Samenvatting Relationele Gegevensbanken 2017–2018

Contents

1 NoSQL

Relationele databanken zijn niet goed genoeg voor enorme toepassingen (bv Facebook, Google, Amazon, ...) wegens drie oorzaken:

1. Enorme hoeveelheden data (+ Petabytes)
2. Gedistribueerd over duizenden servers
3. Niet of wening gestructureerde gegevens

NoSQL betekent “niet relationeel”. Er bestaan verschillende soorten:

- Key/Value (bv. Redis, Voldermort, Riak
- Kolom-georiënteerde databanken (bv. Bitable, Cassandra, HBase, Hypertable
- Document-georiënteerde databanken (bv. MongoDb, CouchDb)
- Graaf-georiënteerde databanken (bv. Neo4J, FlockDb)

2 Gedimensioneerde Gegevensmodellering

Tijdens het ontwerp van een databank zijn er twee keuzes:

1. De transactiedatabank (gewone databank). Dit maakt gebruik van het gewone relationele model.
2. De datawarehouse (bijhouden van historiek). Dit maakt gebruik van gedimensioneerde gegevensmodellering .

De transactiedatabank wordt ontworpen met zo weinig mogelijk redundantie. Bij datawarehouses is zo een model te complex omdat zelfs bij eenvoudige queries vaak tabellen aan elkaar gelinkt moeten worden rekening houdend met de historiek .

Gedimensioneerde gegevensmodellering stelt tabellen van een databank voor in een sterschema. Dit schema bevat:

1. Slechts één feitentabel (kan petabytes aan informatie bevatten). Kenmerken van de feitentabel zijn:
   - Meervoudige sleutels die elk verwijzen naar een dimensietabel.
   - Meerdere attributen (de meetwaarden, meestal numeriek want ze moeten geaggregeerd kunnen worden.
2. Meerdere dimensietabellen. Kenmerken van een dimensietabel zijn:
   - Een enkelvoudige sleutel (meestal artificieël gegenereerd zonder logische betekenis).
   - Kolommen die relevant zijn bij de bevraging van de tabel.
   - De tabel moet niet aan de derde normaalvorm voldoen. De reden is omdat er dan minder tabellen moeten aangesproken worden. De query verloopt dus sneller, maar er zijn dus wel meer kolommen en redundante gegevens.

De feitentabel kan enkel geraadpleegd worden via een dimensietabel omdat het sequentieël aflopen van de feitentabel geen haalbaar alternatief is. Een query verloopt op volgende manier:

1. Men zoekt eerst in de diverse dimensietabellen op welke de sleutels zijn van de rijen die aan de selectiecriteria voldoen.
2. Vervolgens construeert men de sleutels in de feitentabel om de feiten effectief op te halen.

3 Implementatie van Joins

Er bestaan vier implementaties van joins:

1. Block Nested Loop Join

2. Index Nested Loop Join
3. Sort-Merge Join
4. Hash Join

De keuze ligt niet aan de programmeur, maar aan de databaseomgeving zelf. De database beslist aan de hand van bestaande indices welke implementatie gebruikt zal worden.

Er worden een aantal symbolen gebruikt om de tijdsduur van een algoritme te beschrijven. De gebruikte symbolen zijn:

- T : De uiteindelijke verwerkingstijd
- β_x : Het aantal disk-blokken voor tabel X
- β_y : Het aantal disk-blokken voor tabel Y
- $\#_x$: Het aantal rijen van tabel X
- $\#_y$: Het aantal rijen van tabel Y

3.1 Block Nested Loop Join

- Het eenvoudigste algoritme maar ook het slechtste
- Wordt uitgevoerd indien er geen indices zijn
- In vier geneste lussen worden elke rij van X vergeleken met elke rij van Y
  - De twee buitenste lussen lezen blokken van elke tabel in het geheugen
  - De twee inwendige lussen vergelijken elke rij van het eerste blok met elke rij van het tweede blok
- Het aantal blokken wordt te groot wat een impact heeft op performantie door het blokkeren van I/O
- Uitvoeringstijd: $T \sim [\beta_x + (\beta_x * \beta_y)]$. β_X komt overeen met de linker-operand van de join operatie, en bevat best dus het minst rijen om de uitvoeringstijd te doen dalen.

