Data Management

Notities 22/02/2024

**Connection Pooling:**

Doel:

* We willen connecties beperken

Wat?:

* Pool = cache van de connecties van de databank
* Kan zowel client side als server side poolen
* Conclusie:
  + Met pooling: je bent niet direct/rechtstreeks aan de databank geconnecteerd
    - Haalt resources naar beneden omdat die vb 3 connecties met de pooling over een verbinding naar de databank te sturen
  + Zonder pooling: je bent direct/rechtstreeks met de databank geconnecteerd

Verschillende vormen:

* Transactie pooling:
  + Je kan enkel dingen doen die binnen de transactie nuttig zijn
  + Omgevingsvariabelen kunnen fucken met wat je verwacht in uw sessie
    - DUS: wat andere aanpassen
* Instructie pooling:
  + Elke instructie wordt apart doorgestuurd
    - Bespaart resources
    - Is heel beperkt in wat je kan doen

Prepared statements:

* Prepared statements werden vroeger onthouden en de databank was dan confused en gooide een error omdat die dacht dat die die nog een keer kon oproepen (die onthoud die)
  + MAAR: je kan dit omzeilen door ergens in advanced bij uw dashboard ofzo instellen dat die prepared statements niet onthoudt

**Data(banken) geschiedenis:**

DBMS soorten:

Hiërarchis:

* Onstaan in jaren ‘60
  + Om raketten naar de ruimte te sturen
* Ondertussen kan er ook SQL op toepassen
* Bouwstenen:
  + Segmenttypes ipv tabellen
  + Parent-child relationships
    - 1 op veel relaties
  + Wortelsegment (root) en bladeren (zoals bij bomen en grafen)
  + Veel op veel relaties niet toegestaan
* Enkele eigenschappen
  + Als een bestand in een map staat, kan dit niet in een andere map staan
  + Voordelen:
    - Eenvoudige …
    - …
  + Nadelen:
    - …
    - …
  + …
    - …
* ER-model naar hiërarchisch model:
  + Je moet beslissen wat de parent is en wat de child
  + …
* Leefregels van hiërarchisch model
  + …
  + …
* Terminologie hiërarchisch model
  + Parent-segment
  + Child-segment
  + …
  + … (iets over volgorde)
  + …
* Voorbeeld:
  + Dokterspraktijk met meerdere dokters. Men wil de gegevens van de consultaties in een gegevensbank opslaan
    - Dokter en patiënt op dezelfde hoogte beide verbinden met de consultaties
      * Niet hiërarchisch
    - Dokter als parent voor patiënt patiënt als parent voor consultatie
      * Redundantie als de patiënt meerdere dokters heeft (w dan minstens dubbel weergegeven in de db)
  + Reisbureau
    - Reis als parent van toerist en van bestemming
      * Zorgt voor minder redundantie
  + School:
    - Klas als parent van leerling en vak; vak als parent van handboek

Netwerk:

* Nu relationeel database model (is een beetje een tussenstap geweest tussen hiërarchisch en relationeel model)
* Bouwstenen
  + Recordtypes (= zoals segments bij hiërarchisch)
  + …
  + …
  + …
* ER-model naar netwerk model:
  + …
* Verschil hiërarchisch – netwerk
  + Kan een recordtype zijn in meerdere settypes
  + … (iets over meerdere relaties of zoiets)
  + …
* Gegevensbanktalen:
  + Gegevensdefinitie taal: DDL
  + …

**Vergelijking met relationele structuur:**

Hiërarchische structuur in een relationele databank:

* Gaat dit?
  + Ja:
    - Een-op-veel relaties
    - Past er perfect in
* Vormen
  + Rechtstreekse omzetting (IMS)
  + …

Netwerk structuren in een relationeel model:

* Gaat dit?
  + Ja
    - Is een soort van tussenstap geweest tussen hiërarchische en relationele
    - Is geëvolueerd naar eerder relationeel nu

Terminologie:

* Relationeel
  + Tabellen
  + Rijen
  + Kolommen
* Netwerk
  + Recordtypes (+settypes)
  + Records
  + Fields
* Hiërarchische
  + Segmenten
  + Records
  + Fields
  + W sequentieel gelezen (goed voor vroeger toen 5mb geheugen een meter groot was bij wijze van spreken (om het zo efficiënt mogelijk te gebruiken)

