Creatie van een tabel

CREATE TABLE

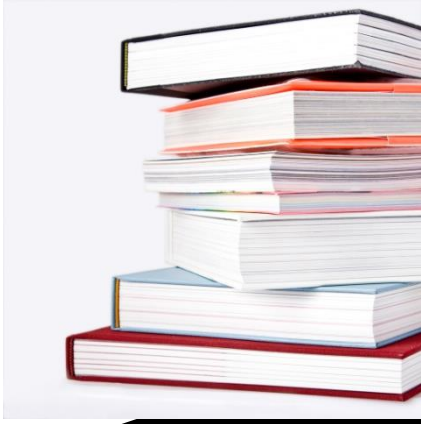
KdG Karel de Grote Hogeschool

Cursus
Boek

cursusmateriaal

- cursus 'Databanken 1' blz. 47
- SQL Reference blz. 109
- Deze powerpoint

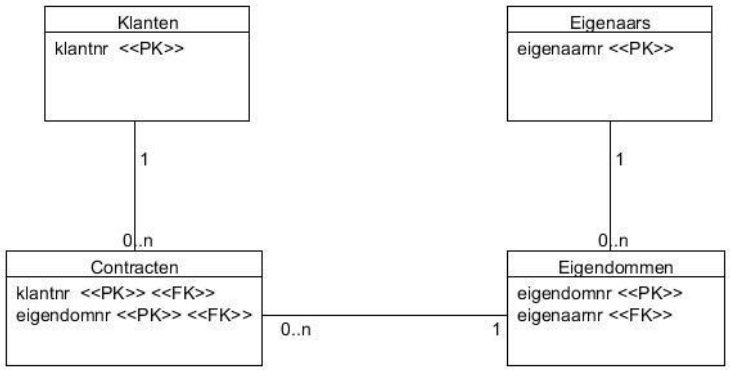
- Extra's:
- Hoofdstuk 11 handboek 'SQL fundamentals I Exam Guide' blz. 450-482
- https://docs.oracle.com/cd/B28359_01/server.111/b28310/tables003.htm
- http://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_7002.htm



KdG Karel de Grote Hogeschool

Creatie van een tabel

ERD immo-kantoor



zal leiden tot ➔

CREATE TABLE *klanten*

(klantnr NUMBER(5) CONSTRAINT pk_klant PRIMARY KEY,

Klantnaam VARCHAR2(20) CONSTRAINT c_klantnaam CHECK(klantnaam=UPPER(klantnaam)));

CREATE TABLE *eigenaars*

(eigenaarnr NUMBER(5) CONSTRAINT pk_eigenaar PRIMARY KEY,

eigenaarnaam VARCHAR2(20) CONSTRAINT c_eigenaarnaam CHECK(eigenaarnaam =UPPER(eigenaarnaam)));

CREATE TABLE *eigendommen*

(eigendomnr NUMBER(5)CONSTRAINT pk_eigendom PRIMARY KEY,

eigendomadres VARCHAR2(50),

Huurprijs NUMBER(4) CONSTRAINT c_huurprijs CHECK(huurprijs BETWEEN 500 AND 2000),

eigenaarnr NUMBER(5) CONSTRAINT fk_eigendom_eigenaar REFERENCES eigenaars);

CREATE TABLE *contracten*

(klantnr NUMBER(5) CONSTRAINT fk_contract_klant REFERENCES klanten,

Eigendomnr NUMBER(5) CONSTRAINT fk_contract_eigendom REFERENCES eigendommen,

Beginhuur DATE,

Eindehuur DATE,

CONSTRAINT pk_contract PRIMARY KEY (klantnr,eigendomnr),

CONSTRAINT c_begin_eind CHECK(beginhuur<eindehuur));

Creatie van een tabel

Besluit:

Voor elk van de entiteiten uit het ERD, moeten we bepalen

hoe we die **entiteiten** gaan **noemen**,
hoe we de **attributen** gaan **noemen**,
hoe groot elk van de **attributen** moet zijn ,
welke soort informatie ze mogen **bevatten**.

