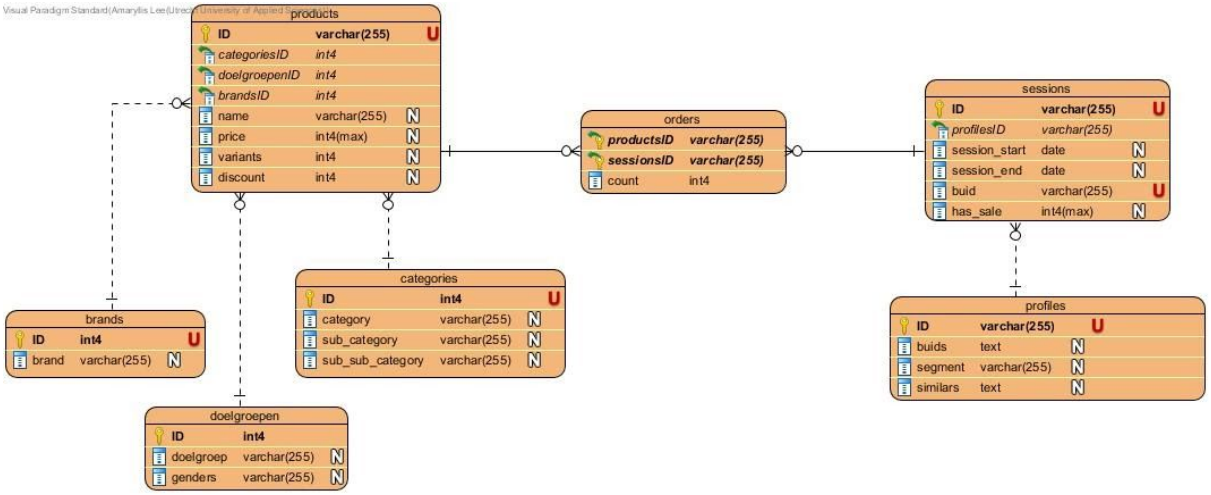
Document Store naar Relationele Database

Studenten:
Ruben van Raaij
Koen van Heertum
Floris Videler
Amaryllis Lee

Voor het vak Structured Programming hebben we de opdracht gekregen om een actuele dataset van de HU te leren begrijpen. Vervolgens moesten wij een relationele database ontwerpen waarin de juiste data opgeslagen kan worden.
Als laatst moesten we een programma schrijven in een procedurele programmeertaal die de gegevens overzet van de document store naar de relationele database. Dit document geldt als onderbouwing voor de database en de code

Ontwerp Relationele Database



Als hoofdtabellen hadden we gekozen voor Products, Sessions en Profiles. Vervolgens hebben we alle relevante informatie van products in aparte tabellen. Hierdoor hadden we losse tabellen voor categories, doelgroepen, brands, genders, variants, sessions en discounts. Hierna zijn we gaan kijken welke van deze weer terug konden bij products, en welke als aparte tabellen konden blijven. We hebben categories, doelgroepen en brands als losse tabellen gelaten, terwijl de rest gemerged is bij products of doelgroepen. We hebben categories als losse groep gelaten omdat het handig is om makkelijk op categories te zoeken later, en zodat we een aantal universele categories hebben waar geen typefouten en dergelijken in staan. We hebben doelgroepen een losse tabel gemaakt omdat je ook hierbij universele richtlijnen kan hebben. Wel hebben we genders hier aan toegevoegd, zodat je makkelijk kan zoeken op producten die bijvoorbeeld gericht zijn op vrouwen tussen de 20 en de 40, of mannen van 65+. Als laatst hebben we brands als losse tabel gemaakt omdat je hierdoor een lijst van universele merken hebt waarin je makkelijker dingen kan veranderen als hier wat mee gebeurt. Alle producten van een bepaald merk zouden op deze manier

makkelijker geschrapt kunnen worden of een merk zou makkelijk omgezet kunnen worden naar een nieuwe naam.

Vervolgens hebben we sessions aan profiles gelinked. Hierbij hebben we specifiek de keuze gemaakt om de *buid* in beide tabellen te laten staan. De buid in sessions is namelijk de buid van die specifieke session, terwijl de buids in profiles een array is van de meerdere buids die aan een profile hangen. Omdat array geen optie in de database is, hebben we gekozen voor *text* als datatype van deze buids.

Als laatst hebben we sessions aan products gehangen via de koppeltabel *orders*. Deze koppeltabel hebben we orders genoemd omdat in deze sessions een order geplaatst is op een product. Via deze koppeltabel kunnen we producten aan de sessions hangen, waardoor we via een aantal *joins* kunnen zorgen dat de producten aan de profiles gehangen kunnen worden. Hierop kunnen we recommendations bazeren. Bovendien hebben we in deze koppeltabel nog een count toegevoegd. Deze kan verhoogd worden als hetzelfde product meerdere keren besteld is. Hierdoor komen dezelfde producten die meerdere keren geordered zijn door dezelfde klant, in één koppeltabel te staan.

Gekozen Oplossing

cursor.execute(verwijder alle tabellen als ze eerder zijn gemaakt met gegevens)
cursor.execute(maak tabellen aan met hun relaties en keys)

We hadden gekozen en om eerst de tabellen te verwijden en daarna opnieuw maken , want er wordt continu een updated tabellen gemaakt .

Vervolgens is ervoor gekozen om eerst de benodigde data in csv files te importeren.Hierdoor is er een back up van de data beschikbaar.Bovendien wordt de data sneller in de relationele database geïmporteerd.

```
with open('profiles.csv', 'r') as profs:
   next(profs)
   cur.copy_from(profs, 'profiles', sep=';')
   conn.commit()
print("Profiles copied!")
```

Verder wordt er gebruik maken van *copy_from* om de csv bestand in de bepaalde SQL tabel te importeren.

De pseudocode en de code is te bevinden op onze Github:
https://github.com/GameModes/1ATeam1