Database ontwikkel opdracht - Steam

Door Rick van Duijnhoven, S26 – 03-04-2015

1. [Database case beschrijving](#h.s0wb2svice1q)
2. [ERD - Entiteiten Relatie Diagram](#h.plmzau8jyzon)
3. [Database ontwerp](#h.qpvbulpekr1i)
4. [SQL Scripts](#h.6to9b9io2uq7)

# Database case beschrijving

Ik heb voor een van de vooraf goedgekeurde websites gekozen, namelijk store.steampowered.com. Dit is mijn keuze geworden omdat ik zelf elke dag gebruik maak van Steam, zowel voor gamen als contact met vrienden. Steam is een platform voor op pc (en sinds kort ps3/4) waar voornamelijk games worden verkocht. Steam heeft ook functionaliteiten zoals een vriendenlijst, gebruikers maken hun eigen accounts aan en gebruiken deze om met elkaar te chatten, samen te gamen en discussies te voeren op het forum.

Als je op de homepage komt, zie je meteen dat de site eigenlijk uit vier delen bestaat:

1. Store (de winkel, waar games worden verkocht, reclame staat voor games en nog veel meer).
2. Community (hier valt alles onder wat leden kunnen doen naast games kopen/aanbevelen. Denk aan een forum, delen van screenshots etc.)
3. About (aparte pagina met uitleg van wat steam nou eigenlijk is).
4. Support (hier ga je heen als je hulp ergens nodig bij hebt wat Steam-gerelateerd is).

De store bestaat uit een paar aspecten. Games (welke je kunt filteren op categorie), een lijst van featured items (games), een lijst van software (welke ook gefilterd kunnen worden op categorie), demos (mini-versies van games zodat je ze kunt uitproberen voordat je een game koopt), News (nieuws artikelen gerelateerd aan games) en een tab ‘For You’ waar aanbevelingen voor jou staan gebaseerd op de games die je al hebt.

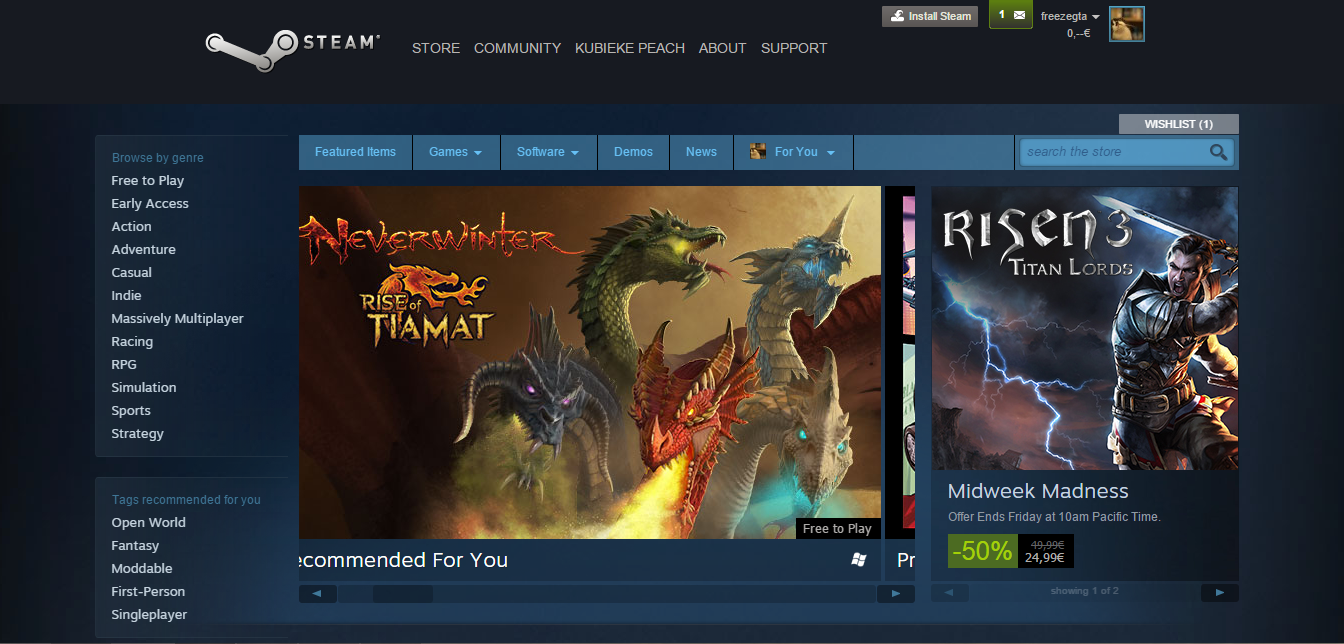
Elke game heeft een prijs, reviews, en informatie over het spel zelf zoals minimale specificaties van je pc om het spel te draaien.

