

Samenvatting Databases

Robin Vanhove

Juni 2017

Inhoudsopgave

	2
I Conceptueel en Relationeel Model & Query's	3
1 ER & EER	3
2 Relationele algebra	3
2.1 Operatoren	3
2.1.1 Selectie	3
2.1.2 Projectie	3
2.1.3 Hernoeming	4
2.1.4 Unie doorsnede en verschil	4
2.1.5 Cartesisch product	4
2.1.6 Join Operator	4
2.1.7 Deling	4
2.2 Aggregaatfuncties	4
3 SQL	5
3.1 Operatoren	5
3.2 Aggregaatfuncties	5
3.2.1 Aanpassingen	5
3.3 Views	5
3.4 Geneste Queries	5
3.5 Transactie	5
3.6 Permissies	6
3.7 Restricties	6
3.8 Triggers	6
4 Relationele Calculus	6
4.1 Queries	6
5 Programma's verbinden met een Database	7
6 Ontwerp van een database	7
6.1 Informele richtlijnen	7
6.2 Functionele Afhankelijkheden	7
6.2.1 Afleidingsregels	7
6.3 Normalisatie	8
6.3.1 Eerste Normaalvorm	8
6.3.2 Tweede Normaalvorm	8
6.3.3 Derde Normaalvorm	9
6.3.4 Boyce-Codd Normaalvorm	9
6.3.5 Vierde Normaalvorm	9

6.3.6 Vijfde Normaalvorm	10
II Het Fysiek Model	10
7 Indexeren	10
7.1 Zoeken in bestanden	10
7.1.1 Zoeken in ongeordende bestanden	10
7.1.2 Zoeken in geordende bestanden	10
7.2 Indexen: Definitie en soorten	10
7.2.1 Primaire indexen	11
7.2.2 Cluster indexen	11
7.2.3 Secundaire indexen	11
7.2.4 Hash indexen	11
7.3 Indexen in MySQL	12
8 Queryverwerking en Optimalisatie	12
8.1 Transformatie regels	12
8.2 Heuristische optimalisatie	13
8.3 Uitvoeringsplan	13
8.4 Extern Sorteren	13
9 Transacties	13
10 Concurrentiecontrole	13
11 Herstel	13

Beknopte samenvatting voor het OPO Gegevensbanken. Voornamelijk gebaseerd op de slides, maar ook het boek.

Versie 0.0

Gecompileerd op 22 juni 2017

This work is licensed under a Creative Commons “Attribution-NonCommercial-ShareAlike 4.0 International” license.



Deel I

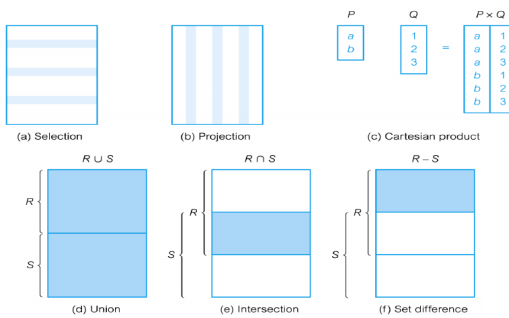
Conceptueel en Relatieveel Model & Query's

1 ER & EER

Niet te kennen voor examen

2 Relatievele algebra

2.1 Operatoren



Naam	Teken
Selectie	σ
Projectie	π
Hernoeming	ρ, \leftarrow
Unie	\cup
Doorsnede	\cap
Verschil	$-$
Caresisch product	\bowtie
Join Operator	$*$
Deling	\div

Fundamentele operatoren $\{\sigma, \pi, \cup, -, \times\}$, zijn de enige nodige operatoren. De andere kunnen er op gebaseerd worden.

2.1.1 Selectie

$$\sigma_{\text{selectiecriterium}}(R)$$

Selecteerd een aantal tupels uit een rij R met het criterium. Het resultaat is een nieuwe relatie (tabel) met hetzelfde schema.

bv.

