

# Indexen



**Cursus  
Boek**

- cursus 'Databanken 1' blz. 53-58
- Deze slides
- Aanvullend: Oracle Database 11g: SQL Fundamentals I Exam Guide  
§12.5 Blz. 509-518

# Agenda



- Wat is een index?
- Hoe werkt een index?
- Syntax
- Waarom indexen?
- Op welk moment een index creëren?
- Verwijderen van een index
- Dictionary tabellen

# Wat is een index?

Je kan een index vergelijken met de trefwoordenlijst van een boek.

## Trefwoordenlijst A tot Z

### A

Academie voor muziek en woord .....	14
Academie voor schone kunsten (SASK) .....	14
Adres .....	14
Adviesraden .....	14
Afval .....	15
Afvalwater .....	15
Arbeidskaarten .....	15
Attesten .....	15

### B

Begraafplaatsen .....	15
Beheersorgaan gemeenschapscentrum .....	16
Beheersorgaan openbare bibliotheek .....	16
Belgische nationaliteit .....	16
Bevolkingsdienst .....	16
Bibliotheek .....	16
Bouwen en verbouwen .....	17
Brandweer .....	17
Budgetbegeleiding/budgetbeheer .....	17
Buitenschoolse kinderopvang .....	17
Burgerlijke stand & sociale zaken .....	18

### C

Composteren .....	18
Communicatiedienst .....	18
Conformiteitsattest .....	18
Containerpark .....	18
Crisisopvang .....	18
Cultuurdienst .....	18
Cultuurraad .....	19

### D

Dagopvang .....	19
Dagrestaurant .....	19

Dorpshuizen .....	19
Drankvergunning .....	21
Drugpunt Waas - Lokaal Drugoverleg .....	21

### E

Echtscheiding .....	21
E-loket .....	21
Energiescan .....	21
Erediensten .....	21
Erkenning .....	22

### F

Feestmateriaal .....	22
Feesten / evenementen .....	22
Financieel beheerder .....	23
Financiële dienst .....	23
Financiële steun .....	23

### G

Geboorte .....	23
Geluidsnorm afwijking .....	24
Gemeenschapswacht .....	24
Gemeenteberichten .....	24
Getijdenmolen .....	24
Gezinsraad .....	24
GIS (geografisch informatiesysteem) .....	24
GIS-dienst .....	25
Grabbelpas .....	25
GROS (gemeentelijke raad voor ontwikkelings- samenwerking) .....	25

### H

Huisbezoek .....	25
Huisnummers .....	25

# Wat is een index?

- Je moet bv. veel zoeken op het salaris van Medewerkers.
- We plaatsen op salaris uit de tabel MEDEWERKERS een

index: **CREATE INDEX** ind\_med\_sal **ON**  
medewerkers (salaris) ;

<i>salaris</i>	<i>rowid's</i>
25000	--- --- --- (3 personen met salaris 25000)
30000	---
38000	---
43000	--- ---

Bij elke mogelijke attribuutwaarde van het geïndexeerde attribuut staan de rowid's van de rijen die die attribuutwaarde bevatten.

Een ROWID bevat de fysische locatie van een rij in de database- files achter de database. (geeft weer in welke file, welk data block en binnen het data block op welke locatie de rij zich bevindt)

# Hoe werkt een index?

Bij uitvoering van een query wordt de index enkel gebruikt wanneer het geïndexeerd attribuut in de WHERE clausule staat.

Bovendien mag het geïndexeerd attribuut niet onderworpen zijn aan een functie!!

Stel dat er een index gedefinieerd is voor salaris:

```
SELECT *  
FROM medewerkers  
WHERE salaris=25000;
```

