Student 1: David Demmers 1760329

Aanleiding

Het bedrijf registreert aankoopgegevens van klanten bij supermarkten. Het is de intentie om de dienst aan te bieden aan veel meer supermarkten. Zowel het volume als snelheid van aankopen dat verwerkt moet worden zal daardoor stijgen. De groeiverwachting van het bedrijf is om in de nabije toekomst tien tot twintig keer zo veel data te verwerken, en bij uitbreiding naar het buitenland nog meer.

Kwestie

*(Beschrijf in eigen woorden waarom tot welke grenzen jij denkt dat de huidige oplossing kan opschalen. Probeer de onderliggende oorzaak te beschrijven van de beperkingen.)*

De complexiteit van de analytische query’s met; producten uit dezelfde tabel en vergelijkingen over verschillende tabellen gelimiteerd zijn nu al tot een snelheid die te langzaam is.

Het opschalen van de data of het loslaten van de query’s op alle mogelijkheden zou nu al niet slagen.

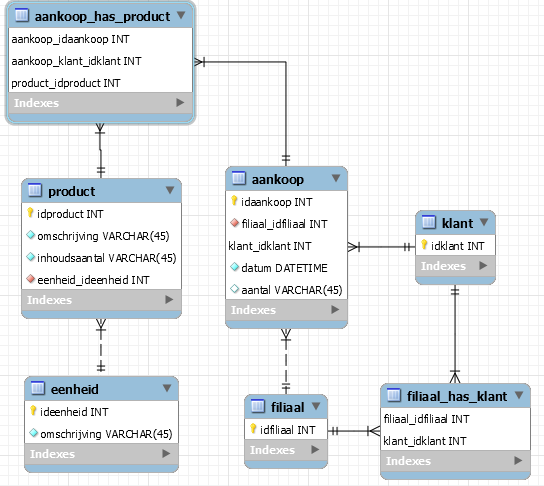
De reden hier voor is de verspreiding van de data.

|  |  |  |  |
| --- | --- | --- | --- |
| De vergroting van de data na een inner-product | | | |
| Omgeving grootte | Klein | Middelklein | Opgeschaald ***Geschatte waardes*** |
| Rijen | 20754 rijen | 10x~ 207.710 | *100x~ = 20.000.000~* |
| Q1 | 4.6~\* = 113.853 | 54~\* = 11.212.972 | *50~\* = 1.000.000.000~* |
| Q2 | 5.7~\* = 118.828 | 5.1~\* = 1.053.426 | *5~\* = 100.000.000~* |
| Q3 | - | - | *-* |

|  |  |  |  |
| --- | --- | --- | --- |
| Tijds duur voor het oplossen van de query | | | |
| Omgeving grootte | Klein | Middelklein | Opgeschaald ***Geschatte waardes*** |
| Rijen | 20.754 | 10x~ 207.710 | *100x~ = 20.000.000~* |
| Q1 | 3.8s | 50~\* = 193.6 | *268 uur* |
| Q2 | 0.69s | 50~\* = 34.6s | *48 uur* |
| Q3 | - | - | *-* |

Voorgestelde oplossing(en)

Het voordeel van mijn structuur is het samen voegen van sommige van informatie waardoor het opzoeken en combineren sneller zal gaan, alleen zal er wel redundantie voorkomen.



In dit model is te zien dat een aankoop moment uniek is aan het moment en aan welke klant dit doet, deze informatie kan je gebruiken zonder dit later op elkaar te hoeven vergelijken door een koppel tabel te maken van alle producten die gekocht werden in dat moment, ook is er duidelijk te zien in mijn model dat de klant meer aandacht krijgt door de filiaal tabel te koppelen met een N:M relatie.

Zo kan er gemakkelijk bijgehouden worden welke klant bij welke winkel is geweest, mochten er meer analyses gedaan worden als een eerste of laatste bezoek dan kan dat gemakkelijk worden toegevoegd aan de koppel tabel.