- $\sigma_{ID=1}(USERS)$
- $\sigma_{color='red' \vee color='green'}(BOATS)$
- $\sigma_{AGE < 50}(USERS)$

Selectie is cumulatief, dus $\sigma_a(\sigma_b(T)) = \sigma_{a \wedge b}(T)$

2.1.2 Projectie

$$\pi_{\text{attributenlijst}}(R)$$

Een aantal kolommen uit een tabel halen.

bv.

- $\pi_{\text{first_name}, \text{last_name}}(USERS)$
- $\pi_{\text{color}}(\sigma_{ID=1}(BOATS))$

2.1.3 Hernoeming

$$RESULT \leftarrow \sigma_{Dno=1}(EMPLOYEE)$$
$$\rho_{RESULT}(\sigma_{Dno=1}(EMPLOYEE))$$

2.1.4 Unie doorsnede en verschil

Unie	\cup
Doorsnede	\cap
Verschil	$-$

Enkel op vergelijkbare relaties.

2.1.5 Cartesisch product

$$Q = R \times S$$

Geeft als resultaat een nieuwe relatie die elke mogelijke combinatie van de twee tupels bevat.

2.1.6 Join Operator

$$R \bowtie_F S$$

Is hetzelfde als een cartesisch product gevolgd door een selectie.

Er zijn meerdere soorten joins

- **Theta join** een join waarbij de voorwaarde in de vorm is van $A\theta B$
 - Met $\theta = \{=, <, >, \leq, \geq, \neq\}$
- **Equi-join**, $R \bowtie_{a=b} S$
- **Natuurlijke join**, $R * S$. Een join waarbij de sleutel gebruikt wordt op een conditie.

Een uitwendige join (links, rechts of volledig) is een speciale join die sowieso ieder element uit de (linker, rechter of beide) relatie, met en null als er geen paar gevonden is. Het teken hiervoor is een strikje met twee lijntjes in de richting van de join.

2.1.7 Deling

$$T = R \div S$$

Tegengestelde van het cartesisch product.

Bijvoorbeeld, voor welke zeilers bestaan reserveringen voor alle boten in een verzameling

2.2 Aggregaatfuncties

$$groepering \mathfrak{S}_{functions}(R)$$

Functies die op een verzameling waarden uitgevoerd worden. SUM, AVERAGE, MAX, MIN, COUNT.

- Groepering is de verzameling van attributen waarop de groepering gebeurt
- Functies is de lijst van koppels (functie, attributt)

bv. $Dno \mathfrak{S}_{AVERAGESalary}(EMPLOYEE)$

3 SQL

Structured Query Language

3.1 Operatoren

Formularium van SQL beschikbaar op examen.

3.2 Aggregaatfuncties

AVG, SUM, MIN, MAX, COUNT

Eerst groeperen met GROUP BY

3.2.1 Aanpassingen

INSERT, UPDATE, DELETE

3.3 Views

Een **view** is een afgeleide relatie, dit wil zeggen dat de tupels niet expliciet worden opgeslagen. Implementatie op twee manieren

- **Query modification**, de query wordt aangepast voor hij wordt uitgevoerd op de onderliggende tabellen.
- **View materialization**, de afgeleide tabel (view) wordt aangemaakt en daarna wordt de query er op uitgevoerd.

Aanpassingen zijn mogelijk als de view maar uit 1 tabel (basisrelatie) bestaat en een pk bevat.

3.4 Geneste Queries

Query in en Query.

IN, ALL, ANY, EXISTS

```
SELECT xxxx
FROM xxxx
WHERE xxxx IN (
    SELECT yyyy
    FROM yyyy
    WHERE yyyy
);
```

3.5 Transactie

Een atomaire eenheid. Standaard is iedere query een transactie. Een transactie wordt als permanente aanpassing gezien.

