KU LEUVEN

Inleiding tot databanken

3. Vraagtalen – Deel 2

Prof. dr. Paolo Pilozzi

Overzicht

3.1 Vraagtalen
3.2 Relationeel Model
3.3 Relationele Algebra
3.4 SQL (Deel 1 & 2)

Herh. HC 1

HC 3

3.4 SQL (Structured Query Language)

SQL is de standaard taal voor manipulatie in RDBMSs en heeft een groot aandeel in het success van relationele databanken.
Hoofdstuk 8 – Oefenzitting 2/3

HC3 (Deel 1):
* **Inleiding**
* SQL als vraagtaal (DML): basis

HC4 (Deel 2):
* SQL als vraagtaal (DML): geavanceerd
* SQL als DDL en VDL

Inleiding

Algebraïsche talen
- Steunt op Relationele Algebra
- Operatoren over relaties
- Proceduraal (Hoe)

Calculus talen
- Steunt op Relationele Calculus
- Formele beschrijving: Predikatenlogica
- Declaratief (Wat)

Queryverwerking -en optimalisatie

Declarativiteit

Praktisch

SQL (Structured Query Language)
- Vooral declaraties (Wat)
- Alle bewerkingen op databanken

Inleiding - Voorbeeld

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.

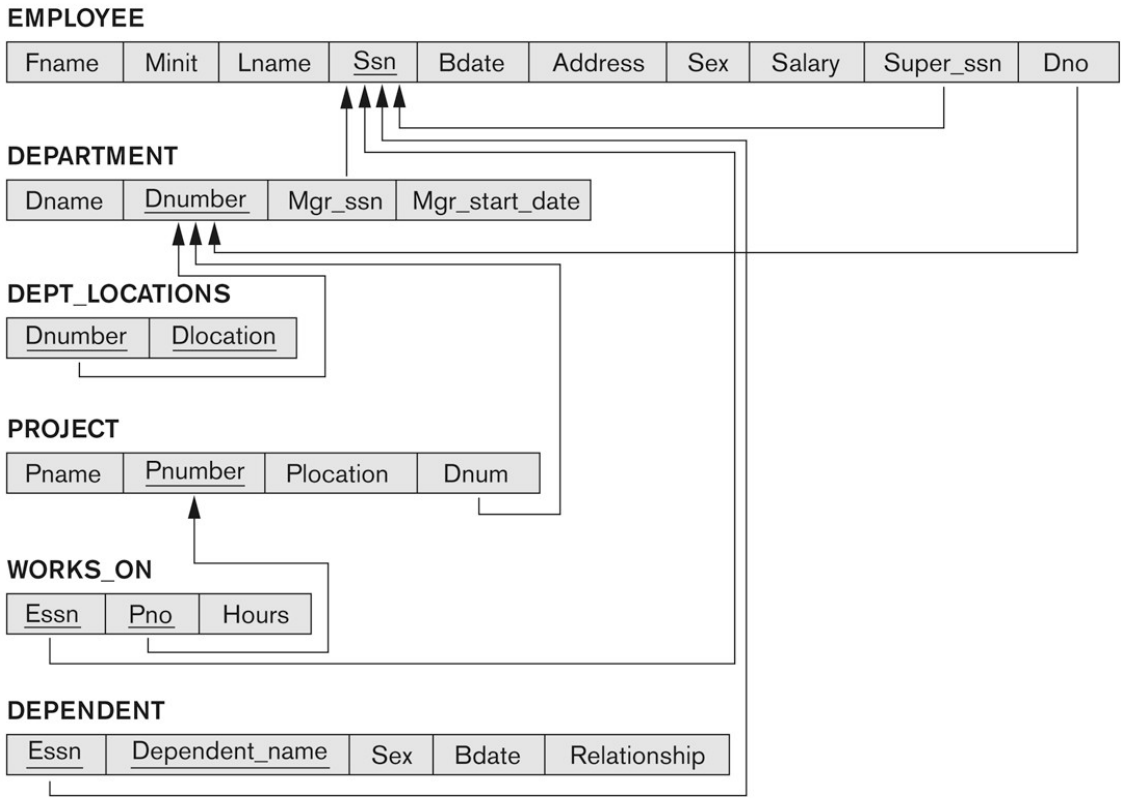


Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

3.4 SQL (Structured Query Language)

SQL is de standaard taal voor manipulatie in RDBMSs en heeft een groot aandeel in het success van relationele databanken.
Hoofdstuk 8 – Oefenzitting 2/3