Om gebruik te maken van de services van steam moet je een account hebben, je account bevat de volgende informatie: username (staat vast), wachtwoord, display name(kan continu zelf veranderd worden door gebruiker), steam balans (soort steam portemonnee), e-mail adres en natuurlijk een lijst van games welke je gekocht hebt (deze staan in de gebruikers Library). Op het profiel kun je zien bij welke groups die persoon hoort, welke avatar hij heeft (plaatje welke hem ‘identificeert’, vrij te veranderen door de gebruiker), welke content hij heeft geüpload (denk aan screenshots, video’s e.d.) en welke badges de gebruiker heeft verdiend door bepaalde spellen te spelen. Elke gebruiker heeft ook een inventory met spullen welke hij verdiend/verzameld heeft in bepaalde games.

Een gebruiker kan een of meerdere artikelen toevoegen aan zijn winkelmandje. Hij kan ook meerdere kopieën van hetzelfde spel of stuk software toevoegen, voor de extra kopieën krijgt hij dan een gift-code die hij aan iemand anders kan geven, zodat die persoon ook de game of het stuk software heeft. Vanuit het winkelmandje kan de gebruiker gaan afrekenen, hier moeten enkele gegevens over hem/haar ingevuld worden(zoals NAW, voor de veiligheid), waaronder de manier waarop er betaald wordt. Na het afrekenen worden de games/software toegevoegd aan het account van de gebruiker en kan hij deze direct downloaden.