START ... COMMIT

ROLLBACK kan gebruikt worden om een huidige transactie ongedaan te maken.

```
START;
UPDATE xxx SET xxxx WHERE xxxx;
UPDATE xxx SET yyyy WHERE yyyy;
COMMIT;
```

3.6 Permissies

Verschillende database gebruikers kunnen ander rechten hebben.

```
GRANT right ON table TO user;
REVOKE right ON table TO user;
```

3.7 Restricties

Een attribuut kan een primaire sleutel zijn (id). Gewoon uniek. Of een sleutel die naar een andere tabel wijst.

```
PRIMARY KEY <attr>
UNIQUE <attr>
```

```
FOREIGN KEY <attr> REFERENCES <table><attr>
```

Een attribuut kan een standaard waarde hebben. Zou niet 'null' mogen zijn. Of kan aan andere voorwaarde onderhevig worden.

```
NOT NULL <attr>
DEFAULT <value>
CHECK <condition>
```

Algemene beperking opleggen met ASSERTION

```
CREATE ASSERTION <name> CHECK <cond>
```

3.8 Triggers

Event - voorwaarde - actie

```
CREATE TIGGER <name>
{BEFORE | AFTER} <event> ON <table>
FOR EACH ROW
WHEN <cond>
    <action>
```

4 Relationele Calculus

Relationele Algebra: **Hoe** Relationele Calculus: **WAT**

- Tupelcalculus
- Domeincalculus

Gebruik van predikatenlogica.

4.1 Queries

Een query is in de vorm van $\{t \mid \text{formule}(t)\}$ of $\{t.A, t.B, \dots T.z \mid \text{formule}(t)\}$

```
{ t.Bdate, t.Address | EMPLOYEE(t) and t.Fname = 'John' and t.Lname = 'Smith' }
{t | BOATS(t) and (t.color='red' or t.color='green')}
```

De kwantoren \exists en \forall zijn ook mogelijk.

5 Programma's verbinden met een Database

Niet te kennen voor examen

6 Ontwerp van een database

1. Hoogniveau modellering (top-down)
 - (E)ER schema
2. Meteen een relationeel gegevensschema (bottom-up)

Informatie bewaren, minimale redundantie (\rightarrow maximale performantie)

\rightarrow **normaliseren**

6.1 Informele richtlijnen

1. De betekenis van een relatie moet gemakkelijk verklaard kunnen worden.
 - Betekenis van een relatie moet duidelijk zijn.
 - Betekenis van een attribuut moet duidelijk zijn.
2. Redundantie en anomalieën vermijden.
3. Vermijd de waarde 'null'.
 - Niet van toepassing, ongekend.
4. Vermijd dat na equi-joins onechte tupels kunnen ontstaan. Vermijd attributen met dezelfde naam in verschillende relaties (als ze geen foreign keys zijn).

6.2 Functionele Afhankelijkheden

Een **functionele afhankelijkheid** ($X \rightarrow Y$) tussen twee (verzamelingen van) attributen X en Y is een beperking op de mogelijke tupels die gevormd kunnen worden. Voor twee tupels t_1 en t_2 is het namelijk zo dat als $t_1[x] = t_2[x]$ geldt dan ook $t_1[y] = t_2[y]$ geldt.

In andere woorden de waarden van de Y component van de tuple wordt bepaald door de waarde van X. Of de waarde van de X component bepalen uniek (functioneel) de waarde van de Y component.

bv.

- $Ssn \rightarrow Ename$
- $Pnumber \rightarrow \{Pname, Plocation\}$
- $\{Ssn, Pnumber\} \rightarrow Hours$

6.2.1 Afleidingsregels

1. Reflexiviteitsregel $Y \subseteq X \Rightarrow X \rightarrow Y$
2. Uitbreidingsregel $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
3. Transitiviteitsregel $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
4. Decompositieregel $\{X \rightarrow YZ\} \models X \rightarrow Y$
5. Verenigingsregel $\{X \rightarrow Y, Y \rightarrow Z\} \models \{X \rightarrow YZ\}$
6. Pseudo-transitiviteitsregel $\{X \rightarrow Y, WY \rightarrow Z\} \models \{WX \rightarrow Z\}$

F^+ is de verzameling die alle afleidingen die uit F volgen bevat.

bv. $F = \{SSN \rightarrow ENAME, PNUMBER \rightarrow \{PNAME, PLOCATION\}, \{SSN, PNUMBER\} \rightarrow HOURS\}$

- $\{SSN\}^+ = \{SSN, ENAME\}$
- $\{PNUMBER\}^+ = \{PNUMBER, PNAME, PLOCATION\}$
- $\{SSN, PNUMBER\}^+ = \{SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS\}$

6.3 Normalisatie

Een **normaalkorm** legt bepaalde eisen op aan een relatie. Normalisatie is een relatie in een bepaalde normaalkorm brengen.

