*Samenvatting Data - SQL*

***COMMANDO’S DATA MANIPULATIE:***

SELECT kolom selecteert een kolom om weer te geven uit een tabel

FROM tabel;

INSERT INTO tabel VALUES (#, #, #, #); nieuwe rij in tabel maken

INSERT INTO tabel gegevens uit een tabel naar een andere kopiëren

SELECT FROM kolom1, kolom2, kolom3

FROM tabel2

WHERE # = #;

UPDATE tabel SET kolom = #; gegevens updaten in een tabel

DELETE FROM tabel; maakt de tabel leeg

TRUNCATE TABLE tabel; maakt de tabel leeg

DELETE FROM tabel verwijdert bepaalde rijen uit een tabel

WHERE # = #;

CREATE TABLE # maakt een nieuwe tabel aan

ALTER TABLE # tabel na aanmaking aanpassen

ALTER TABLE # READ ONLY; zet een tabel naar alleen lezen

ALTER TABLE # READ WRITE; zet een tabel naar lezen en schrijven

DROP TABLE #; tabel uit bestaande databank halen

DROP TABLE # CASCADE CONSTRAINTS; tabel uit bestaande databank halen incl. references

COMMENT # opmerking # in instructie

GRANT toegang of privileges te verlenen aan de

database-objecten voor de gebruikers

REVOKE verwijderen van gebruikers toegangsrechten

COMMIT; wijzigingen permanent opslaan

SAVEPOINT maakt een opslagpunt in een groep acties waarin

rollback wordt uitgevoerd

SAVEPOINT #; een savepoint voor rollback aanmaken

ROLLBACK; wijzigingen terugdraaien naar het laatste savepoint

ROLLBACK TO #; wijzigingen terugdraaien naar savepoint #

SHOW ALL; toont alle systeemvariabelen

FROM # specificeert uit welke database de resultaten gehaald moeten

worden

# || ‘literal’ || # || zorgt ervoor dat meerdere verschillende literals of waarden in 1 kolom gezet kunnen worden

# || q’ [literal] ‘ || # zelfde functie als bovenstaand commando, enkel met Quote Operator

DESCRIBE # toont de structuur van de tabel #

WHERE # toont enkel de rijen waar de conditie # voldaan is

ORDER BY # sorteert de resultaten op kolom # of kolom alias

DEFINE # slaagt de waarde van de variabele # op

UNDEFINE # verwijdert de waarde van de variabele #

SET VERIFY ON ??

***PARAMETERS***

DISTINCT Verwijdert dubbele resultaten  
AS “#” # specificeert de naam die de kolom moet hebben  
DESC sorteert de resultaten van groot naar klein of van Z naar A  
ASC sorteert de resultaten van klein naar groot of van A naar Z  
&# vraagt een input van de gebruiker en slaagt dit op in variabele #  
&&# vraagt een input van de gebruiker en slaagt dit op voor later gebruik

***OPERATORS***

+ telt 2 waarden op

- trekt 2 waarden af

\* vermenigvuldigt 2 waarden

/ deelt 2 waarden

= is gelijk aan

> is groter dan

>= is groter of gelijk aan

< is kleiner dan

<= is kleiner of gelijk aan

<> is niet gelijk aan

!= is niet gelijk aan

BETWEEN # AND # ligt tussen # en #

IN (#, #, #, #) komt overeen met #, #, # of #

LIKE komt overeen met een patroon (\_ = 1 karakter, % = 0 of meer karakters) = hoofdletter gevoelig

IS NULL is gelijk aan een veld dat leeg is (geen waarde)