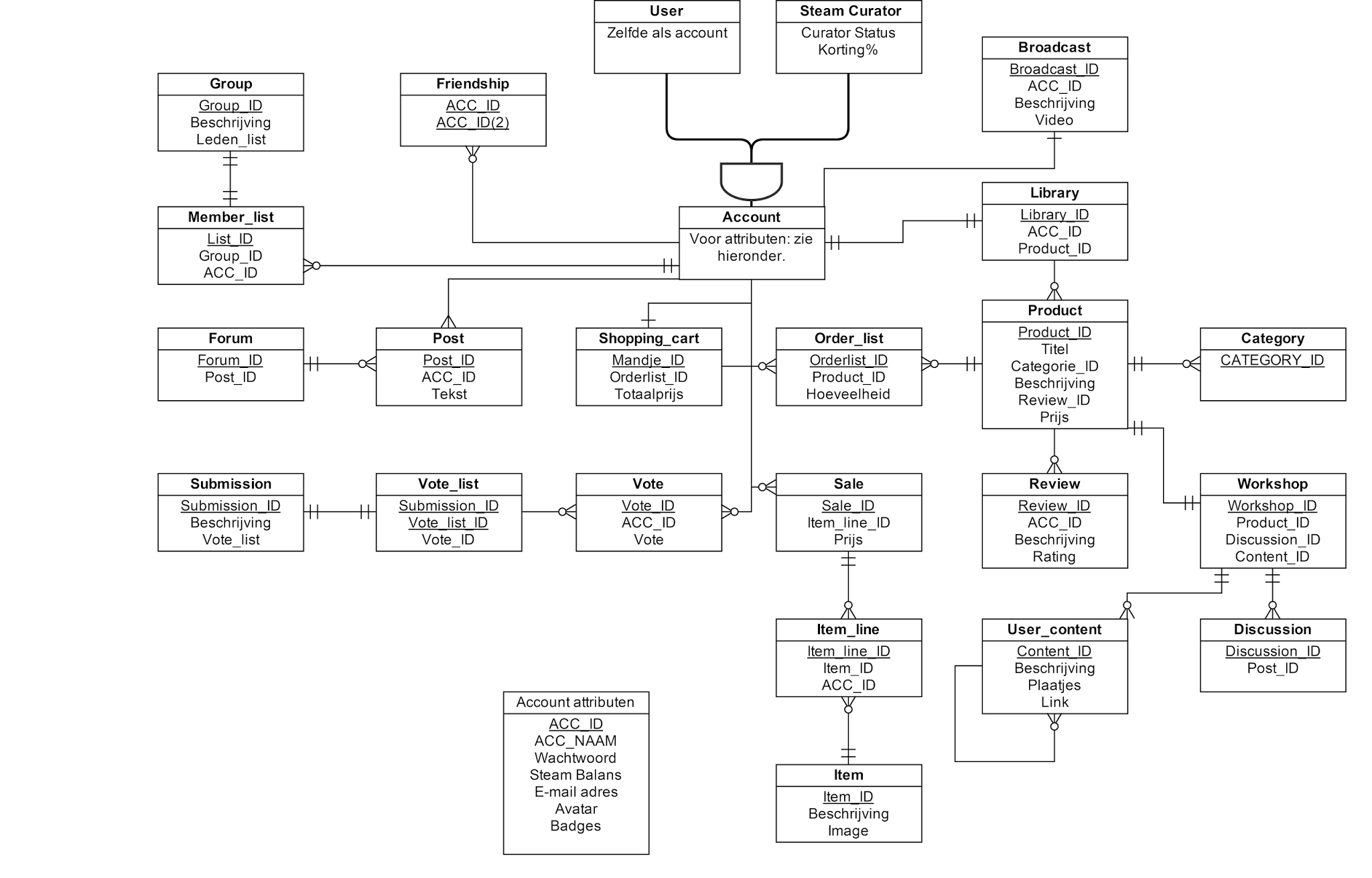
Gebruikers kunnen onder community deelnemen aan discussies op het forum, ze kunnen in de greenlight discussiëren en stemmen welke nieuwe games er naar steam moeten komen, de workshop is een verzamelplaats voor door spelers gemaakte inhoud, en tools om die inhoud te publiceren, organiseren en downloaden naar je games. Op de market kunnen spelers items uit hun spellen aanbieden om te ruilen of verkopen. Onder broadcasts kunnen gebruikers mensen live volgen wie games aan het spelen zijn en wie er voor kiezen om hun gameplay te streamen. Ze kunnen ook reviews achterlaten bij games welke zij in hun library hebben, om het spel aan te raden (of juist niet) aan anderen wie er aan zitten te denken om dat spel of stuk software aan te schaffen.

Enkele leden zijn ook zogenaamde ‘Steam Curators’. Zij reviewen(geven hun mening over) games op een professionele manier, om zo voor potentiële kopers duidelijk te maken of de games wel opleven tot hun verwachtingen. Gebruikers kunnen de Curators volgen om sneller te horen te krijgen wanneer ze een review uitgebracht hebben.



Aan de linkerkant zie je al duidelijk de verschillende genres van de games en aanbevelingen voor de gebruiker. Rechtsboven is te zien dat de gebruiker ingelogd is op zijn account en het eerder genoemde Steam Balans (Wallet).

# ERD - Entiteiten Relatie Diagram



Mocht het ERD hier niet goed te lezen zijn, hij zit er ook bij in het .zip-bestand, voor uw gemak.