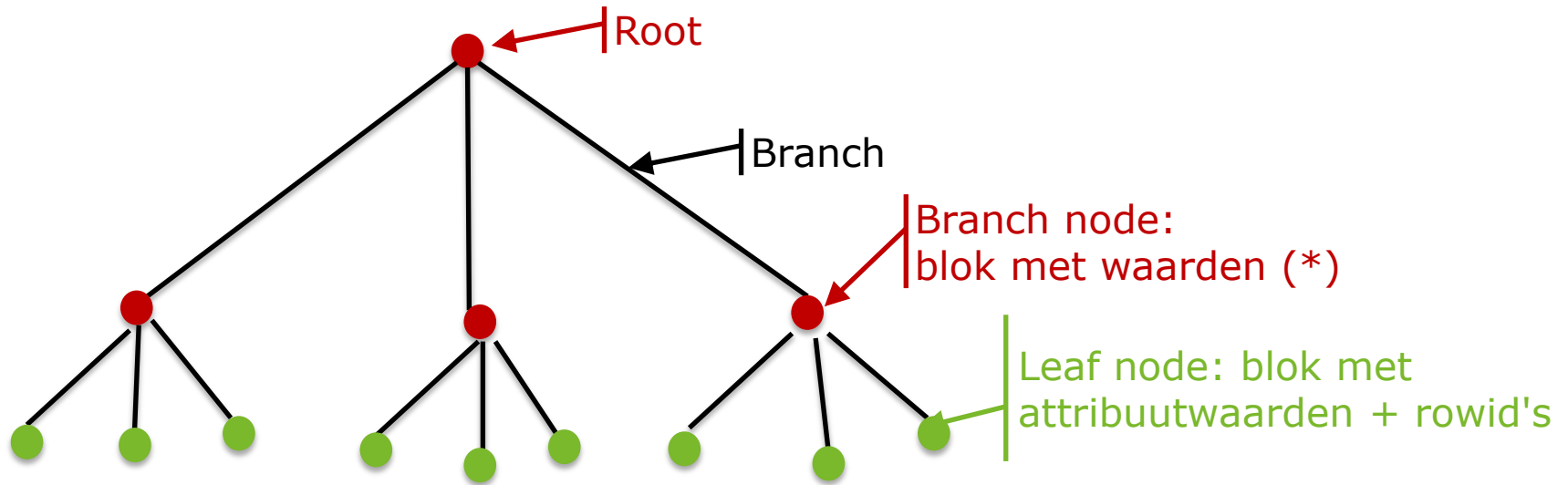
1. Er wordt in de indextabel gericht gezocht naar de waarde 25000
  - De overeenkomstige rowid's worden uit de index tabel opgehaald
2. In MEDEWERKERS worden de rijen met de gevonden rowid's opgehaald.

# Hoe werkt een index: B-Tree

In ORACLE wordt standaard een B-tree index gebruikt.

- boomstructuur met knooppunten (branch node), vertakkingen (branches) en eindpunten (leaf node)
  - B-tree staat voor Balanced-tree
- de boomstructuur wordt zo opgezet dat het aantal vertakkingen per knooppunt = het aantal knooppunten dat doorlopen moet worden om de waarde te vinden.