3.2 Index Nested Loop Join

- Wordt uitgevoerd indien alle optredende attributen van één van de tabellen geïndexeerd zijn.
- Drie geneste lussen
  - De mest uitwendige lus leest blokken in van de niet-geïndexeerde tabel X.

  - De middenste lus loopt doorheen elke rij van de blokken.
  - De binnenste lus zoekt via de indices naar de corrseponderende rijen van Y.

- Uitvoeringstijd: $T \sim [\beta_x + \#_x * f_y]$. f_y is een functie van de gebruikte indexmethode en van de aantallen rijen die aan het join-predicaat voldoen. De Index Nested Loop Join is bij een equijoin (een join waarbij het on predicaat een gelijkheid heeft. bv on "x.id = y.id"

3.3 Sort Merge Join

- Wordt uitgevoerd indien beide tabellen gesorteerd zijn volgens de attributen van het equijoin predicaat.
- Geen lussen
- De tabellen worden gemerged.
- Er wordt van elke tabel één blok in het geheugen genomen en wordt er telkens één rije beschouwd. Afhankelijk van de relatieve volgorde van deze twee rijen, wordt er een rij verder geschoven in één van de blokken.
- Uitvoeringstijd: $T \sim [\beta_x + \beta_y]$. Enkel indien beide tabellen al op voorhand gesorteerd zijn

3.4 Hash Join

idk

4 DDL Aspecten

DDL = Data Definition Language (create, updates enz)

4.1 Tabel-En-Kolomdefinities

...

4.2 Constraints

Constraints zijn beperkingen die opgelegd worden aan kolommen. De belangrijkste constraints zijn:

- Not Null
  - De waarde is verplicht
  - Enkel inline definitie van deze constraint is mogelijk
- Primaire Sleutel

  - De waarde is uniek voor elke rij
  - Wordt geïmplementeerd door “primaire index” (een algoritme)
  - Zonder primaire sleutel kunnen duplicate rijen optreden

- Verwijzende sleutel
  - Een sleutel die verwijst naar een rij in een andere tabel
  - Als een rij waarnaar verwijzen wordt verwijdert wordt, kunnen er drie acties optreden
    * No Action : Het verwijderen van de ouderrij wordt onmogelijk indien er nog verwijzingen naar zijn. Het kindrij moet dus eerst aangepast worden.
    * Cascade : Bij een wijziging van een ouderrij worden alle kindrijen trapsgewijs aangepast. Indien er andere regels zijn die deze aanpassingen tegenhouden, wordt de actie in het geheel afgebroken
    * Set Null/Set Default : Als een ouderrij aangepast of verwijdert wordt, kan de waarde van de verwijzende sleutel op de null/default waarde gezet worden. Indien er andere regels zijn die deze aanpassingen tegenhouden, wordt de actie in zijn geheel afgebroken.

- Unique
  - Om meerdere kolomcombinaties uniek te maken. Dit wordt vaak gebruikt bij kandidaatsleutels die niet primaire sleutel zijn. (bv primaire sleutel is “id”, de combinatie van “id” en “naam” kan unique gedeclareerd worden.

- Check
  - Wordt gebruikt om bv. intervallen op te leggen (enkel waarden tussen 10 en 20 toegelaten), of enkel woorden met de letters ’a’, ’b’, ’c’

4.3 Views

Een view is een aparte tabel dat uitsluitend bedoelt is om te lezen en te gebruiken in queries (analoog aan Common Table Expressions).

- Materialisation : De view wordt op voorhand uitgevoerd en de resulterende tabel wordt opgeslagen in het geheugen of de schijf

- Resolution : De view wordt selectief met de hoofdquery verwerkt.

Een view kan in meerdere queries gebruikt worden. Views kan in de meeste gevallen enkel door de database administrator aangemaakt worden. CTE heeft het voordeel dat iedereen ze kan maken. De inhoud van een view kan gewijzigd

worden, voor zover de view aan dezelfde restricties voldoet als wijzigbare CTE's. De with check option zal de view controleren indien deze view gewijzigd wordt. Indien er rijen worden toegevoegd of kolommen worden aangepast, zal er gekeken worden of de view nog altijd aan de constraints voldoet. Indien dit niet het geval is wordt de operatie in het geheel afgebroken.