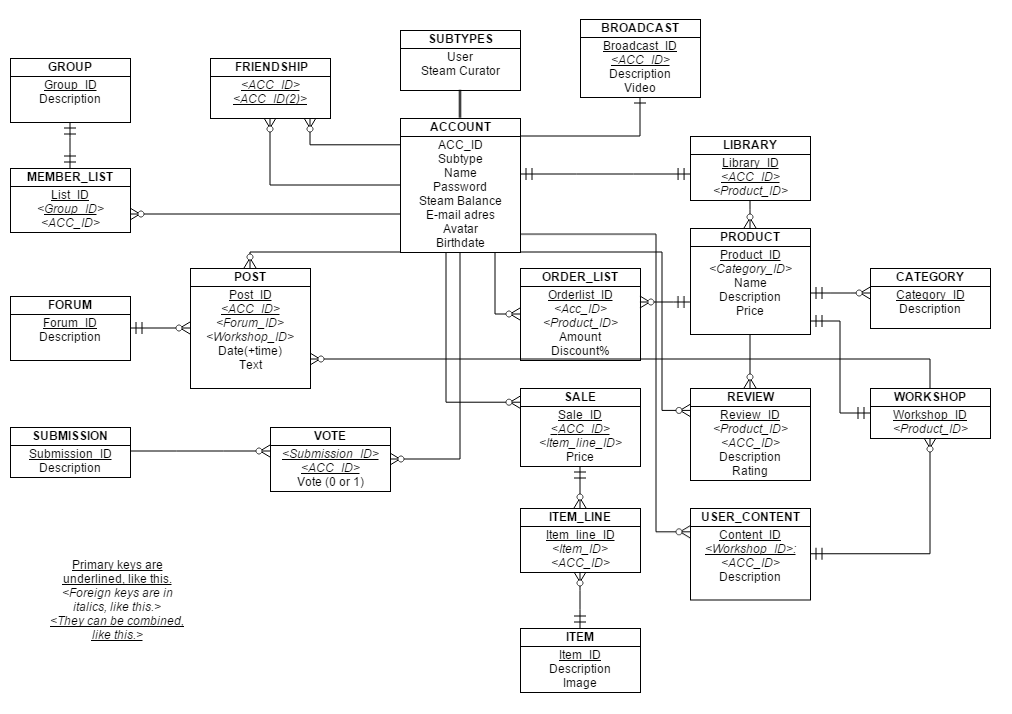
Feedback ERD

In eerste instantie had ik in mijn ERD al een aparte Friendship entiteit, waar twee verschillende foreign keys van Account in zaten. Hier kreeg ik als feedback op dat het in het ERD nog gewoon een recursieve relatie zou zijn.

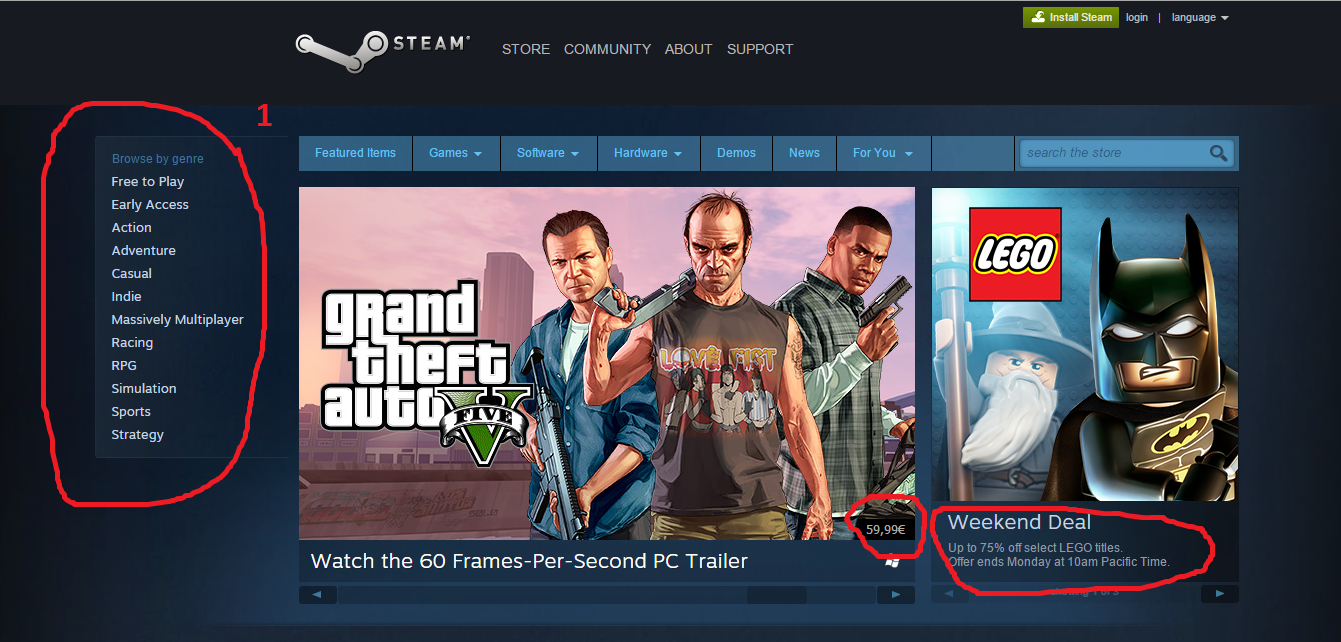
Dhr. De Loose: *“...de account en friendship entiteit. Als je de foreign keys weg laat, blijft er dan niet gewoon een recursieve relatie over?”*

Na hier even over nagedacht te hebben bleek dit inderdaad zo te zijn. Ik had in mijn model al meteen een oplossing voor de recursieve relatie erin geplaatst, terwijl deze had mogen blijven staan. Ik heb toen besloten om dit niet meer aan te passen, omdat de volgende opdracht was het ombouwen van het ERD naar een database ontwerp. Deze oplossing zou dan toch naar voren zijn gekomen, dus ik heb het maar zo gelaten.