# Hoe werkt een index: B-Tree

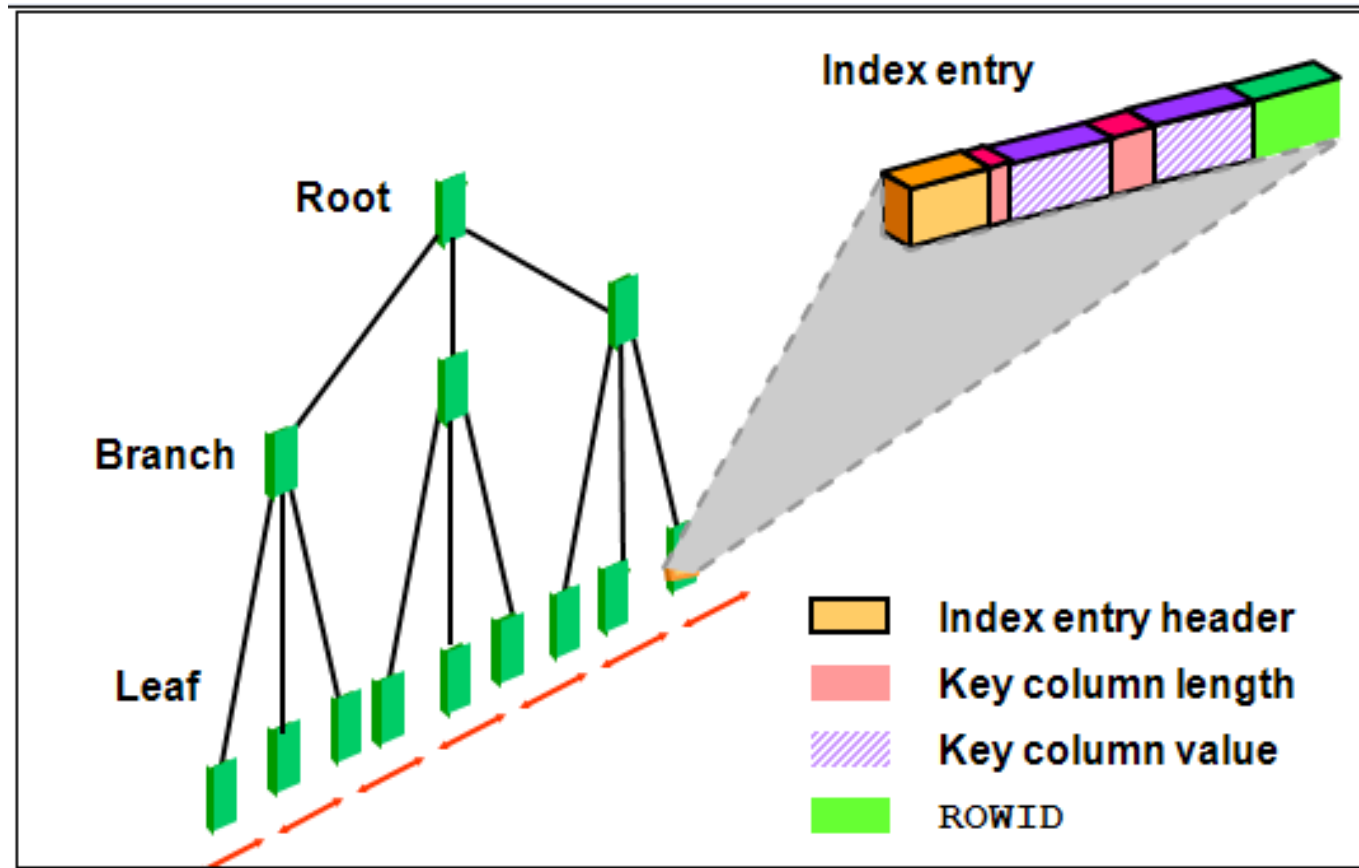


(\*) op basis van de waarden in een branch node kan beslist worden welke vertakking moet genomen worden.

Het aantal vertakkingen (branches) per branch node (hier 3) is gelijk aan het aantal nodes dat moet doorlopen worden om de attribuutwaarde + rowid in een leaf node te vinden.



# Hoe werkt een index?



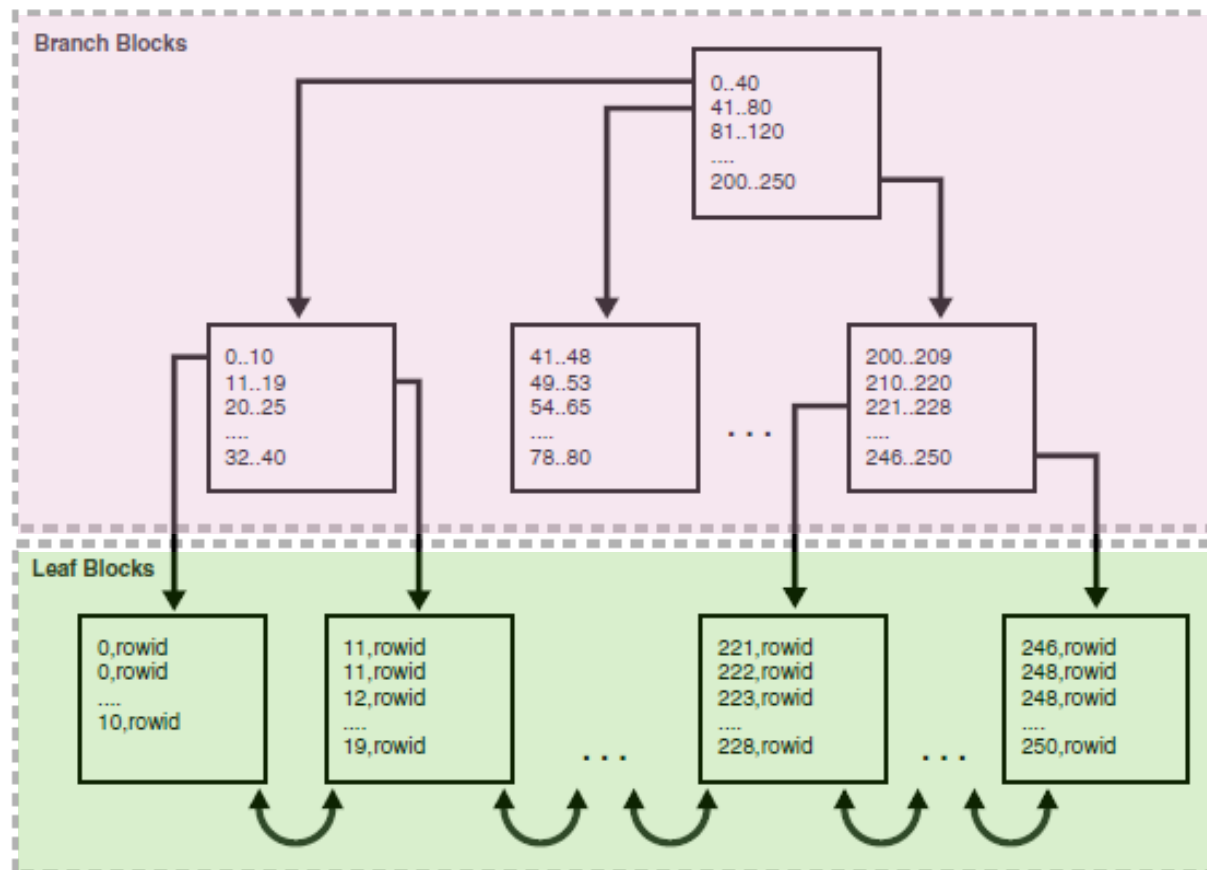
# Hoe werkt een index?