Dit en de invoer van aankoop\_has\_prodcut tabel zorgt ervoor dat er veel minder afhankelijk is van de aankoop tabel

Argumenten*.*

Ik heb het probleem na gebootst door een lokale mysql server te draaien met de complete (uitgebreide) database van 200 duizend rijen in de aankoop tabel.

Ook zijn er kleinere varianten gemaakt toen het duidelijk was dat de query’s al snel een te groot product maakte op de kleinste variant.

De schaalbaarheid is lokaal getest en daaruit geëxtrapoleerd is de data van 20 miljoen regels, dit zijn gunstige schattingen omdat de lokale machine maar met een taak tegelijk bezig is en dit is uiteraard anders in een productie omgeving.

Ook komt er veel meer overhead bij kijken dan wat er is mee gerekend voor 20 miljoen regels.

Conclusie

 Mijn is conclusie is breid de database uit, dit zorgt voor een beetje redundantie, maar wel een duidelijk inzicht in de data/informatie die er is zonder deze telkens samen te hoeven voegen.

En nog belangrijker zonder de grootst groenende aankoop tabel nog meer te belasten.

Bijlagen.

SELECT Count(DISTINCT( pairname )) AS totaal\_aantal\_klantparen

FROM   (SELECT Count(\*)

                      aantal\_gelijke\_producten,

               Concat(kl1.klant\_idklant, "-", kl2.klant\_idklant, "-", kl1.datum,

               "-",

               kl2.datum)                                        named,

               Concat(kl1.klant\_idklant, "-", kl2.klant\_idklant) pairname

        FROM   (SELECT \*

                FROM   aankoop,

                       klant

                WHERE  aankoop.klant\_idklant = klant.idklant) AS kl1,

               (SELECT \*

                FROM   aankoop,

                       klant

                WHERE  aankoop.klant\_idklant = klant.idklant) AS kl2

        WHERE  kl1.product\_idproduct = kl2.product\_idproduct

               AND kl1.klant\_idklant < kl2.klant\_idklant

        GROUP  BY named,

                  pairname

        HAVING aantal\_gelijke\_producten > 3) AS allrecords

LIMIT  30;

/\*

Vind welke productparen het vaakst (top-50) tegelijk gekocht worden (optioneel per filiaal) .

\*/

SELECT Count(\*) AS tegelijk\_gekocht,

              Concat(a1.product\_idproduct, "-", a2.product\_idproduct) pairname

FROM   (SELECT \*

        FROM   aankoop

        WHERE  aankoop.datum = aankoop.datum

        AND      aankoop.product\_idproduct = aankoop.product\_idproduct

        AND    aankoop.filiaal\_idfiliaal = 0) AS a1,

       (SELECT \*

        FROM   aankoop

        WHERE  aankoop.datum = aankoop.datum

        AND      aankoop.product\_idproduct = aankoop.product\_idproduct

              AND    aankoop.filiaal\_idfiliaal = 0) AS a2

WHERE  a1.product\_idproduct != a2.product\_idproduct

       AND a1.product\_idproduct < a2.product\_idproduct

GROUP BY pairname

ORDER BY tegelijk\_gekocht  DESC LIMIT  50

/\*

Als klant A filiaal 1 bezoekt, en klant B bezoekt filiaal 1 en 2, en klant C bezoekt filiaal 2,

dan kun je stellen dat klant A en B 0 filialen van elkaar verwijderd zijn, en klant A en C 1 filiaal.

Vind hoeveel filialen twee willekeurige klanten van elkaar verwijderd zijn.

***Niet werkende.***

\*/

SELECT DISTINCT a1.klant\_idklant, a1.filiaal\_idfiliaal, a2.klant\_idklant, a2.filiaal\_idfiliaal

FROM

(SELECT \*

FROM aankoop

WHERE aankoop.klant\_idklant = 0) as a1,

(SELECT \*

FROM aankoop

WHERE aankoop.klant\_idklant = 1) as a2

# GROUP BY a1.klant\_idklant, a1.filiaal\_idfiliaal, a2.klant\_idklant, a2.filiaal\_idfiliaal