Elke volgende normaalkorm is een speciaal geval van de vorige.

Een **super sleutel** of super key is een aantal attributen van een tupel die uniek zijn voor de tupel.

Een **sleutel** is een supersleutel waarvan geen attribuut verwijderd kan worden. Dus een sleutel is minimaal. Alle mogelijke sleutels zijn **kanidaat sleutels** maar er is maar 1 **primaire sleutel**.

6.3.1 Eerste Normaalkorm

Een relatieschema is in de **eerste normaalkorm** als het domein van elk attribuut is enkelvoudig (atomair).

Zo kan een attribuut geen lijst zijn. Maar moet er voor ieder element van de lijst een nieuwe tupel zijn.

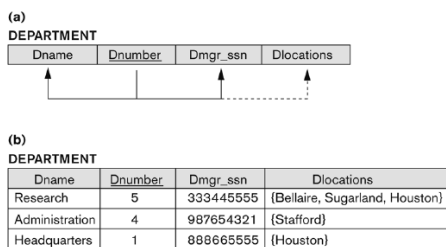


Figure 10.8
Normalization into 1NF.
(a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

(c)

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

6.3.2 Tweede Normaalkorm

We spreken over een **partiele functionele afhankelijkheid** of een **partial dependency** als voor een afhankelijkheid $X \rightarrow Y$ geldt dat we een X door Z kunnen vervangen waarbij $Z \subset X$ en de afhankelijkheid $Z \rightarrow Y$ nog altijd geldig is. Dus als we een attribuut kunnen weglaten is het partieel functioneel afhankelijk.

Anders spreken we over een **volledige functionele afhankelijkheid**.

Een relatieschema is in de **tweede normaalkorm** als ieder niet-sleutel attribuut volledig functioneel afhankelijk is.

In andere woorden voor elk niet-sleutel-attribuut moet de hele primaire sleutel nodig zijn om het te determineren.

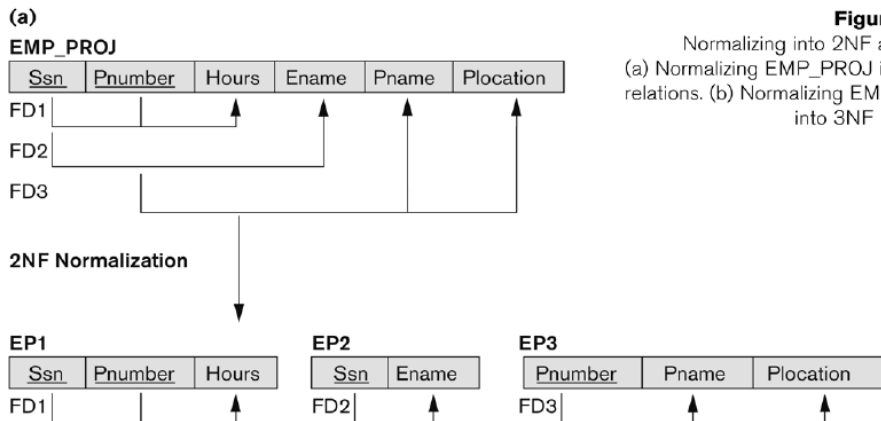


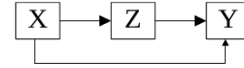
Figure 10.10
Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

6.3.3 Derde Normaalvorm

Y is **triviaal functioneel afhankelijk** van X asa $Y \subseteq X$.