ESCAPE ‘\’ zegt tegen oracle dat het escape teken \ is

AND geeft true terug als beide statements voldaan zijn

OR geeft true terug als 1 van de 2 statements voldaan zijn

NOT geeft true terug als de conditie niet voldaan is

***CREATE TABLE OPTIES:***

NUMBER(plaatsen) getal

NUMBER(plaatsen totaal,plaatsen na de komma) getal

VARCHAR2(plaatsen) string

CHAR(plaatsen) string met vaste lengte

DATE datum

LONG string tot 2gb

CLOB string tot 4gb

DEFAULT # zet indien null, standaard op #

NOT NULL veld mag niet leeg zijn

UNIQUE moet uniek zijn in de gehele tabel

PRIMARY KEY is een PK

FOREIGN KEY # REFERENCES #(#) is een FK die naar een PK verwijst

CHECK controleert het veld a.d.h.v. een conditie

CONSTRAINT # geeft een naam aan opvolgende constraint

CREATE TABLE # AS [subquery] maakt tabel zoals subquery (select) zegt

***FUNCTIES***

LOWER (literal) verandert de literal naar kleine letters

UPPER (literal) verandert de literal naar grote letters

INITCAP (literal) verandert de eerste letter van elk woord naar een hoofdletter, de rest naar kleine letters

CONCAT (literal, literal) plakt de 2 literal’s aan elkaar vast

SUBSTR (literal, #, #) neemt de tekst van de literal van positie # tot # (met # als optie)

LENGTH (literal) keert het aantal karakters in de literal terug

INSTR (literal, ‘literal’, #, #) keert terug op welk karakter de literal in de literal is gevonden, met

als optie van positie # tot positie # te zoeken

LPAD (literal, #, ‘literal’) vult de literal # keer links op met literal en keert dit terug

RPAD (literal, #, ‘literal’) vult de literal # keer rechts op met literal en keert dit terug

TRIM (literal FROM literal) verwijdert literal uit literal

REPLACE (literal, literal, literal) zoekt in literal naar literal en indien gevonden, vervangen met literal

ROUND (literal, #) rond de literal op # aantal decimalen af

TRUNC (literal, #) kapt de literal op # aantal decimalen af

MOD (#, #) keert de rest terug van de deling tussen # en #

MONTHS\_BETWEEN (#, #) aantal maanden tussen 2 datums # en #

ADD\_MONTHS (#, #) voegt een # aantal maanden bij de datum #

NEXT\_DAY (#, #) geeft de volgende datum die een # dag van de week is na #

LAST\_DAY (#) geeft de laatste dag van die maand van de datum #

ROUND (literal, #) rond de literal op # dag, maand of jaar af

TRUNC (literal, #) kapt de literal op # dag, maand of jaar af

TO\_CHAR(#, [#, #]) zet een getal of datum # om naar een karakter in formaat # met

parameters #

TO\_NUMBER(#, [#, #]) zet een karakter # om naar een getal in formaat # met

parameters #

TO\_DATE(#, [#, #]) zet een karakter # om naar een datum in formaat # met

parameters #

NVL (#, #) als # gelijk is aan *null* dan wordt er # getoond

NVL2(#, #, #) als # niet gelijk is aan *null* wordt # getoond, anders #

NULLIF(#, #) geeft *null* terug als # en # gelijk zijn aan elkaar

COALESCE(#, #, …, #) geeft # terug als voorgaande # gelijk is aan *null*, als #, # en … allemaal

*null* zijn wordt # terug gegeven

CASE expr1 WHEN expr2 THEN return\_expr

[WHEN expr3 THEN return\_expr

WHEN expr4 THEN return\_expr

ELSE return\_default]

END

DECODE (expr1, expr2, return\_expr [, expr3, return\_expr, expr4, return\_expr, return\_default])

***FORMATEN***

***Met datums:***

YYYY 2017

YEAR two thousand and seventeen

MM 01

MONTH january

MON Jan

DY Mon

DAY Monday

D 1

“#” # wordt letterlijk getoond

DD 14

SP fourteen DDSPTH fourteenth

TH th

fm overbodige tabs en spaties weglaten

***Met getallen:***

9 een getal

0 forceert om een 0 te tonen

$ toont een dollar teken

U toont een euro teken

L toont de lokale geldeenheid

. toont een decimaal punt

, toont een komma om de duizendtallen

***JOINS:***

Natural joins:

SELECT department\_id, department\_name, location\_id, city

FROM departments  
 NATURAL JOIN locations;

USING joins:

SELECT employee\_id, last\_name, location\_id, department\_id

FROM employees JOIN departments  
 USING (department\_id);

ON joins:

SELECT e.employee\_id, e.last\_name, e.department\_id, d.department\_id, d.location\_id

FROM employees e JOIN departments d  
 ON (e.department\_id = d.department\_id);

Self joins:

zelfde als ON joins?