We moeten ook nadenken over **beperkingen die we op attributen van de entiteiten gaan opleggen.**

Creatie van een tabel

Om een tabel te kunnen creëren moet je over de volgende informatie beschikken:

- een **naam** voor de **tabel (we kiezen voor meervoudsvorm)**
- **namen** voor de **attributen** uit de **tabel**
- **voor** elk **attribuut** het **gegevenstype en** de **grootte**
- **eventueel** een **default waarde** voor een attribuut
- **eventueel** beperkingen (=**constraints**) die je aan de attributen oplegt

Creatie van een tabel

De syntax voor het create table statement ziet er als volgt uit:

CREATE TABLE *tabelnaam*

(*attribuutnaam gegevenstype [default waarde] [column constraint…],*

…,

[table constraint],

…);

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

- uit de tabelnaam moet duidelijk blijken wat de inhoud van de tabel is
  bv. tabel waarin klantgegevens zitten noem je best KLANTEN

- een tabelnaam moet uniek zijn binnen een schema.
  user theorie kan bv maar 1 tabel KLANTEN hebben

- een tabelnaam moet voldoen aan een aantal voorwaarden:
  - max. 30 teken
  - beginnen met een letter
  - toegelaten tekens : letters, cijfers, _, @, $, #
  - geen spaties
  - geen SQL sleutelwoorden (CREATE, ALTER, …)

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

- moeten uniek zijn binnen de tabel

- moeten aan dezelfde voorwaarden voldoen als tabellen

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Voor elk attribuut moet er bepaald worden welk soort gegevens het mag bevatten (=domein) en hoeveel tekens het mag bevatten.

We maken een onderscheid tussen

alfanumerieke attributen,

numerieke attributen,

datum attributen.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

ALFANUMERIEK

voor alfanumerieke attributen gebruiken we de gegevenstypes

CHAR(n) en

VARCHAR(n) of VARCHAR2(n)

Creatie van een tabel

tabelnaam – atributnaam – gegevenstype – default waarde – constraints

ALFANUMERIEK

CHAR(n) wordt gebruikt voor alfanumerieke attributen waarvan de **inhoud** een **vaste lengte** heeft

Vb. *telefoonnummer CHAR(9)*
artikelcode CHAR(7)
rijksregisternummer …

Als de attribuutwaarde van artikelcode 6 posities bevat wordt artikelcode achteraan aangevuld met een blanco.
Als de attribuutwaarde van artikelcode 8 posities bevat wordt een foutmelding getoond

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

ALFANUMERIEK

VARCHAR2(n) wordt gebruikt voor alfanumerieke attributen met **variabele lengte**

bv	*naam*	*VARCHAR2(50)*
	straat	*VARCHAR2(40)*

Er wordt maar zoveel ruimte gebruikt als nodig is voor de attribuutwaarde.

Als de attribuutwaarde van naam 20 posities groot is , worden maar 20 posities gebruikt

Als de attribuutwaarde van naam 55 posities groot is wordt een foutmelding getoond.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

ALFANUMERIEK

TIP: gebruik het datatype CHAR(n) enkel wanneer je zeker weet dat de attribuutwaarden een vaste lengte hebben.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

ALFANUMERIEK

Andere alfanumerieke datatypes:

CLOB (=Character Large Object) :

- kan tot 4GB maal de database blokgrootte bevatten
- wordt gebruikt voor stockering van grote documenten

LONG

- kan tot 2GB aan karakters bevatten
- is verouderd en wordt volledig vervangen door het CLOB dataype.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

NUMERIEK

Numerieke attributen zijn attributen waarmee kan gerekend worden.

Voor deze attributen gebruiken wij de gegevenstypes

NUMBER(n)
NUMBER(n,m)
NUMBER

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

NUMERIEK

NUMBER(n) wordt gebruikt voor attributen die enkel gehelen zullen bevatten.

n geeft het aantal gehelen weer waarvoor ruimte wordt voorzien.

Vb. *aantal_stuks NUMBER(4)*

wanneer de attribuutwaarde meer dan n tekens bevat ➔ foutmelding
wanneer de attribuutwaarde ook decimalen bevat wordt er afgerond.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

NUMERIEK

NUMBER(n,m) wordt gebruikt voor attributen die decimalen kunnen bevatten.

n geeft het totaal aantal tekens weer,
m geeft het aantal decimalen weer (kan een negatief getal zijn)

Vb. *eenheidsprijs NUMBER(5,2)*

wanneer de attribuutwaarde meer dan n-m gehelen bevat ➔ foutmelding

wanneer de attribuutwaarde meer dan m decimalen bevat➔ afronding

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

NUMERIEK

NUMBER wordt gebruikt wanneer men niet kan inschatten hoe groot de attribuutwaarde wordt.