Voorbeeld

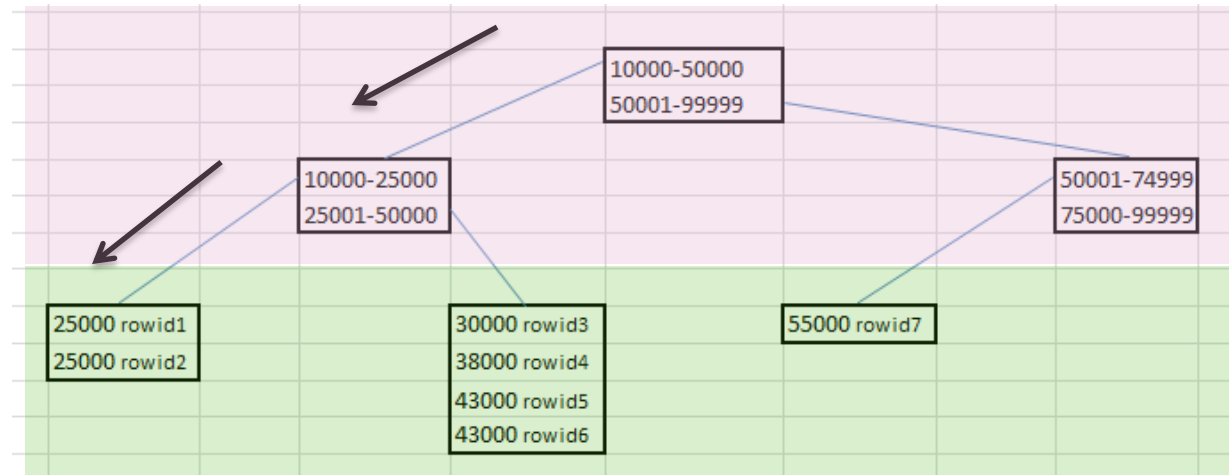
Stel dat een geïndexeerd attribuut momenteel waarden tussen 1 en 250 zou bevatten, dan wordt er als volgt gezocht:

Figure 3-1 Internal Structure of a B-tree Index



# Hoe werkt een index?

de B tree index op salaris zou er als volgt kunnen uitzien:



```
SELECT *  
FROM medewerkers  
WHERE salaris=25000;
```

# Hoe werkt een index?

- een **branch blok (node)** bevat:
  - een minimum key prefix om te kunnen kiezen tussen 2 key waarden
  - een pointer naar het child blok dat die key bevat
- een **leaf block (node)** bevat:
  - de betreffende attribuutwaarde + rowid

Alle paren van attribuutwaarden en rowid's zijn gesorteerd op attribuutwaarden en binnen attribuutwaarde op rowid.

# Hoe werkt een index?

- **Rijen** van een tabel zitten in datafiles
- In die datafiles zitten die rijen **in data blokken**.
- Oracle kan enkel data blokken lezen/schrijven (dus geen rijen)
- Als men geen index gebruikt moet men alle data blokken, die rijen uit de tabel bevatten, doorlopen (=Full Table Scan) en er de rijen die aan de WHERE conditie voldoen uithalen
- Als men een index gebruikt wordt gezocht via de B-tree index.