HC3 (Deel 1):
* Inleiding
* **SQL als vraagtaal (DML): basis**

HC4 (Deel 2):
* SQL als vraagtaal (DML): geavanceerd
* SQL als DDL en VDL

SQL – Gegevens opvragen

Syntax v. **vraag-query**:

```
SELECT    <attributen>
FROM      <tabellen>
WHERE     <condities> ;
```

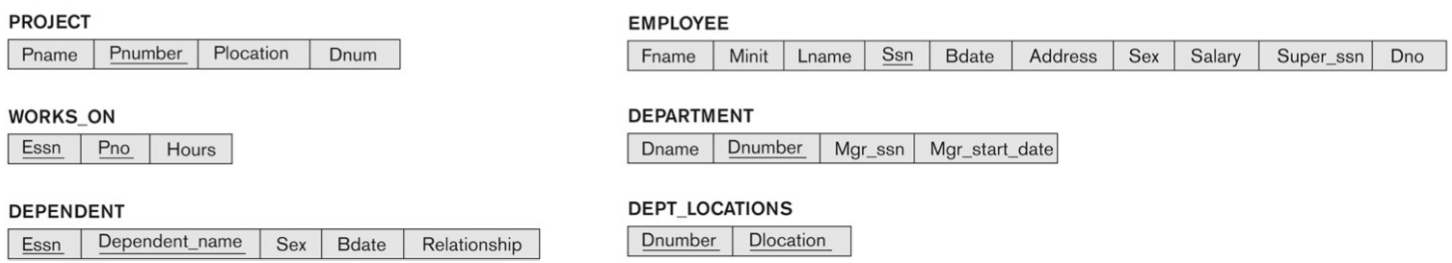
Vergelijking met Relationele Algebra:

```
SELECT    'Projectie'
FROM      'Cartesisch product' (of join)
WHERE     'Selectie'
```

SQL – Joins opbouwen (Voorbeeld)

Geef voor elk project met locatie 'Stafford', het projectnummer, controllerend departementsnummer, en de familienaam van de manager van het departement.

```
SELECT    Pnumber, Dnum, Lname
FROM      PROJECT, DEPARTMENT, EMPLOYEE
WHERE     Plocation='Stafford' AND
          Dnum=Dnumber AND Mgr_ssn=Ssn ;
```



SQL – Impliciete vs. Expliciete JOIN

Impliciet:

```
SELECT    Pname, Essn
FROM      PROJECT, WORKS_ON
WHERE     Dnum=5 AND Pno=Pnumber ;
```

Expliciet:

```
SELECT    Pname, Essn
FROM      (PROJECT JOIN WORKS_ON ON Pno=Pnumber)
WHERE     Dnum=5 ;
```

SQL – NATURAL en OUTER JOINs

NATURAL JOIN:

```
SELECT    Dname, Dlocation
FROM      (DEPARTMENT NATURAL JOIN DEPT_LOCATIONS)
WHERE     Dnum=5 ;
```

OUTER JOIN:

```
SELECT    Fname, Lname
FROM      (EMPLOYEE LEFT OUTER JOIN DEPARTMENT
          ON Ssn=Mgr_ssn)
WHERE     Sex='F' ;
```

SQL – Dubbelzinnige attribuutnamen

Verschillende tabellen kunnen attributen met dezelfde naam bevatten. Dit wordt opgelost via: <tabel>.<attribuut>

Vb. als Dno in EMPLOYEE, Dnumber was:

```
SELECT    Lname, Fname
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dname='Research' AND
          DEPARTMENT.Dnumber = EMPLOYEE.Dnumber ;
```

Exactere notatie mag altijd gebruikt worden in alle attributen van query!

SQL – Aliasen

Wat als we dezelfde tabel willen joinen?

Vb. Geef voor elke werknemer voor -en achternaam alsook de achternaam van zijn overste.

```
SELECT    E.Lname, E.Fname, S.Lname
FROM      EMPLOYEE AS E, EMPLOYEE AS S
WHERE     E.Sup_ssn = S.ssn ;
```

Dus met sleutelwoord AS, maar ook onmiddelijk na tabelnaam, en via oplijsting:

```
SELECT    E.Lname, E.Fname, S.Lname
FROM      EMPLOYEE E S
WHERE     E.Sup_ssn = S.ssn ;
```

Moet als dubbelzinnig, maar mag altijd om overzichtelijk te maken.