# Database ontwerp



Mocht het ontwerp niet goed te lezen zijn, hij zit er ook bij in het .zip-bestand.



Hierboven ziet u een voorbeeld van de Steam home page. Aan de linkerkant zijn de genre’s (category in de database) te zien. Rechtsonder als voorbeeld is te zien dat elk product een prijs heeft en ook nog eens in de aanbieding kan zijn. Het kortingspercentage is in de database terug te vinden onder ‘Discount’ in de tabel ‘Order\_list’.

Voor een beschrijving van elke entiteit, zie bijlage: entiteiten beschrijving.

Voor mijn database ontwerp besloot ik om feedback te vragen aan een van mijn mede-studenten, van wie ik uit ervaring wist dat hij best goed is in het ontwerpen van ERD’s/database ontwerpen. Deze feedback komt van Jules Kreutzer, S26.

Jules heeft mij goed op weg geholpen met zijn feedback. Omdat het aardig wat feedback betreft, copy-paste ik het gewoon even hieronder, inclusief een plaatje (ook van Jules!) ter ondersteuning van de feedback. Hierna loop ik er punts gewijs even doorheen om uit te leggen wat ik met zijn feedback heb gedaan.

Jules: “1)

Misschien kun je op een of andere manier duidelijk maken welke velden van de ene tabel naar de andere gaan, is voor mij nu soms onduidelijk. Wat je misschien kan doen is de verbindings lijn op de juiste hoogte zetten, als je de tabel account en library heb, dan staat die verbindingslijn bij library iets te hoog, als die op de zelfde hoogte zou staan als <ACC\_ID> zou het misschien al duidelijker zijn (dan moet je het wel overal toepassen ;))

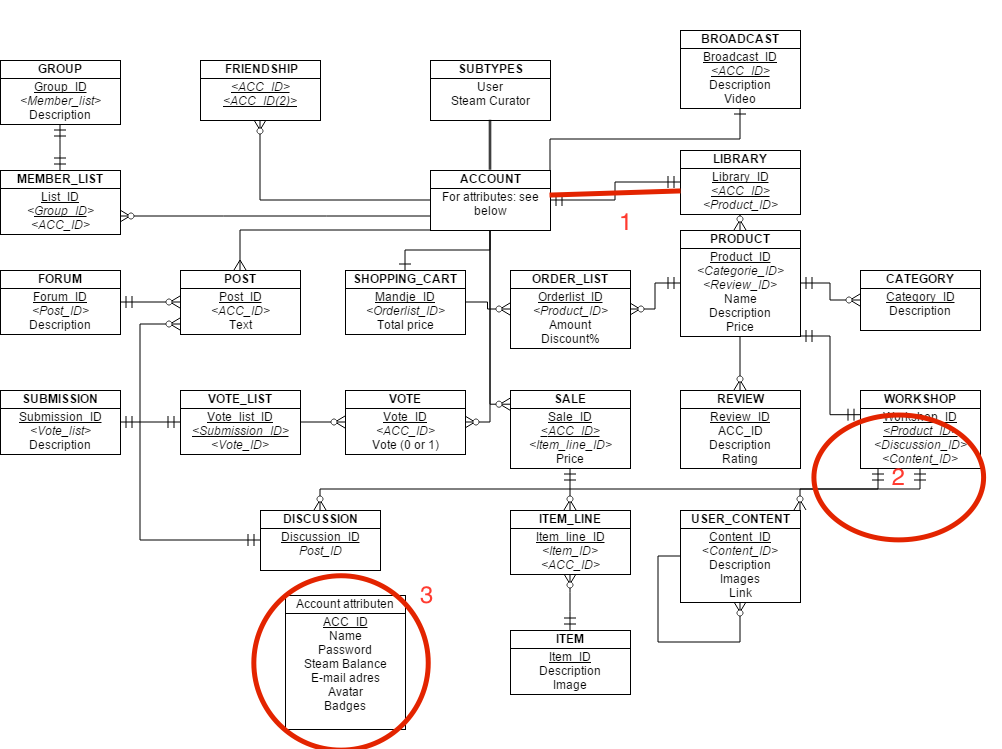
2)