# Index aanmaken

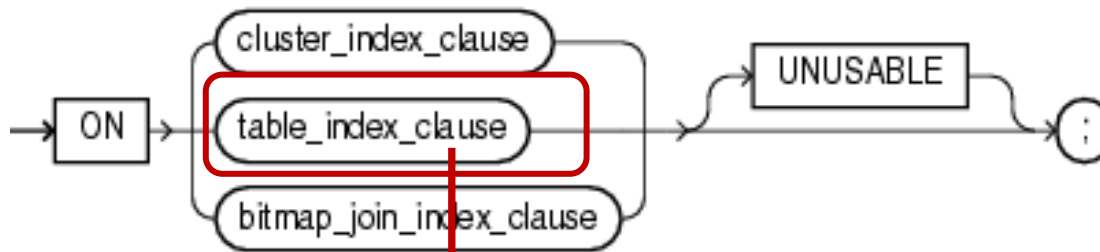
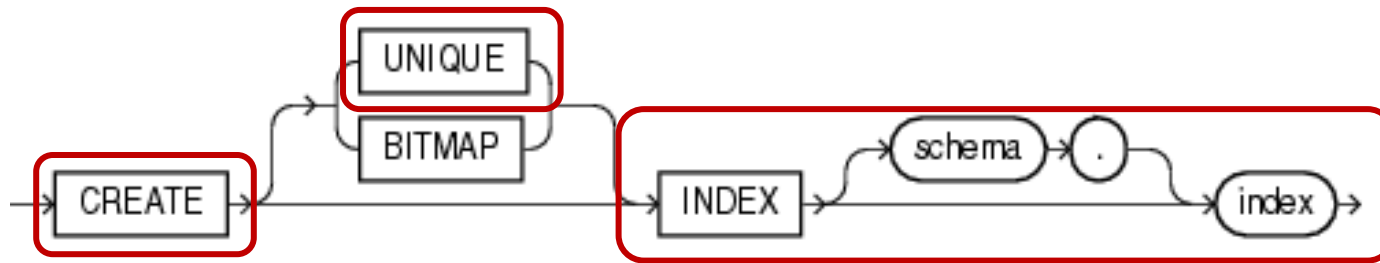


Definitie

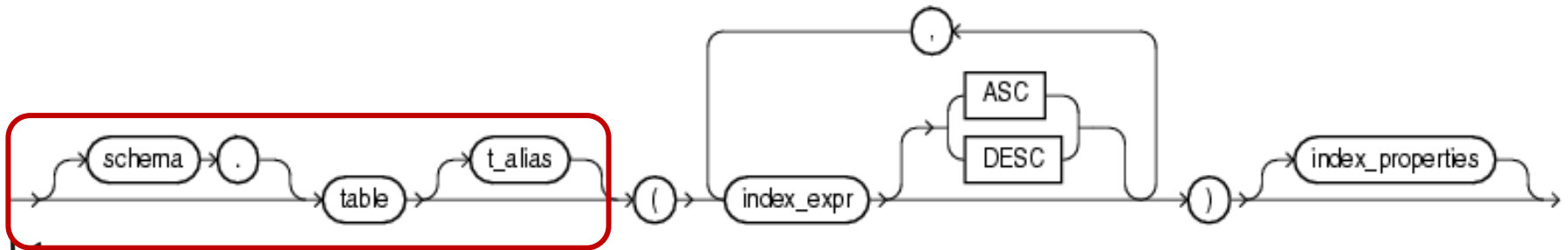
**SQL**

Reference

**p. 89**



**table\_index\_clause**



# Index aanmaken

- Enkelvoudige/samengestelde index: één / meer *attributen* per index entry
  - Op attributen waarop in de WHERE clause vaak samen getest wordt kan een samengestelde index gedefinieerd worden
- Unieke/gewone index: één / meer *rowid's* per index entry
  - Als je een primaire sleutel of UNIQUE constraint maakt, wordt automatisch een unieke index aangemaakt.
  - Is de primaire sleutel of constraint samengesteld, dan krijg je een samengestelde unieke index



- Gewone index

```
CREATE INDEX ind_med_afd_nr ON medewerkers (afd_nr) ;
```

- Unieke index

```
CREATE UNIQUE INDEX ind_afd_naam ON afdelingen (afd_naam) ;
```

- Samengestelde gewone index (=composite key=concatenated key)

```
CREATE INDEX ind_med_afd_sal  
ON medewerkers (salaris, afd_nr) ;
```

- Samengestelde unieke index

```
CREATE TABLE opdrachten (sofi_nr...,  
CONSTRAINT pk_opdrachten PRIMARY KEY (sofi_nr, proj_nr) ;
```



# Index aanmaken



```
CREATE INDEX ind_med_afd_sal  
ON medewerkers (salaris, afd_nr) ;
```

The screenshot shows the Oracle SQL Developer interface with the 'MEDEWERKERS' table selected. The 'Indexes' tab is active, displaying a list of indexes. The index 'IND\_MED\_AFD\_SAL' is highlighted with a blue arrow.