Het attribuut kan dan een onbeperkt aantal gehelen en tot 38 decimalen bevatten.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

<u>Voorbeeld</u>:

***waarde*: 1234.567**

Gegevenstype	**inhoud**
NUMBER(8)	1235
NUMBER(4)	1235
NUMBER(3)	FOUTMELDING
NUMBER	1234.567
NUMBER(5,2)	FOUTMELDING
NUMBER(6,2)	1234.57
NUMBER(6,-2)	1200

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

DATUMATTRIBUTEN

Voor datumattributen geldt het gegevenstype **DATE**. Standaard staat de datum DD-MON-YYYY. Je kan de datum anders instellen.

De datum bevat eeuw, het jaar, de maand, de dag, het uur, de minuten, de seconden

Als een datum opgeslagen wordt zonder tijd, dan is deze 0 (middernacht).

SYSDATE is standaard de datum van vandaag.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

DATUMATTRIBUTEN

TIMESTAMP:

het DATE datatype kan eeuw, jaar, maand, dag, uren, minuten en seconden stockeren met een precisie tot op 1 seconde

met het TIMESTAMP datatype kan je tot op fracties van seconden stockeren.
TIMESTAMP gebruikt standaard 6 decimale posities voor seconden.

Vb de tabel TESTDATUM:

COLUMN_NAME	DATA_TYPE
DATUM1	DATE
DATUM2	TIMESTAMP(6)

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

DATUMATTRIBUTEN

COLUMN_NAME	DATA_TYPE
DATUM1	DATE
DATUM2	TIMESTAMP(6)

TIMESTAMP:

```
CREATE TABLE testdatum(
datum1 DATE,
datum2 TIMESTAMP(6));
```

```
SELECT TO_CHAR(datum2,'DD-MON-YYYY HH24:MI:SS FF') Datum
FROM testdatum;
```

DATUM
23-SEP-2016 09:57:03 000000

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

DATUMATTRIBUTEN

Datum datatypes die verschil tonen tussen 2 datums

INTERVAL YEAR TO MONTH

INTERVAL DAY TO SECOND

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

ANDERE DATATYPES

Binaire datatypes worden gebruikt om binaire gegevens zoals beelden, audio, video… te stockeren.

- BLOB (binary large object)
- RAW

  voor variabele lengte binaire gegevens (verouderd)
- LONG RAW

  zoals LONG maar voor binaire gegevens (verouderd)

  vervangen door BLOB

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

Bfile:
bevat een pointer naar een binaire file in het operating system van de database server.
wordt gebruikt voor stockering van videobestanden, grafische bestanden, audio bestanden

ROWID:
elke rij in de database bevat een rowid. Deze bevat het exacte fysieke adres van die rij in de database.
Dus elke tabel uit de database bevat een rowid pseudokolom waarin 6 byte binaire waarden kunnen opgeslagen worden.
het ROWID datatype wordt gebruikt om het rowid in een leesbaar formaat op te slagen.
(functie ROWIDTOCHAR)

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde – constraints

Je kan een attribuut een **default** waarde toekennen.

Wanneer de tabel met gegevens gevuld wordt en het attribuut krijgt voor een bepaalde rij geen waarde, dan wordt de default waarde ingevoerd voor die rij.

status	*CHAR(1) DEFAULT*	***'Y'***
aankoopdatum	*DATE*	*DEFAULT SYSDATE*

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde
- constraints

```
CREATE TABLE  afdelingen
(afd_nr NUMBER(2,0) CONSTRAINT pk_afdelingen PRIMARY KEY,
afd_naam VARCHAR2(20) CONSTRAINT nn_afd_naam NOT NULL,
mgr_sofi_nr CHAR(9) CONSTRAINT fk_afd_mdw
        REFERENCES  medewerkers(sofi_nr),
mgr_start_datum DATE  DEFAULT SYSDATE);
```

Als je in de tabel AFDELINGEN een nieuwe rij toevoegt en je vult geen waarde in voor afd_mgr_start_datum, dan wordt default de datum van vandaag ingevuld.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Door middel van **constraints** kan je extra beperkingen opleggen aan attributen.

Je kan een beperking opleggen aan 1 attribuut (=**column constraint**)

OF

je kan een beperking opleggen aan een combinatie van attributen (**table constraint**).