**Subqueries 1:**

Niet-gecorreleerde subqueries:

* Hoe zou je de output van een SELECT statement beschrijven
  + Als een tabel

Wat?:

* Je kan subqueries blijven schrijven (in elkaar oa)
* Een tabelexpressie binnen een tabelexpressie
* Resultaat w aangeroepen …
* …

Waarom?:

* Query opsplitsen in deelproblemen

Soorten subqueries:

* Scalaire subquery: 1 rij, 1 kolom
  + Voorbeeld:
    - Geef voor elke planeet hoeveel groter of kleiner die is dan de zon
      * Deelprobleem = grote van de zon
        + Zie ppt voor oplossing
    - Geef de hemellichamen met een diameter groter dan Venus
      * Deelprobleem = grote van Venus
        + Zie ppt voor oplossing

Subquery staat in de WHERE

* Rij subquery: 1 rij
  + Voorbeeld:
    - Geef voor alle spelers met hetzelfde geslacht en dezelfde woonplaats als de speler nummer 7
      * Deelprobleem = geslacht en woonplaats van speler 7
        + Zie ppt voor oplossing

Subquery staat in de WHERE

* Kolom-subquery: meerdere rijen, elk 1 waarde
  + Voorbeeld:
    - Geef alle namen
      * Deelprobleem = wat zijn de planeten van ons zonnestelsel
        + Zie ppt voor oplossing

Subquery staat in de WHERE

Als je van het heelal of van de melkweg of whatever zou willen, dan moet je satteliet van leeg laten

* Tabel subquery: meerdere rijen en kolommen
  + Voorbeeld
    - Geef de reizen die een hemelobject bezoeken dat over alle reizen heen minstens 5x bezocht is
      * Deelprobleem = hemelobjecten die al minstens 5x bezocht zijn
        + Zie ppt voor oplossing

Subquery staat in de FROM

* Voorbeeld
  + Geef alle reizen die geen bezoek hebben gebracht aan de maan
    - Oplossing1 op ppt:
      * Reizen kunnen meerdere hemellichamen bezoeken en ook de maan komen er ook tussen te staan
    - Subquery:
      * Reizen die de maan wel bezoeken opzoeken
        + Die er dan uitfilteren
        + Subquery staat in de WHERE
      * Bij eerste oplossing met de subquery staat er een INNER JOIN, maar dat moet niet
* Uitdaging
  + Probeer deze zonder subquery te schrijven (op een regenachtige dag..., als je echt teveel tijd hebt…)
    - Wat toont de query:
      * Het gemiddelde die door de spelers aan boetes worden betaald (elke speler heeft hetzelfde gewicht)
    - Fouten:
      * (zie opname +-14u24)
      * Je kan geen average van … (zie opname +- 14u26)

**Subqueries 2:**

Gecorreleerde subqueries:

* Die hiervoor konden apart uitgevoerd worden

Subqueries in WHERE:

* …

Subqueries in FROM:

* …

Hoofdqueries en subqueries:

* Subqueries kunnen alles zien wat de hoofdqueries kunnen zien? (klopt dit? Zie opname +-14u40)

Gecorreleerde subqueries:

* …
* …
* Oefeningen
  + 1: geef voor iedere reis het bezoek met de langste verblijfsduur
    - Oplossingen:
      * 1: max verblijfsduur als het gaat over die reis (zonder WHERE in de subquery zorgt hiervoor (anders krijg je gwn de langste reis in het algemeen)
  + 2: geef de spelers die meer keer bestuurslid zijn geweest dan dat ze wedstrijden hebben gespeeld. Toon spelersnr.
    - In de HAVING w een subquery gestoken om het aantal wedstrijden te tellen (om die dan met de count van het aantal keer dat die bestuurslid geweest zijn)
      * Door de WHERE wordt er gekeken naar de mensen die bestuursleden zijn geweest en die hun wedstrijden

EXISTS operator oefeningen:

* Geef alle reizen met een bezoek aan Jupiter. Toon reisnr en …
  + Oplossing:
    - Kijken naar reis ofdat die een bezoek heeft gemaakt aan Jupiter
      * Hoe kan je de oplossing nog verbeteren
        + WHERE EXISTS als je gwn reisnr geeft (PK) is dat ook genoeg

