Data Management

Notities 18/04/2024

Optionele opdracht:

* We hebben 2 examens
  + 1 theorie
    - Schriftelijk examen
  + 1 praktijk
    - Op eigen laptop
    - Geen internet
    - Geen communicatie (duh)
    - Eigen verantwoordelijkheid dat laptop werkt en dat de applicaties gedownload en werkende zijn
    - Bestanden op laptop mogen gebruikt worden
* Wat wordt er voor de optionele opdracht verwacht?
  + Minstens 1x per maand uploaden
  + Minstens de helft van de oefeningen proberen oplossen en op de repository zetten
  + Moet jou werk zijn!!!
  + Als dit goed is mag je je eigen notities, samenvatting, de cursus etc gebruiken (voor het theorie examen)

Inleiding procedurele SQL:

ISO standaard:

* = persistent stored modules
  + Je gebruikt je data om je producten in op te slagen??? (klopt dit? Zie opname +-13u44)
* Kan niet zomaar uw prooducten copy pasten naar andere producten???
* Programmeren op de server

Postgresql:

* Meeste PSM kunnen maar 2 talen, postgresql niet (kan er meer) (klopt dit???)

Vertrouwd en niet-vertrouwd:

* … (zie opname +-13u45)

…

How Trustly and Zalando are using PostgreSQL:

* Gebruiken PostgreSQL om bijvoorbeeld u door te verwijzen naar een betaalpagina (klopt dit??? Zie opname +-13u47)

Objecten op de server:

* Welke objecten staan op de server (want programmeren op de server):
  + Stored procedures
  + Stored functions
    - Vb: zie ppt
      * De effectieve code staat altijd ergens tussen (vb tussen ‘$$ … $$’)
  + Triggers

Verwerking:

* Krijgt codes om te zien of het goed werkt

Parameters Algemeen:

* Communicatie met buitenwereld
* 3 soorten
  + Invoerp
    - Worden ingegeven en (zie opname +-13u50)
  + Uitvoerp (niet bij functions (kan wel werken))
    - Zie opname hoe het juist gebruikt w
  + Invoer/uitvoerp
    - Zie opname hoe het juist gebruikt w
* Return (kan enkel bij …)

Voorbeeld IN OUT:

* $1 = … (zie opname +-13u52) (parameter???)

Functions vs …

* Zie opname +-13u53

Waarom?:

* Historisch verschil tussen berekeningen (functies) en manipulaties (procedures)
* Daarnaast:
  + Transacties kunnen enkel binnen procedures
  + Zo kan de planning daar rekening mee houden

Instructies mogelijk:

* Zie opname +-13u55

Structuur: functies:

* Zie ppt
* Waarom ‘ of $$
  + De kans dat er een ‘ in komt is groter (in de functies zelf)

…

Verwijder functies:

* Zie opname +-13u56

SELECT INTO voorbeeld:

* Zie opname +-13u58

$$ delimiter dialect:

* Zie opname 13u58

PERFORM:

* Als je SELECT vervangt door PERFORM moet je de uitvoer nergens in wegschrijven
* Waarom gebruiken?
  + In combinatie met FOUND
    - Met PERFORM krijg je niets terug, maar met FOUND wel (klopt dit??? Zie opname +-13u59)

RETURN(S):

* Kan een of meerdere dingen returnen
* Signatuur (met S (dus RETURNS)):
  + …
* Code (zonder S (dus RETURN)):
  + …
* Met of zonder S
  + Geeft aan of het 1 of meerdere dingen teruggeeft

Foutboodschappen:

* SQL-error-code: beschrijvende tekst
* SQLSTATE: code (getal)
* Bij EXCEPTION som je op wat je doet bij elke exception

RAISE:

* Zonder iets extra geeft het de laatste error terug, maar je kan er parameters erbij zetten
* Als je het op debug zet dan krijg je de meeste informatie terug

ASSERT:

* Je maakt een veronderstelling hier
  + Voldaan: geen probleem
  + Niet voldaan: krijgt een boodschap terug

SECURITY:

* Vb je hebt een functie gemaakt om een tabel te droppen, je geeft een persoon execute rights, die kan een tabel droppen die die eigenlijk niet mag droppen
  + Hiervoor kan je SECURITY toevoegen
    - (definer:) Je geeft rechten van de eigenaar door aan de persoon die die functie gebruikt (zoals sudo)
    - Met invoker w de functie uitgevoerd met de rechten van de uitvoerder, niet de eigenaar

LEVEL of immutability:

* IMMUTABLE
  + = (zie ppt)
* STABLE
  + = geen aanpassingen
* VOLATILE
  + = standaard

Transactie voorbeeld: stored procedure:

* … (zie opname +-14u07)

Praktisch: script:

* CREATE OR REPLACE: als het al bestaat dan w het vervangen en anders w het gewoon aangemaakt

Triggers:

* Net zoals functies en procedures zijn ze opgeslagen op de DB
* MAAR: w automatisch uitgevoerd (denk aan events in programmeren)

PostgreSQL:

* Triggers roepen trigger functies op
  + Je maakt een triggerfunctie en je gebruikt de trigger om die op te roepen

Voorbeeld: trigger functie:

* De new (van new.spelers) komt van de nieuwe data die aan de spelers w toegevoegd
* Je kan die code in meerdere tabbelen gebruiken (je gebruikt de trigger om die op te roepen)

Voorbeeld: trigger:

* OLD en NEW verwijzen naar de huidige toestand en de nieuwe toestand
* After insert:
  + After = tijdstip/moment (after laat blijken dat er ook een before is)
  + Insert = iets dat gebeurt/een instructie (insert laat blijken dat er ook een … (zie opname +-14u14) is)
  + (iets over iets in commentaar zetten)

Onderdelen trigger:

* Trigger-moment + trigger-gebeurtenis:
  + Wnr activeren:
    - AFTER
    - BEFORE
    - INSTEAD OF
  + Voor welke rij activeren
    - FOR EACH ROW
    - FOR EACH STATEMENT
  + Voor welke gebeurtenis:
    - INSERT
    - UPDATE
    - DELETE
    - TRUNCATE
      * TRUNCATE = een hele tabel deleten??? (zie opname +-14u16 voor meer uitleg)

Syntax:

* Foreign keys zijn eigenlijk ook een soort van triggers
* … (zie ppt + opname +-14u15)
* REFERENCING:
  + Vb OLD en NEW wil je door … kunnen oproepen (zie opname +-14u17)
* CREATE/DROP/ALTER TRIGGER
* … (zie opname +-14u17)

Standaard:

* De standaard laat meer acties dan een trigger functie toe

Notas:

* Er zitten verschillen tussen producten
  + Meerdere triggers op 1 tabel? En wat met de volgorde?
    - Bekijk de documentatie van de producten
      * De standaard is degene die als eerste is geschreven w als eerste uitgevoerd
  + Kan een trigger een andere trigger activeren?
  + … (zie ppt en opname +-14u19)

Gebruik:

* Denk gebeurtenis gestuurd
* Voorbeelden:
  + (Integriteits)regels
    - Triggers kunnen dit voor jou nakijken
  + Audit
  + Consistentie
    - Triggers gebruiken om te controleren dat het altijd hetzelfde doet??? (klopt dit? Zie opname +-14u20)
  + … (zie ppt + opname +-14u21)

Voordelen:

* … (zie ppt +- opname +-14u21)

Nadelen:

* Nadenken over architectuur en code organisatie
  + Vb niet alles op de DB zetten en er enkel een front-end aan verbinden
* Algemeen zoals bij andere talen:
  + Logische fouten vs syntaxfouten
    - Vooral oppassen voor logische fouten

De derde golf/Ethiek:

Nep nieuws waar iedereen in gelooft:

Feiten?