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

soorten constraints:

- **PRIMARY KEY constraint**
  (=*key constraint + entity integrity constraint*)
- **NOT NULL constraint**
- **CHECK constraint**
- **UNIQUE constraint**
- **FOREIGN KEY constraint**
  (=*referential integrity constraint*)

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Een **column constraint** is een **beperking** die je **op één kolom** legt. De constraint wordt bij die kolom gedefinieerd!

Syntax:

CONSTRAINT *constraintnaam column-constraint*

We spreken af dat we een constraint steeds benoemen!

Een constraint naam moet uniek zijn binnen de gebruiker.

Column constraint

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

NOT NULL constraint

De constraint waakt ervoor dat het **attribuut** steeds een **geldige waarde** krijgt. Vult de gebruiker geen waarde in voor het attribuut, dan zal het DBMS automatisch reageren met een foutmelding.

Syntax: CONSTRAINT *constraintnaam* **NOT NULL**

Vb.

achternaam VARCHAR2(25)
CONSTRAINT nn_achternaam NOT NULL

Column constraint

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

PRIMARY KEY constraint

Deze constraint geeft aan dat het betreffende attribuut de primaire sleutel is van de tabel.

Syntax:
CONSTRAINT *constraintnaam* **PRIMARY KEY**

Vb

sofi_nr CHAR(9)
*CONSTRAINT pk_medewerkers **PRIMARY KEY***

Het plaatsen van de PK constraint creëert automatisch

- ***een unieke index*** (key constraint)
- ***een NOT NULL constraint*** (entity integrity constraint)

Column constraint

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

CHECK constraint

Deze constraint bepaalt het **bereik van de attribuutwaarden (=domein)** van een bepaald attribuut

vb. Salaris moet kleiner zijn dan 25000

Syntax:
CONSTRAINT *constraintnaam* CHECK (*conditie*)

vb.
salaris NUMBER(7,2) CONSTRAINT c_salaris
CHECK(salaris <=25000)

Column constraint

Condities in CHECK constraint

Syntax:

CONSTRAINT *constraintnaam* CHECK (*conditie*)

De volledige syntax rond condities wordt in periode 4 aangeleerd.

In de CHECK constraint wordt aangegeven aan welke conditie attribuutwaarden moeten voldoen.

We bekijken een aantal van de condities die je in een CHECK constraint kan definiëren.

Column constraint

Condities in CHECK constraint

vergelijkingen

Syntax:

[NOT] *identifier1* > *identifier2*
<
=
>=
<=
< > (verschillend van)
!= (verschillend van)

Identifier kan een attribuut, een constante of een rekenkundige bewerking zijn.

Column constraint

Condities in CHECK constraint

vergelijkingen

Voorbeelden:

- salaris < 3000
- afdnaam='ADMINISTRATIE'

(*inhoud van een attribuut is hoofdlettergevoelig)*

- begindatum='31-OCT-2017'

Column constraint

Condities in CHECK constraint

BETWEEN … AND

Waarden moeten binnen een bepaalde boven- en ondergrens liggen

Syntax:

[NOT] *identifier* BETWEEN *ondergrens* AND *bovengrens*

- Identifier kan een attribuut, een constante of een rekenkundige bewerking zijn.
- De onder- en bovengrens zijn inbegrepen!!
- De ondergrens moet steeds kleiner zijn dan de bovengrens

Column constraint

Condities in CHECK constraint

BETWEEN … AND

Voorbeelden:

- salaris BETWEEN 3000 AND 5000

- naam BETWEEN ‘Jansen’ AND ‘Vervoort’

- begindatum BETWEEN ’01-JAN-2017’
  AND ’30-JUN-2017’

Column constraint

Condities in CHECK constraint

IN

- Waarden moeten in de opsomming voorkomen

Syntax:

[NOT] *identifier* IN *(waarde1,waarde2, …)*

Identifier kan een attribuut, een constante of een rekenkundige bewerking zijn.

Column constraint

Condities in CHECK constraint

IN

Voorbeelden:

- afd in (10,20,30,40)
- afdnaam in (‘ADMINISTRATIE’,’PRODUCTIE’)
- begindatum in (’30-NOV-2017,’5-DEC-2017’)

Column constraint

Condities in CHECK constraint

gebruik UPPER en LOWER functies

- UPPER functie

  De UPPER functie zet karakterwaarde om in uppercase

  Syntax :

  UPPER(karakterwaarde)

- LOWER functie

  De LOWER functie zet karakterwaarde om in lowercase

  Syntax:

  LOWER(karakterwaarde)

Column constraint

Condities in CHECK constraint

gebruik UPPER en LOWER functies

- wanneer de CHECK constraint eist dat een attribuutwaarde steeds in uppercase moet ingegeven worden schrijf je:

  CHECK (attribuutnaam=UPPER(attribuutnaam))

- wanneer de CHECK constraint eist dat een attribuutwaarde steeds in lowercase moet ingegeven worden schrijf je:

  CHECK (attribuutnaam=LOWER(attribuutnaam))

Column constraint

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

UNIQUE constraint
Deze constraint dwingt voor het betreffende attribuut unieke waarden af.