Nonequijoins:

SELECT e.last\_name, e.salary, j.grade\_level

FROM employees e JOIN job\_grades j  
 ON e.salary BETWEEN j.lowest\_sal AND j.highest\_sal;

Bovenstaande joins zijn allemaal voorbeelden van een INNER join. Dit betekent dat als er gegevens zijn die niet in de gejoinde tabel voorkomen, deze ook niet getoond worden.

Hier komen OUTER joins tot de redding. Een LEFT OUTER JOIN zorgt ervoor dat alle elementen uit de linkse tabel getoond worden, ook al is er rechts geen resultaat (null). Een RIGHT OUTER JOIN zorgt ervoor dat alle elementen uit de rechtse tabel getoond worden, ook al is er links geen resultaat (null). Een FULL OUTER JOIN zorgt ervoor dat alle elementen uit beide tabellen worden getoond, het maakt niet uit of er niets overeenkomt (null) is.

Een Cartesian product of een CROSS JOIN, joint 2 tabellen waarbij elk deel van links gematcht wordt met elk deel van rechts. Een tabel van 8 links en een tabel van 20 rechts zorgen voor een tabel van 160 lang (8 x 20 = 160).

***Groep-functies:***

AVG([DISTINCT|ALL] #) Bepaald het gemiddelde van #

COUNT(\*|[DISTINCT|ALL] #) Bepaald het aantal rijen van \* of #

MAX([DISTINCT|ALL] #) Bepaald het maximum van #

MIN([DISTINCT|ALL] #) Bepaald het minimum van #

SUM([DISTINCT|ALL] #) Bepaald de som van de #-rijen

~~STDDEV([DISTINCT|ALL] #) Bepaald de standaard deviatie van #~~

~~VARIANCE([DISTINCT|ALL] #) Bepaald de variantie van #~~

GROUP BY #, # groepeert gegevens op # en nadien op #

HAVING # i.p.v. WHERE bij groepfuncties

***Subqueries:***

IN komt voor in 1 of meerdere returns van de subquery

ANY vergelijkt een waarde met elke waarde in de return van de subquery, toont false als er geen

rijen zijn, moet voorgegaan zijn door =, !=, >, <, <= of >=

(moet overeenkomen met 1 van de returns)

ALL vergelijkt een waarde met elke waarde in de return van de subquery, toont true als er geen

rijen zijn, moet voorgegaan zijn door =, !=, >, <, <= of >=

(moet overeenkomen met ALLE returns)

***Views:***

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW viewnaam [alias, alias, alias, …]

AS subquery

[WITH READ ONLY];

OR REPLACE vervang indien reeds bestaat

FORCE maak ook als tabel niet bestaat

NOFORCE maak alleen als tabel ook bestaat

alias kolomalias van de view

***Sequences:***

CREATE SEQUENCE sequencenaam

[INCREMENT BY #]

[START WITH #]

[MAXVALUE # | NOMAXVALUE]

[MINVALUE # | NOMINVALUE]

[CYCLE | NOCYCLE]

[CACHE # | NOCACHE];

CYCLE sequentie moet verder blijven genereren ook als min of max bereikt is

CACHE zegt dat Oracle deze gegevens in snel bereik zet voor snelle uitvoeringen in de

toekomst, # is hoeveel waarden hij bijhoudt

NEXTVAL geeft de volgende waarde in de sequentie

CURRVAL geeft de huidige waarde in de sequentie

om CURRVAL te gebruiken moet NEXTVAL minstens 1 keer uitgevoerd zijn

***Indexes:***

CREATE [UNIQUE] [BITMAP] INDEX indexnaam

ON TABLE (kolom, kolom, kolom, …);

UNIQUE waardes van de geïndexeerde kolom moeten uniek zijn

BITMAP creëert een bitmap voor elke PK i.p.v. indexering per rij

***Synonyms:***

CREATE [PUBLIC] SYNONYM synonymnaam

FOR object;

PUBLIC toegankelijk voor alle gebruikers