Een functionele afhankelijkheid $X \rightarrow Y$ is een **transitieve functionele afhankelijkheid** asa er geen Z bestaat waarbij

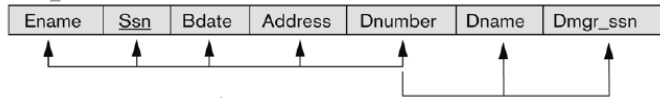
1. Z is volledig en niet-triviaal functioneel afhankelijk van X .
2. Z is geen (echte of onechte) deelverzameling van een kandidaatsleutel.
3. Y is niet-triviaal functioneel afhankelijk van Z .



Een 2NF-relatieschema is in de **derde normaalvorm** asa voor geen enkel niet-sleutelattribuut A geldt dat A *transitief* functioneel afhankelijk is van een kandidaatsleutel. Dus alles is *direct* afhankelijk van een sleutel.

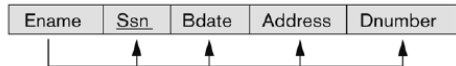
(b)

EMP_DEPT

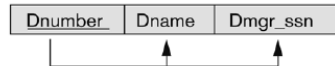


3NF Normalization

ED1



ED2



6.3.4 Boyce-Codd Normaalvorm

De **boyce-Codd Normaalvorm** (BCNF) geldt als er voor iedere niet-triviale functionele afhankelijkheid $X \rightarrow Y$ geldt dat X een supersleutel is.

De BCNF haalt alle ongewenste functionele afhankelijkheden weg. Maar dit is niet steeds bereikbaar zonder andere problemen te creëren.

6.3.5 Vierde Normaalvorm

Een **meerwaardige afhankelijkheid** genoteerd met

$X \twoheadrightarrow Y$ als er vier tupels zijn zodat

- $t_1[X] = t_2[X] = t_3[X] = t_4[X]$
- $t_3[Y] = t_1[Y]$ en $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$ en $t_4[Z] = t_1[Z]$

Tupel	X	Y	X
t_1	a	b_1	c_1
t_2	a	b_1	c_2
t_3	a	b_2	c_1
t_4	a	b_2	c_2

bv. T-shirts hebben een *model*, *maat* en *kleur*. $\text{Model} \twoheadrightarrow \text{maat}$ betekent dat:

- Voor elk *model* wordt in welbepaalde *maten* geleverd, onafhankelijk van de *kleur*.
- Dus elke combinatie van *kleur* en *maat* is mogelijk.

Een **meerwaardige afhankelijkheid** $X \twoheadrightarrow Y$ is **triviaal** asa $Y \subseteq X$.

Een relatieschema is in de **vierde normaalvorm** asa voer iedere niet-triviale meerwaardige afhankelijkheid van de vorm $X \twoheadrightarrow Y$ van F^+ geldt dat X een supersleutel is.

6.3.6 Vijfde Normaalvorm

Een **join-afhankelijkheid** $JD(U_1, \dots, U_n)$ in een relatie is een restrictie die aangeeft dat voor elke r van R geldt dat er een verliesloze decompositie in relaties $R_1, \dots, R_n (n > 2)$ is met attributen U_1, \dots, U_n .

Model	Maat	Kleur	
x	y	z	} als dergelijke tupels voorkomen in een extensie van R...
x	y	z	
x	y	z	dan ook dit tuplel

Dit is de join-afhankelijkheid
 $JD (\{Model, Maat\} , \{Maat, Kleur\} , \{Model, Kleur\})$

Een relatieschema is in de **vijfde normaalvorm** asa voor elke niet-triviale join-afhankelijkheid $JD(U_1, \dots, U_n)$ van F^+ geldt dat U_1, \dots, U_n supersleutels zijn.

Deel II

Het Fysiek Model

7 Indexeren

Je leest of schrijft nooit 1 record van de opslaghardware, altijd in blokken.

Een blok die in het geheugen gelden wordt kan snel doorzocht worden.

7.1 Zoeken in bestanden

7.1.1 Zoeken in ongeordende bestanden

In **ongeordende bestanden** moet er **lineair** gezocht worden. Gemiddeld wordt de helft van de records bekeken.