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	THEORIE	IND_MED_SAL	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	SALARIS
2	THEORIE	IND_MED_AFD_NR	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	AFD_NR
3	THEORIE	PK_MEDEWERKERS	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	SOFI_NR
4	THEORIE	IND_MED_AFD_SAL	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	SALARIS, AFD_NR
5	THEORIE	UN_PARKEERPLAATS	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	PARKEERPLAATS

# Zoeken op samengestelde index: **combinatie**

**INDEX op salaris + afd\_nr**

**salaris afd\_nr rowid**

25000	3	AAAHUCAAEAAAC18AAD
25000	3	AAAHUCAAEAAAC18AAE
25000	7	AAAHUCAAEAAAC18AAH
30000	7	AAAHUCAAEAAAC18AAF
38000	7	AAAHUCAAEAAAC18AAG
43000	3	AAAHUCAAEAAAC18AAB
43000	7	AAAHUCAAEAAAC18AAC
55000	1	AAAHUCAAEAAAC18AAA

```
SELECT * FROM medewerkers  
WHERE salaris=43000 AND afd_nr=3;
```

● Hoe wordt gezocht? **Range scanning**

- De optimizer zal in de index gericht zoeken naar de **combinatie salariswaarde 43000 en afd 3** en de bijbehorende rowid(s) ophalen.

In de indextabel zitten de  
attribuutwaarden (automatisch)  
gesorteerd op salaris en binnen  
salaris op afdeling

# Zoeken op samengestelde indexen: combinatie

Het DBMS beschikt over een optimizer. Deze bepaalt voor een query het uitvoeringspad.

Om te achterhalen welk **pad** de optimizer koos voor een query, gebruiken we EXPLAIN PLAN (F10).

theorie x

0,031 seconds

Worksheet Explain Plan... (F10)

```
1 SELECT * FROM medewerkers where salaris=43000 AND afd_nr=3;
```

Script Output x Query Result x Explain Plan x

SQL | 0,031 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	MEDEWERKERS	BY INDEX ROWID	1	2
INDEX	IND_MED_AFD_SAL	RANGE SCAN	1	1

Access Predicates

AND


SALARIS=43000

AFD\_NR=3

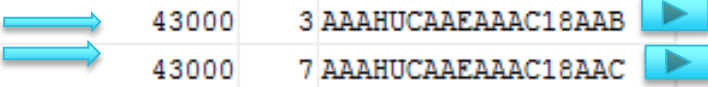
# Zoeken op samengestelde indexen:

## 1<sup>e</sup> component

INDEX op **salaris** + afd\_nr



salaris	afd_nr	rowid
25000	3	AAAHUCAAEAAAC18AAD
25000	3	AAAHUCAAEAAAC18AAE
25000	7	AAAHUCAAEAAAC18AAH
30000	7	AAAHUCAAEAAAC18AAF
38000	7	AAAHUCAAEAAAC18AAG
43000	3	AAAHUCAAEAAAC18AAB
43000	7	AAAHUCAAEAAAC18AAC
55000	1	AAAHUCAAEAAAC18AAA



```
SELECT * FROM medewerkers  
WHERE salaris=43000;
```

(dus er wordt getest op **1e component** van samengestelde index)

- Hoe wordt gezocht? **Range scanning**
- De optimizer zal in de index gericht zoeken naar de **salariswaarde 43000** en de bijbehorende rowid(s) ophalen.