* Kwetsbaarheid maatschappij
  + Mensen zijn soms emotioneel ergens van overtuigd en die gaan heel hun rationele brein gebruiken om dit te ondersteunen
* Hedendaagse verbondenheid werkt in 2 richtingen
  + De gemiddelde mens …

Humanetech stelling:

* Sociale media wilt gwn dat je er zo lang mogelijk opblijft (heeft een dubbelgezicht)
* Eerste AI doode is een Belg (heeft alles gedaan wat zijn AI zei)
* Blue Whale challenges
* Trump die gwn mag liegen en bedriegen en die w gwn aanvaard door een hele groep mensen, maar politici die vegetarisch zijn en 1x een stukje vlees eten die direct heel veel commentaar opkrijgen

Feiten:

* Een van de dingen die je je kan afvragen is ‘wat is de drijfveer’
  + Vb: Facebook is een business product, is niet voor uw bestwil
  + Nieuwsbronnen
* If it’s free, you’re the product
* Verantwoording
  + Is heel moeilijk om van Google (als die iets fout doen) verantwoording af te dwingen

AI:

* De 7 doodzonden en een krachtig wapen, je kan dit gebruiken om mensen te beïnvloeden
* Kan AI:
  + Liegen?
  + Zie ppt
* Evolutie kunstmatige intelligentie
  + …

Beveiliging:

* Wat moeten we beveiligen
  + Data
  + Toegangscontrole
  + Hardware
  + Het platform waarop het draait
  + DB software
  + Binnen SQL zelf
  + SQL als “vertaler”
  + Applicaties
  + Gebruikers
  + Algemeen terugkerende security problematiek (bv CIA (confidential, I…, A…), AAA (Authetication (= ben je wie je zegt dat je bent), Authorization (= waar heb je toestemming toe), Accounting (= verantwoording)))

Databank software:

* Up to date houden (beveiliging zit in de infrastructuur)
* Configuratie
  + Local
  + Remote
    - E.g; zie ppt
* Known exploits -> wat doe je?

Binnen SQL zelf:

* User Management (roles)
* Privileges (grant/revoke)
* Views (beperken wat bepaalde personen zien)
* Stored procedures (in de code zeggen wat allemaal mag en niet mag)
* Row Security Policies (iets extra op grant/revoke dat fijner kan werken)
* Aandacht voor hebben in SQL:
  + SQL Injection
    - SQL w niet gecompileerd, w gewoon vertaalt (is gewoon tekst die er binnen komt, dus als je daar iets kan tussengooien kan dat voor heel wat problemen zorgen

SQL Injection:

* Zie ppt
* Vb: je kent de code van een formulier niet, maar je kan wel de inhoud guessen dus je smijt daar iets van tussen (klopt dit? Zie opname +-15u06)

Incorrectly Filtered Escape Characters:

* Als je niet filtert op escape characters kan dit voor problemen zorgen (in vb: ‘)

Incorrect Type Handling:

* Types van de input w niet gecontroleerd

Oplossingen:

* Escape characters verwijderen uit de invoervelden
* Invoer validatie
* …
* …

Opgepast:

* Enkel escaping is niet voldoende
  + Het is makkelijker om te zeggen wat wel mag dan wat niet mag
    - Als je wat niet mag gebruikt, kan iemand nog altijd creatief zijn en nog steeds binnen geraken

Handige functies:

* Quote\_indent(text) -> w tekst
* Quote\_literal (zie ppt + opname +-15u10)

Dieper:

* Er moet een kolom zijn die kan bevestigen of een gebruiker een echte gebruiker is die in de databank staat
  + Vb: ON users TO managers
    - Door de USING krijg je als manager ook enkel de dingen te zien die jij specifiek mag zien (vb: je bent manager 1, dan mag je enkel de dingen die manager 1 mag zien zien)