7.1.2 Zoeken in geordende bestanden

Als de bestanden **geordend** zijn volgens het attribuut waarop we zoeken kunnen we **binair zoeken**. Dit kan gemiddeld in $\log_2(N)$ tijd.

7.2 Indexen: Definitie en soorten

Een **index** is een lijst van onderwerpen met wijzers naar het bestand. Of in andere woorden, een index is een datastructuur die toegang tot een bestand via een veld efficient maakt.

Een index is een bestand zoals de data een bestand is.

Er zijn drie soorten indexen, de **primaire index** is de index op het veld dat de ordening van de bestanden bepaald en die ieder bestand uniek identificeerd (pk). De **clusterindex** is een index op het veld dat de ordening bepaalt maar niet noodzakelijk uniek is. En een **secundaire index** is een index op een ander veld dan dat wat der ordening bepaalt (ook niet uniek).

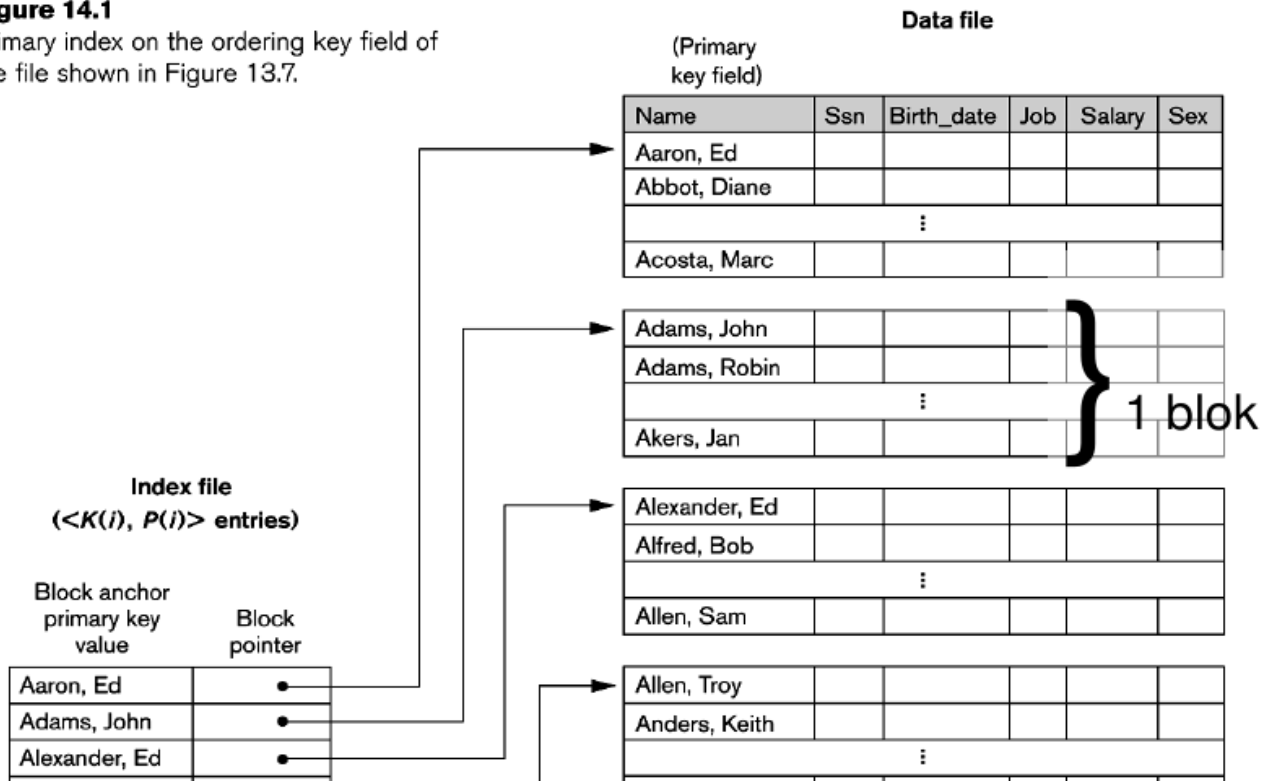
Een index is gemakkelijk omdat het veel kleiner is om in het geheugen te laden, het moet namelijk enkel enkele records en pointers bijhouden.