Ik zou ook de verbinding tussen workshop en user\_content iets anders tekenen omdat hij nu over een andere verbinding heen loopt (van workshop en discussion) of doe het zoals bij de verbinding van account met sale, vote en shopping\_cart.

3)

weet niet of je het extra hebt gedaan, maar misschien de naam van de tabel voor account attributen ook nog in hoofdletters zetten, of alleen de tekst account.

Ik heb een legenda erbij gezet hoe ik primary keys en foreign keys weer geef, weet niet of je dat in de beschrijving hebt staan maar misschien is dag ook handig?



1. Over de scheve lijntjes: dit was mij nog niet echt opgevallen, maar na dat ik dit te horen heb gekregen was ik het er helemaal mee eens. Dit soort lijntjes heb ik op meerdere plekken in het ontwerp terug gevonden en overal heb ik ze aangepast naar een nette, rechte lijn.
2. Overlappende lijnen: ook deze kwam ik op meerdere plekken tegen in het ontwerp, nadat Jules me er op gewezen had. Dit creërde inderdaad onnodige verwarring en ik heb ze overal waar ik ze tegen kwam zo aangepast dat er geen lijntjes meer overlapten.
3. Yup, heb nu van het extra tabelletje met account attributen de titel van de tabel in hoofdletters gezet. Ook heb ik op zijn aanraden een kleine legenda met uitleg van de notatie van primary en foreign keys erbij gezet.

11-03-2015

Vandaag heb ik de feedback ontvangen dat de originele recursieve relatie van account veel-op-veel was. Dit heb ik aangepast in het ontwerp door nog een relatie tussen account > friendship te leggen, zo is de veel-op-veel correct verwerkt. Deze tweede relatie was nog nodig omdat er twee ACC\_ID’s in een ‘Friendship’ zitten.

Hiernaast hebben we het nog even gehad over de shopping\_cart entiteit. Dit bleek typisch iets te zijn wat eigenlijk in een ‘cookie’ in de browser wordt opgeslagen. Het zou ook niet logisch zijn om hier een tabel voor aan te maken (naast order\_list) omdat deze na elke transactie toch weer leeg zou zijn.

De relatie tussen user\_content en user\_content\_list is ook veranderd. Dit was eerst een een-op-veel relatie van user\_content naar user\_content\_list, nu gaat deze relatie beide kanten op. Dit omdat elk stukje content wat in user\_content\_list ook weer afhankelijk kan zijn van andere user\_content.

22-03-2015

Afgelopen week zijn we voor de proftaak langs geweest bij het bedrijf Copaco. Hier hebben we onder andere de feedback gekregen dat het vaak niet handig is om lijsten bij te houden. Vandaar dat er nu geen vote\_list meer is, maar bij elke vote wordt bijgehouden waar er op gestemd is. Zelfde geld met posts, deze behoren nu tot of een forum, of een workshop i.p.v. dat hier nog een entiteit tussen stond.

# SQL Scripts

Aangezien de scripts best wel lang zijn geworden, staan ze apart als bijlage in het .zip-bestand.

***Belangrijk***: om een goede werking van de select-statements te garanderen, draai de scripts in deze volgorde:

1. Create tables steam.sql – maakt alle tabellen van de database aan.
2. Insert data steam.sql – zet data in de tabellen.
3. Select statements.sql – laat relevante data zien welke ook gebruikt wordt door steam zelf.

Om de werking van de toegepaste constraints te testen, heb ik ook een apart .sql script gemaakt: Test constraints.sql

Deze kan op elk moment gedraaid worden, zolang het create script maar gedraaid is van tevoren.

**Om de werking van de Select statements te garanderen, zorg voor het volgende: als u net het script om de constraints te testen hebt gedraaid, zorg dan dat u eerst de create en inserts scripts nog een keer draait alvorens naar de select statements te gaan!**