Syntax:
CONSTRAINT constraintnaam **UNIQUE**

Vb.
nr_identiteitskaart CHAR(9) CONSTRAINT u_nr_id **UNIQUE**

Column constraint

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

FOREIGN KEY constraint

Deze constraint dwingt voor de verwijssleutel de referentiële integriteit af (referential integrity constraint).

Door het plaatsen van deze constraint op een verwijssleutel zal het DBMS steeds automatisch ***controleren of de waarden van de verwijssleutel als primaire sleutelwaarden voorkomen in de tabel waarnaar verwezen wordt*.**

Column constraint

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Syntax:

CONSTRAINT constraintnaam
REFERENCES *tabel (attribuut)* [ON DELETE CASCADE/SET NULL]

tabel waarnaar verwezen wordt

primaire sleutel uit de tabel waarnaar verwezen wordt

Vb.

afd_nr NUMBER(2) CONSTRAINT fk_med_afd ***REFERENCES afdelingen(afd_nr)***

Hierbij verwijst afd_nr uit de tabel MEDEWERKERS naar afd_nr uit de tabel AFDELINGEN.

Creatie van een tabel

Syntax:

CREATE TABLE *tabelnaam*

(*attribuutnaam gegevenstype [default waarde] [column constraint…],*

attribuutnaam2 gegevenstype [default waarde] [column constraint…],

…,

[table constraint],…);

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Table constraints

Een table constraint is een beperking waarbij **meer dan één attribuut uit eenzelfde tabel betrokken** is. Deze constraints worden niet op attribuut niveau gedefinieerd maar worden pas op het einde van de tabeldefinitie gedefinieerd.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Syntax Table constraint:

CONSTRAINT *constraintnaam table constraint*

Opgepast!

De syntax van een table constraint kan verschillen van de syntax van een column constraint.

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

CHECK constraint

Syntax :
CONSTRAINT *constraintnaam* CHECK (*samengestelde conditie*)

Vb. Stel dat commissieloon en salaris 2 attributen zijn uit dezelfde tabel. Als we de beperking opleggen dat commissieloon + salaris < 6000 dan is dat een table constraint omdat bij de constraint 2 attributen betrokken zijn!

constraint c_sal_comm
CHECK(salaris+commissieloon<6000)

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

PRIMARY KEY constraint (sleutel is samengesteld)

Syntax:

CONSTRAINT *constraintnaam* PRIMARY KEY*(attr1,attr2,…)* syntax!!

Vb. de primaire sleutel van de tabel OPDRACHTEN:

CONSTRAINT pk_opdracht PRIMARY KEY

(sofi_nr, proj_nr)

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

UNIQUE constraint
(een combinatie van attributen moet uniek zijn)

Syntax:
CONSTRAINT *constraintnaam* UNIQUE(attr1, attr2,…) syntax!!

Vb. de combinatie sofi_nr en parkeerplaats is uniek

CONSTRAINT u_sofi_nr_parkeerplaats
UNIQUE(sofi_nr , parkeerplaats)

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Foreign key constraint
(de vreemde sleutel is samengesteld)

Syntax:
CONSTRAINT *constraintnaam* FOREIGN KEY(*attr1,attr2,…*) REFERENCES *Tabelnaam(attri,attrj…)*
[ON DELETE CASCADE/SET NULL]

tabel waarnaar verwezen wordt

samengestelde primaire sleutel uit tabel waarnaar verwezen wordt

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Bemerking: de NOT NULL constraint kan **NOOIT** als table constraint voorkomen!