7.2.1 Primaire indexen

De primaire index bevat een fysisch geordende lijst volgens de sleutel. De index bevat 1 record per blok. Dit is het *ankerrecord* en is meestal het eerste of het laatste record van het blok.

Figure 14.1

Primary index on the ordering key field of the file shown in Figure 13.7.



7.2.2 Cluster indexen

De index is fysisch geordend volgens het veld dat niet uniek is. De bestanden zijn wel geordend volgens dit veld. De index bevat per waarde van het veld 1 wijzer naar het blok waar de eerste record met die waarde in voorkomt.

7.2.3 Secundaire indexen

Een secundaire index is een index op een veld dat niet de ordening bepaalt. De index zelf wordt wel geordend volgens dat veld.

Als dit veld een secundair sleutel veld is kan er per waarde van de sleutel (en dus per record) een wijzer in de index staan.

We hebben dichte index die alle records bevat. Dit is nog steeds nuttig omdat het kleiner is dan de data zelf. (minder kolommen)

7.2.4 Hash indexen

Een hashing functie wordt gebruikt om de key meteen om te zetten naar een adres.

Collisions, botsingen mogelijk.

Linear probing: open adressering of Separate chaining: Gesloten adressering.

7.3 Indexen in MySQL

Clustered. reorders the way records in the table are physically sorted.

Nonclustered index. Order in index is different than physically.

8 Queryverwerking en Optimalisatie

SQL Query wordt omgezet naar relationele algebra. Er wordt een boom opgesteld voor de query deze wordt geoptimaliseerd.

Eerst wordt de queryboom opgebouwd in canonieke vorm (dus letterlijk hoe de query is). Vervolgens wordt de boom geherstructureerd zonder equivalentie te verliezen.

8.1 Transformatie regels

- σ cascade
 - Selectie op conjuncties van condities omzetten in opeenvolgende eenvoudige selecties.
 - $\sigma_{c_1 \text{ AND } c_2 \text{ AND } \dots \text{ AND } c_n}(R) = \sigma_{c_1}(\dots(\sigma_{c_n}(R)))$
- σ is commutatief
 - $\sigma_a(\sigma_b(R)) = \sigma_b(\sigma_a(R))$
- π cascade
 - Enkel laatste projectie overhouden
 - $\pi_a(\pi_b(R)) = \pi_a(R)$
- Ommissen van σ en π
 - Enkel als de voorwaarde c toepasbaar is op de attributen
 - $\pi_A(\sigma_c(R)) = \sigma_c(\pi_A(R))$
- Commutativiteit van \bowtie en \times
 - $R \times S \equiv S \times R$
 - $R \bowtie S \equiv S \bowtie R$
- Omwisselen van σ en \bowtie of \times
 - Als de voorwaarde c toepasbaar is op de attributen van R
 - $\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S$
 - Als de voorwaarde c toepasbaar is op de attributen van R en S
 - $\sigma_c(R \bowtie S) \equiv \sigma_{c_1}(R) \bowtie \sigma_{c_2}(S)$
- Omwisselen van π en \times
 - $\pi_L(R \times S) \equiv \pi_{L(R)}(R) \times \pi_{L(S)}(S)$
- Omwisselen van π en \bowtie
 - Als de join conditie allen attributen in L gebruikt
 - $\pi_L(R \bowtie_c S) \equiv \pi_{L(R)}(R) \bowtie_c \pi_{L(S)}(S)$
 - Anders R en S projecteren op join attributen + attributen projectie-lijst daarna joinen en op het einde projecteren op L .
- \cup en \cap zijn commutatief
- \bowtie , \times , \cup en \cap zijn associatief
- Commutativiteit van σ met verzamelings-operaties
 - $\sigma_c(R \cap S) \equiv \sigma_c(R) \cap \sigma_c(S)$
 - $\sigma_c(R \cup S) \equiv \sigma_c(R) \cup \sigma_c(S)$
 - $\sigma_c(R/S) \equiv \sigma_c(R)/\sigma_c(S)$
- Commutativiteit van π met verzamelings operaties.
 - $\pi_L(R \cup S) \equiv \pi_L(R) \cup \pi_L(S)$
- Samenvatten van $\sigma(\times)$ in \bowtie
 - $\sigma_c(R \times S) \equiv R \bowtie_c S$
- ...