In de indextabel zitten de attribuutwaarden (automatisch) gesorteerd op salaris en binnen salaris op afdeling

# Zoeken op samengestelde indexen: 1<sup>e</sup> component

theorie x

0,016 seconds

Worksheet Query Builder

```
1 SELECT * FROM medewerkers WHERE salaris=43000;
```

Script Output x Query Result x Explain Plan x

SQL | 0,016 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
TABLE ACCESS	MEDEWERKERS	BY INDEX ROWID	2	2
INDEX	IND_MED_AFD_SAL	RANGE SCAN	2	1

Access Predicates  
SALARIS=43000

# Zoeken op samengestelde indexen:

## 2<sup>e</sup> component

INDEX op salaris + afd\_nr



salaris	afd_nr	rowid
25000	3	AAAHUCAAEAAAC18AAD
25000	3	AAAHUCAAEAAAC18AAE
25000	7	AAAHUCAAEAAAC18AAH
30000	7	AAAHUCAAEAAAC18AAF
38000	7	AAAHUCAAEAAAC18AAG
43000	3	AAAHUCAAEAAAC18AAB
43000	7	AAAHUCAAEAAAC18AAC
55000	1	AAAHUCAAEAAAC18AAA
55000	1	AAAHUCAAEAAAC18AAA

```
SELECT * FROM medewerkers  
WHERE afd_nr=3;
```

(dus er wordt getest op de 2<sup>e</sup> component van de samengestelde index)

Hoe wordt gezocht? Skip scanning

- De optimizer zal salaris 25000 scannen op zoek naar afd\_nr 3. Bij een hit worden de bijbehorende rowid's opgehaald
- Van zodra een afd\_nr gevonden wordt dat groter is dan 3, stopt het scannen
- Nu wordt er verder gegaan (geskipt) naar salaris 30000, opnieuw op zoek naar een afdeling 3. Bij een hit worden de rowid's opgehaald. Het scannen stopt wanneer een grotere waarde voor afdeling wordt gevonden.
- Er wordt geskipt naar salaris 38000 ...

In de indextabel zitten de attribuutwaarden (automatisch) gesorteerd op salaris en binnen salaris op afdeling

# Zoeken op samengestelde indexen: 2<sup>e</sup> component

theorie x

0,021 seconds

Worksheet Query Builder

```
1 SELECT * FROM medewerkers
2 WHERE afd_nr=3;
3
4
```

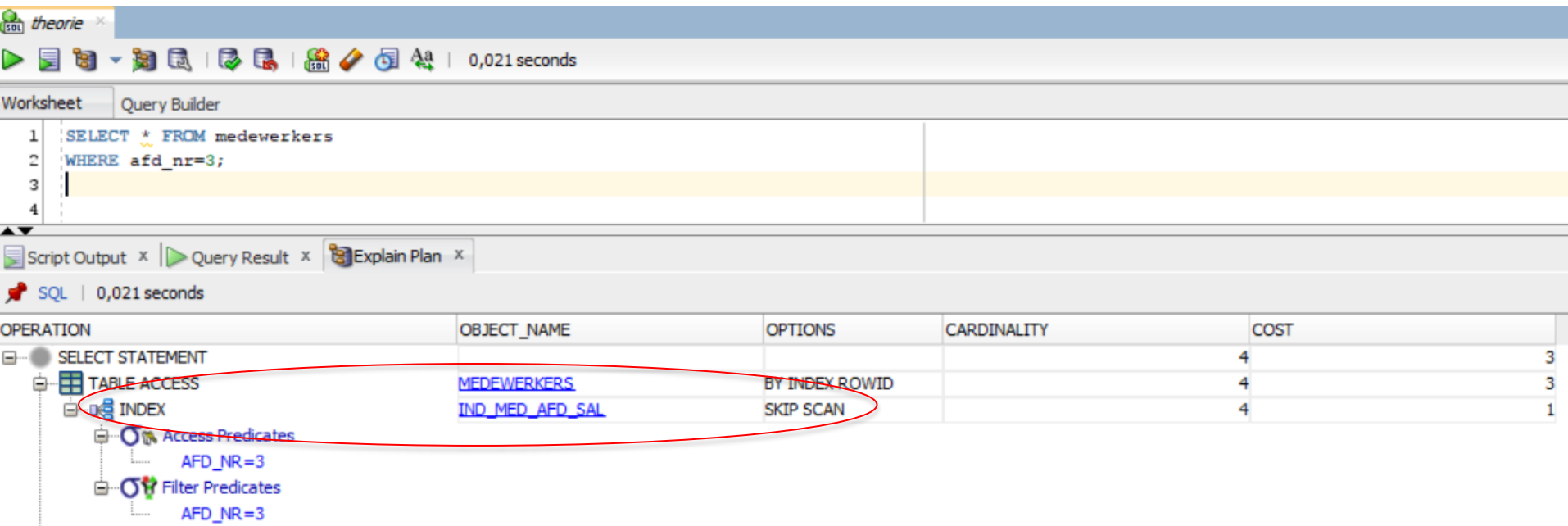
Script Output x Query Result x Explain Plan x