Creatie van een tabel

tabelnaam – atribuutnaam – gegevenstype – default waarde - constraints

Het gebruik ON DELETE CASCADE of ON DELETE SET NULL bij FOREIGN KEY constraint

syntax:

column constraint:

CONSTRAINT *constraintnaam*
REFERENCES *tabel (attribuut)*
[ON DELETE CASCADE /SET NULL]

table constraint:

CONSTRAINT *constraintnaam*
FOREIGN KEY(*attr1,attr2,…)*
REFERENCES *tabel (attri,attrj…)*
[ON DELETE CASCADE/SET NULL]

Situatie 1

Default geldt ON DELETE RESTRICT dwz. noch ON DELETE CASCADE noch ON DELETE SET NULL worden gebruikt.

We nemen als voorbeeld de verwijssleutel van de tabel OPDRACHTEN naar de tabel PROJECTEN.

```
CREATE TABLE projecten
(proj_nr          NUMBER(2)
   CONSTRAINT pk_projecten PRIMARY KEY,
 proj_naam            VARCHAR2(25)
   CONSTRAINT nn_proj_naam NOT NULL,
 locatie      VARCHAR2(25),
 afd_nr       NUMBER(2)
   CONSTRAINT fk_proj_afd REFERENCES Afdelingen(afd_nr));

CREATE TABLE opdrachten
(sofi_nr      CHAR(9)
   CONSTRAINT fk_opd_med REFERENCES medewerkers(sofi_nr),
proj_nr           NUMBER(2)
   CONSTRAINT fk_opd_proj REFERENCES Projecten(proj_nr),
uren          NUMBER(5,1),
CONSTRAINT pk_opdracht PRIMARY KEY (sofi_nr, proj_nr));
```

Situatie 1

Restricted DELETE betekent dat een poging om de rij met PROJ_NR 1 te verwijderen zal geweigerd worden omdat er nog afhankelijke rijen zijn.

Inhoud tabel PROJECTEN

PROJ_NR	PROJ_NAAM	LOCATIE	AFD_NR
1	Orderverwerking	Oegstgeest	7
2	Salarisadministratie	Groningen	7
3	Magazijn	Eindhoven	7
10	Inventaris	Maastricht	3
20	Personeelszaken	Eindhoven	1
30	Debiteuren	Maastricht	3

Inhoud tabel OPDRACHTEN ➔

SOFI_NR	PROJ_NR	UREN
999111111	1	31,4
999111111	2	8,5
999333333	3	42,1
999888888	1	21
999888888	2	22
999444444	2	12,2
999444444	3	10,5
999444444	1	(null)
999444444	10	10,1
999444444	20	11,8
999887777	30	30,8
999887777	10	10,2
999222222	10	34,5
999222222	30	5,1
999555555	30	19,2
999555555	20	14,8
999666666	20	(null)

Situatie 2

Gebruik van ON DELETE CASCADE
We nemen opnieuw als voorbeeld de verwijssleutel van OPDRACHT naar PROJECT.

```
CREATE TABLE projecten
(proj_nr              NUMBER(2)
   CONSTRAINT pk_projecten PRIMARY KEY,
 proj_naam            VARCHAR2(25)
   CONSTRAINT nn_proj_naam NOT NULL,
 locatie     VARCHAR2(25),
 afd_nr      NUMBER(2)
   CONSTRAINT fk_proj_afd REFERENCES Afdelingen(afd_nr));

CREATE TABLE opdrachten
(sofi_nr    CHAR(9)
   CONSTRAINT fk_opd_med REFERENCES
medewerkers(sofi_nr),
proj_nr           NUMBER(2)
   CONSTRAINT fk_opd_proj REFERENCES
Projecten(proj_nr) ON DELETE CASCADE,
uren          NUMBER(5,1),
CONSTRAINT pk_opdracht PRIMARY KEY (sofi_nr, proj_nr));
```

Situatie 2

Inhoud tabel PROJECTEN:

PROJ_NR	PROJ_NAAM	LOCATIE	AFD_NR
1	Orderverwerking	Oegstgeest	7
2	Salarisadministratie	Groningen	7
3	Magazijn	Eindhoven	7
10	Inventaris	Maastricht	3
20	Personeelszaken	Eindhoven	1
30	Debiteuren	Maastricht	3

Inhoud tabel OPDRACHTEN:

SOFI_NR	PROJ_NR	UREN
999111111	1	31,4
999111111	2	8,5
999333333	3	42,1
999888888	1	21
999888888	2	22
999444444	2	12,2
999444444	3	10,5
999444444	1	(null)
999444444	10	10,1
999444444	20	11,8
999887777	30	30,8
999887777	10	10,2
999222222	10	34,5
999222222	30	5,1
999555555	30	19,2
999555555	20	14,8
999666666	20	(null)

Wanneer de rij uit PROJECTEN wordt verwijderd dan zullen de rijen uit OPDRACHTEN mee worden verwijderd.