8.2 Heuristische optimalisatie

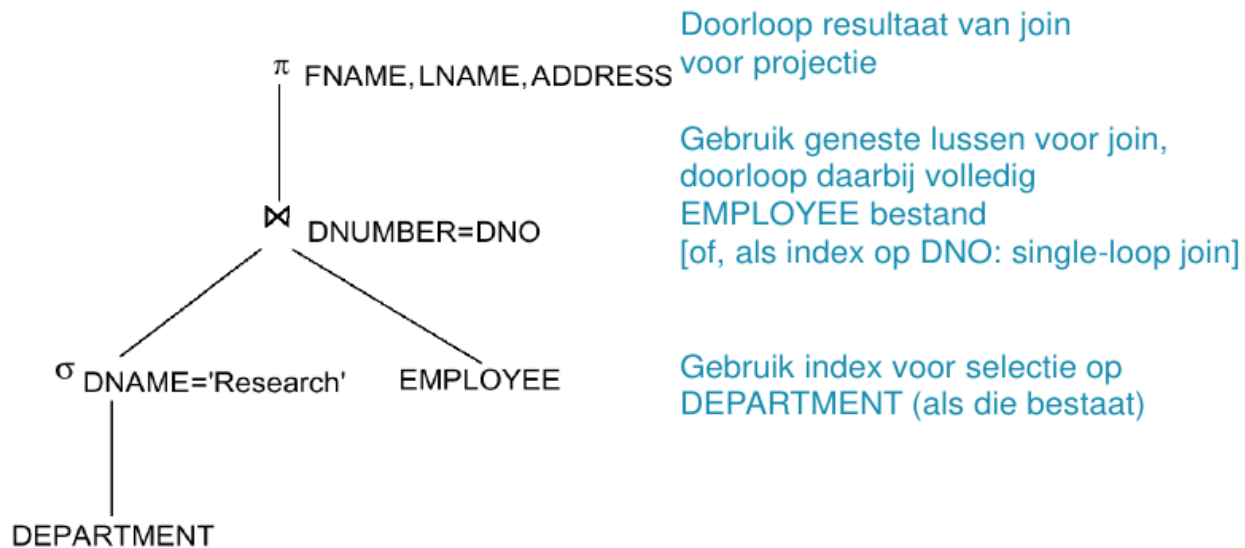
1. Splits conjunctie van selecties
2. Schuif selecties zo ver mogelijk naar beneden
 - selectie over 1 relatie: net boven de relatie
 - selectie over 2 relaties: zo dicht mogelijk boven hun cartesisch product
3. Schuif kleine relaties zo ver mogelijk naar links
 - maar houd join condities bij cartesische producten
4. Combineer cartesisch product gevold door selectie met join conditie tot join
5. Splits projecties op, en projecteer zo vroeg mogelijk
 - houd alleen attributen die verder boven nodig zijn
6. Identificeer deelbomen die door één algoritme kunnen uitgevoerd worden (zonder tijdelijke bestanden)

Kleine relaties zijn letterlijk hoeveel tupels er in de relatie zijn. Dus extra informatie, de grote van de tupels en het aantal waardes bijhouden.

8.3 Uitvoeringsplan

Nadat de boom is geoptimaliseerd weten we de volgorde van de operaties en hebben we een indicatie van deelbomen waar mogelijk een algoritme voor bestaat.

Maar nog geen exacte implementatie dus welke indexen, en welk soort evaluatie.



8.4 Extern Sorteren

Vaak niet genoeg geheugen voor intern te sorteren bv. quicksort. Dus extern sorteren (merge sort). Dan kunnen aparte blokken om de beurt in het geheugen geladen worden.

9 Transacties

10 Concurrentiecontrole

11 Herstel