SQL | 0,021 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3
TABLE ACCESS	MEDEWERKERS	BY INDEX ROWID	4	3
INDEX	IND_MED_AFD_SAL	SKIP SCAN	4	1

Access Predicates  
AFD\_NR=3

Filter Predicates  
AFD\_NR=3



# Waarom indexen?

- Om unieke waarden af te dwingen  
(creatie unieke index/ UNIQUE of PK constraint)  
bij elke waarde van het geïndexeerd attribuut staat dan maximum 1 rowid.
- Om performantie redenen  
Full Table Scan versus indexgebruik



# Richtlijnen

- **Indexeer kolommen die veel in WHERE clauses voorkomen**
- **Maak voor kolommen die vaak samen voorkomen in de WHERE clause een samengestelde index.**

Denk aan de volgorde

- **Indexeer kolommen met een grote verscheidenheid aan waarden**

Hoe groter de verscheidenheid aan waarden, hoe interessanter de index

- **Indexeer kolommen met veel NULL waarden**

NULL waarden worden niet geïndexeerd (compacte index)

# Richtlijnen

## ➤ **Gebruik geen indexen op kleine tabellen**

een Full Table Scan is in dat geval sneller

## ➤ **Indexeer bij voorkeur tabellen die weinig veranderen**

bij elke wijziging in de tabel moet immers de indextabel ook aangepast worden.

## ➤ **Gebruik geen indexen wanneer queries vaak meer dan 4% van de rijen ophalen.**

bij een FTS worden de data blokken sequentieel doorlopen => elk data blok wordt 1 keer doorlopen.

bij een index scan is het mogelijk dat een data blok meerdere keren doorlopen wordt

# Op welk moment Indexeren?

- ☐ Tabel creëren
- ☐ Tabel opvullen
- ☐ Index creëren



Bij gewone index

- ☐ Tabel creëren
- ☐ Index creëren
- ☐ Tabel vullen



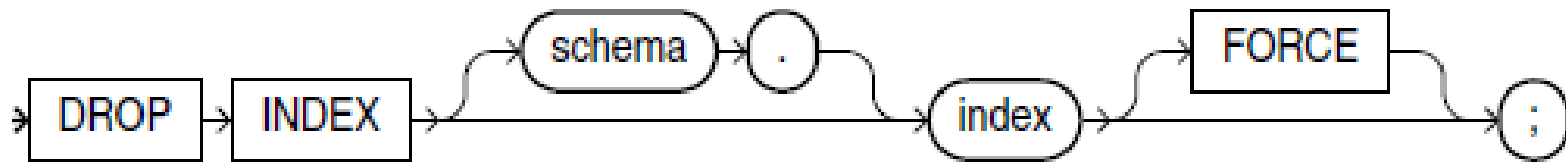
Bij unieke index

# Index verwijderen



Definitie  
**SQL**  
Reference  
p. 134

*drop\_index ::=*



Men verwijdert een index omdat

- hij niet meer gebruikt wordt door de applicatie
- hij niet de verwachte performantie geeft

```
DROP INDEX ind_med_geslacht;
```

# Index verwijderen

## Bemerkingen:

- Bij het verwijderen van een tabel worden de indexen automatisch mee verwijderd
- Een index die geassocieerd is met een PRIMARY KEY of UNIQUE constraint kan niet verwijderd worden via DROP INDEX
- Het disablen van een PK of UNIQUE constraint maakt de index tijdelijk ontoegankelijk

# Data dictionary

ALL_INDEXES	ALL_IND_COLUMNS
DBA_INDEXES	DBA_IND_COLUMNS
USER_INDEXES	USER_IND_COLUMNS

of in SQL Developer:

The screenshot shows the SQL Developer interface. On the left, the 'theorie1' schema is expanded, showing a tree of database objects: Tables (Filtered), Views, Editing Views, Indexes (highlighted), Packages, Procedures, and Functions. On the right, the 'MEDEWERKERS' table is selected, and the 'Indexes' tab is active (circled in red). The tab bar includes Columns, Data, Model, Constraints, Grants, Statistics, Triggers, Flashback, Dependencies, Details, Partitions, and Indexes. The main pane displays a table of indexes for the MEDEWERKERS table.

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PAR
1	THEORIE	PK_MEDEWERKERS	UNIQUE	VALID	NORMAL	N	NO
2	THEORIE	UN_PARKEERPLAATS	UNIQUE	VALID	NORMAL	N	NO