Situatie 3

Gebruik van ON DELETE SET NULL
We nemen opnieuw als voorbeeld de verwijssleutel van OPDRACHTEN naar PROJECTEN.

```
CREATE TABLE projecten
(proj_nr              NUMBER(2)
   CONSTRAINT pk_projecten PRIMARY KEY,
 proj_naam            VARCHAR2(25)
   CONSTRAINT nn_proj_naam NOT NULL,
 locatie      VARCHAR2(25),
 afd_nr       NUMBER(2)
   CONSTRAINT fk_proj_afd REFERENCES Afdelingen(afd_nr));

CREATE TABLE opdrachten
(sofi_nr     CHAR(9)
   CONSTRAINT fk_opd_med REFERENCES medewerkers(sofi_nr),
proj_nr           NUMBER(2)
   CONSTRAINT fk_opd_proj REFERENCES Projecten(proj_nr)
ON DELETE SET NULL,
uren         NUMBER(5,1),
CONSTRAINT pk_opdracht PRIMARY KEY (sofi_nr, proj_nr));
```

Situatie 3

Inhoud tabel PROJECTEN:

PROJ_NR	PROJ_NAAM	LOCATIE	AFD_NR
1	Orderverwerking	Oegstgeest	7
2	Salarisadministratie	Groningen	7
3	Magazijn	Eindhoven	7
10	Inventaris	Maastricht	3
20	Personeelszaken	Eindhoven	1
30	Debiteuren	Maastricht	3

Inhoud tabel OPDRACHTEN:

SOFI_NR	PROJ_NR	UREN
999111111	NULL	31,4
999111111	2	8,5
999333333	3	42,1
999888888	NULL	21
999888888	2	22
999444444	2	12,2
999444444	3	10,5
999444444	NULL	(null)
999444444	10	10,1
999444444	20	11,8
999887777	30	30,8
999887777	10	10,2
999222222	10	34,5
999222222	30	5,1
999555555	30	19,2
999555555	20	14,8
999666666	20	(null)

Wanneer de rij uit PROJECTEN wordt verwijderd dan worden de waarden van de verwijssleutels in de betrokken afhankelijke rijen NULL.

Bemerkingen

Elke column constraint (met uitzondering van NOT NULL) kan als table constraint gedefinieerd worden.

Je kan schrijven:

```
CREATE TABLE afdelingen
  (afd_nr                       NUMBER(2),
   afd_naam                     VARCHAR2(20)
        CONSTRAINT nn_afd_naam NOT NULL,
   sofi_nr        CHAR(9),
   start_datum     DATE,
  CONSTRAINT pk_afdeling PRIMARY KEY(afd_nr));
```

Bemerkingen

Omgekeerd, kan een table constraint (constraint op meer dan 1 attribuut) NIET als column constraint gedefinieerd worden.

Foutief zou zijn:

CREATE TABLE locaties

(afd_nr NUMBER(2)

CONSTRAINT fk_loc_afd REFERENCES afdelingen(afd_nr)

CONSTRAINT pk_locatie PRIMARY KEY(*afd_nr),*

plaats VARCHAR2(20)

CONSTRAINT pk_locatie PRIMARY KEY(*plaats));*

=> je krijgt dan de melding dat je geen 2 PK's op 1 tabel kan plaatsen.

Bemerkingen

Wanneer op een attribuut meerdere column constraints staan zet je tussen deze constraint definities geen ‘, ’ (komma) omdat een komma het einde van een attribuutbeschrijving weergeeft.

Bv.

```
achternaam VARCHAR2(20)
  CONSTRAINT c_achternaam CHECK(achternaam=UPPER(achternaam)
  CONSTRAINT nn_achternaam NOT NULL,
```

Bemerkingen

Moet een attribuut met een DEFAULT waarde nog voorzien worden van een NOT NULL constraint?

Vb.

```
mgr_start_datum DATE default SYSDATE
        CONSTRAINT nn_start_datum NOT NULL,
```

Een default waarde is een **initiële** waarde.

Die kan je achteraf nog altijd vervangen door een NULL waarde. Als het attribuut steeds een geldige waarde moet krijgen, moet je het dus ook voorzien van een NOT NULL constraint.