WHERE EXISTS = waarvoor bestaat

NOT EXISTS operator:

* …
* …
* En deze?: (zie ppt)
  + Wat doet de query?
    - Gaat niets tonen, vanaf er bezoeken zijn is de WHERE NOT EXISTS in deze query gwn WHERE FALSE dus die toont niets
  + Verbetering van de query: correlatie toevoegen??? (zie ppt/opname +-14u51)

ANY en ALL operators:

* …
* …
* …
* ALL = voor alle moet dit gelden
* ANY = voor al die waarvoor het geldt
  + Altijd met < of > gecombineerd (ook zo bij ALL???)
* Oefening:
  + 1: Geef de langste reis = geef de reis waarbij de reisduur groter of gelijk is aan alle reizen. Toon reisnummer.
    - Deelprobleem:
      * Reisduur van alle reizen te weten te komen (we willen gwn de reis die groter is dan de grootste reisduur (je kan dit ook doen door de MAX operator))
        + WHERE reisduur >= ALL (SELECT reisduur FROM reizen)
    - Hoofdprobleem:
      * …
    - Oplossingen zie ppt
      * 1: WHERE reisduur >= ALL (SELECT reisduur FROM reizen)
      * 2: de WHERE in de subquery zorgt ervoor dat de reis die je vergelijkt, niet gelijk mag zijn dan de reis met wat je die vergelijkt. PROBLEEM: als er twee reizen zijn die even lang zijn + langer dan al de rest, dan wordt er geen weergegeven (want het moest groter zijn)
  + 2: geef alle reizen, behalve de langste reis
    - ANY gebruiken: er moet er maar een aan voldoen
* ANY en ALL operatoren: OPGELET:
  + Geef een lijst van alle planeten die groter zijn dan al hun sattelieten
    - WHERE van subquery …
    - PROBLEEM: Mercurius en Venus hebben geen sattelieten
      * DUS: een NULL operator geldt de ALL sws
        + Bij ANY werkt dit niet (dan geeft het een false? (klopt dit? Zie opname +-15u06)
      * Venus en Mercurius geven dus altijd TRUE terug
* ANY en ALL operatoren vs NULL:
  + Het moment dat je NULL tegenkomt moet je opletten
    - ALL: sws TRUE
    - ANY: sws FALSE

Unique operator:

* Geef de spelers voor wie er precies één boete betaald werd
  + (zie opname +-15u08)

OVERLAPS operator:

* …
* Een operator die vanalles voor jou kan doen (of het aan verschillende dingen voldoet (moet een stukje gemeenschappelijk zijn))
* …

Combinatie:

* Oefeningen
  + 1: geef de klant die op een reis zijn meegegaan waar ook een klant met klantnr 126 op meegegaan is. Toon klantnr
    - Deelprobleem:
      * Reizen van klant 126
    - Hoofdprobleem:
      * De reizen van alle klanten
    - Oplossing:
      * Correlatie: reisnr van de andere deelnemers moet overeenkomen met dat van klant 126
  + Geef de planeten die bezocht zijn op een reis waar klantnr 126 niet op meeging. Toon alle gegevens van de hemelobjecten
    - Hoofdquery: alle planeten vinden
    - Subquery: WHERE EXISTS (want er moet een reis bestaan waar 126 niet naartoe is gegaan dus daarmee niet NOT EXISTS) reizen
      * Nog een Subquery: WHERE NOT EXISTS klantnr 126

Extra oefeningen:

* 2:
  + Er worden spelers gevraagd die met speler 57 hebben gespeeld of voor een team waar 57 ook heeft gespeeld
    - Blok 1: WHERE NOT EXISTS
      * Van boven naar beneden
        + Alle spelers die nooit met speler 57 een wedstrijd hebben gespeeld
        + Alle spelers die nooit in een team heeft gespeeld waarvoor 57 ook heeft gespeeld
        + DUS: voor alle spelers die voor teams hebben gespeeld waar speler 57 ook heeft gespeeld (NOT NOT = wel)
    - Blok 2: AND spelersnr NOT IN
      * Vanaf vanonder te beginnen:
        + 1: Teams waarvoor 57 heeft gespeeld
        + 2: Teams waarvoor 57 nog nooit gespeeld heeft
        + 3: (met de AND) Teams waarvoor 57 heeft gespeeld (NOT NOT = wel)