

Het ETL-proces dat we in dit deel ontwerpen en implementeren, moet zich voornamelijk richten op het correct transfereren van gegevens van de brondatabase naar het warehouse. Op dit moment hoef je nog géén rekening te houden met mogelijk problemen omtrent datakwaliteit. Wel is het nodig dat je in het ETL-proces rekening houdt met de volgende aspecten:

- Voorzie de nodige stappen om afgeleide data te berekenen, dit zowel voor dimensietabellen als de feitentabel.
- Hou rekening met een duidelijke naamgeving van zowel tabellen als attributen.
- In de dimensietabellen waarin je versiebeheer voorziet, is het nodig om versies correct te gaan aanmaken en/of aanpassen waar nodig. Bij het inladen van data in een dimensietabel heb je namelijk twee mogelijkheden met betrekking tot versiebeheer. Als je gegevens reeds aanwezig zijn en er zijn geen wijzigingen voor SCD type II attributen, verleng je de geldigheid van de meest recente versie. Als er nog geen gegevens aanwezig zijn, of er zijn wijzigingen voor SCD type II attributen, voeg je een nieuwe versie toe. Daarnaast is het ook zo dat, als je voor een attribuut van SCD type I een wijziging ziet, je dit wil overschrijven voor *alle* versies die momenteel aanwezig zijn.
- Bij het inladen van de gegevens in de feitentabel, zal het nodig zijn om vreemde sleutels op te zoeken. Dit komt omdat de inkomende data voorzien is van *natuurlijke* sleutels die verwijzen naar dimensies. In het warehouse zijn deze niet langer uniek en dus ook geen sleutels. Het is daarom nodig om de natuurlijke sleutels te mappen op sleutels uit dimensietabellen. Zie ook pg. 47-50 in de syllabus voor een gedetailleerd voorbeeld.

TECHNISCHE IMPLEMENTATIE

De implementatie van het ETL-proces dient te gebeuren in de scriptingtaal van PostgreSQL genaamd PL/pgSQL. Deze scriptingtaal voegt enkele nuttige constructies zoals condities, loops… toe bovenop standaard SQL. Bovendien kan je variabelen declareren binnen de scope van een blok, hetgeen de zaken vaak sterk vereenvoudigt. Algemene documentatie kan je vinden op deze webpagina. De belangrijkste zaken worden hieronder samengevat.

De basisconstructie voor een script is de volgende:

```
do $$
declare
-- Declaraties van variabelen
begin
-- Inhoud van het script
end;
$$ language plpgsql;
```

Het script wordt typisch geschreven tussen $$ en $$ wat ervoor zorgt dat “escaping” van quotes niet nodig is. Variabelen worden gedeclareerd in het “declare” construct en het script komt tussen de “begin” en “end” keywords.

Aangezien je verschillende snapshots hebt, zal je het script verschillende keren moeten laten lopen. Het is dan ook handig dat je aan de interpreter laat weten in welk schema je wil werken. Je kan in het script het standaard schema waarin je werkt, vastleggen met:

```
set search_path to <schema_name>
```

Boodschappen kunnen worden afgedrukt met het commando “raise notice” gevolgd door een string. In die string kan je het karakter ‘%’ gebruiken om de waarde van variabelen te injecteren. Bijvoorbeeld:

```
raise notice ‘Hello, %’, your_name
```

Condities kan je testen met de constructie

```
if(<condition>) then

end if;
```

en loops nemen de vorm

```
for x in … loop

end loop;
```

Enkele constructies die zeer nuttig zijn bij het opzet van je ETL-proces zijn hieronder opgesomd.

- Met “generate_series” kan je reeksen genereren van bijvoorbeeld getallen en datums. Deze functionaliteit is handig in het maken van mini-dimensies of dimensies waar de “closed world” veronderstelling opgaat.
- De vensterfuncties lag() en lead() zijn heel nuttig in het vergelijken van twee opeenvolgende tuples. Dit komt van pas bij het ondersteunen van versiebeheer in dimensietabellen.
- De machine die je ter beschikking hebt, is een PostgreSQL 9.5 server, wat betekent dat je bij insert statements kan specificeren hoe moet worden omgegaan met bepaalde conflicten via de ON CONFLICT clause. Zo kan je bij het overtreden van bepaalde constraints (of wanneer bepaalde voorwaarden zijn voldaan) kiezen voor een update van de data in plaats van een insert. Dit komt van pas bij versiebeheer.
- De “with” clause van een query laat toe om een complexe query op te delen in een aantal eenvoudige SQL fragmenten. Dit helpt bij het overzichtelijk houden van je code en is vaak ook handig in het debuggen.
- PL/pgSQL is een procedurale taal. Je kan zelf functies aanmaken via “create function”.

Merk tot slot op dat, hoewel hier over “het script” wordt gesproken, je er allicht alle baat bij hebt om je code wat te organiseren in verschillende bestanden. Je kan bijvoorbeeld een apart bestand voorzien voor elke tabel. Dit houdt de zaken overzichtelijk en bovendien kan je op die manier in een verdere fase van het project heel gericht verbeteringen aanbrengen.

VERSLAG

In het uiteindelijke verslag van je project, voeg je je script files als bijlagen toe. Zorg ervoor dat je je scripts hebt uitgevoerd tegen het eind van het project. Het is **niet** de bedoeling dat wij deze zullen uitvoeren. Voorzie in je verslag een korte bespreking van de verschillende stappen van het ETL-proces ter verduidelijking van je code. Verlies je hierbij niet in details, maar bespreek de belangrijkste zaken.