SQL – Weglaten WHERE of attributen

Weglaten van WHERE? M.a.w. Geen selectie

```
SELECT      Ssn
FROM        EMPLOYEE ;

SELECT      Ssn, Dname
FROM        EMPLOYEE, DEPARTMENT ;      => Cartesisch Product
```

Weglaten van attributen? M.a.w. Geen projectie

```
SELECT      *
FROM        EMPLOYEE
WHERE       Dno=5 ;
```

=> Alle attributen selecteren!

SQL – Vermijden van dubbelen

In SQL worden dubbelen niet automatisch verwijderd:

```
SELECT      Salary
FROM        EMPLOYEE ;

SELECT      ALL Salary
FROM        EMPLOYEE ;      => Expliciete versie
```

Deze moeten altijd expliciet verwijderd worden:

```
SELECT      DISTINCT Salary
FROM        EMPLOYEE ;
```

SQL – Verzamelingoperaties

UNION, INTERSECT, EXCEPT (= MINUS)

* Tabellen moeten unie-compatibel zijn
  => Zelfde attributen en volgorde (cfr. RA met verschil volgorde)
* Dubbelen worden hier wel automatisch verwijderd (cfr. RA)

Toch met multisets werken? ALL toevoegen:

* UNION ALL
* INTERSECT ALL
* EXCEPT ALL

SQL – Verzamelingoperaties

Vb. Alle projecten waar Smith aan meewerkt of waar hij manager is van het controlerende departement.

```
(   SELECT      DISTINCT Pnumber
    FROM        PROJECT, DEPARTMENT, EMPLOYEE
    WHERE       Dnum=Dnumber AND Mgr_ssn=Ssn AND Lname='Smith' )

UNION

(   SELECT      DISTINCT Pnumber
    FROM        PROJECT, WORKS_ON, EMPLOYEE
    WHERE       Pno=Pnumber And Essn=Ssn And Lname='Smith' ) ;
```

SQL – String operaties

Wanneer we willen testen op patronen in Strings:

* Gebruik LIKE in combinatie met '_' voor één willekeurig teken en '%' voor willekeurige rij tekens.
* Escape (vb. '\%') als karakter voorkomt of speciaal (vb. '\t' of '\n')

Vb. Naam van alle werknemers in Leuven.

```
SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       Address LIKE '%Leuven%' ;
```

Vb. Naam van alle werknemers geboren in januari van de jaren '90.

```
SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       Bdate LIKE '199_-01-__' ;          of ('199_-01%')
```

SQL – Rekenkundige operatoren

Kunnen gebruikt worden op numerieke waarden:

* Als deel van condities in WHERE.
* Als deel van attributen in SELECT om hun uitkomst te beïnvloeden.

Vb. Geef naam en salaris, verhoogd met 10%, van werknemers werkende op projecten startende met de term 'Product'.

```
SELECT    Fname, Lname, 1.1*Salary AS Incr_sal
FROM      EMPLOYEE, WORKS_ON, PROJECT
WHERE     Ssn=Essn And Pno=Pnumber AND Pname LIKE 'Product%' ;
```

SQL – Andere operatoren

BETWEEN

Vb. Werknemers van dept. 5 met salaris tussen 30000 en 40000.

```
SELECT      *
FROM        EMPLOYEE
WHERE       (Salary BETWEEN 30000 AND 40000) AND Dno=5 ;
```

En dan is er nog:

* String concatenatie: ||
* Datum, tijd, tijdsstempel
  - Verhogen, verlagen met interval
  - Verschil geeft interval
* ...

SQL – Ordenen van resultaten

De standaard ordening is stijgend! Om dit te beïnvloeden:

ORDER BY <attributenlijst ([ASC|DESC])>

Vb. Werknemers (namen) en projecten waarop ze werken (namen) per departement, en ik elk departement alfabethisch geordend volgens familienaam en voornaam.

```
SELECT      Dname, Fname, Lname, Pname
FROM        DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE       Dnumber=Dno AND Ssn=Essn AND Pno=Pnumber
ORDER BY    Dname, Lname, Fname ;
```

Opgelet: In volgende geval ORDER BY niet nodig!

```
SELECT      Dname, Lname, Fname, Pname
...
```

SQL – Ordenen van resultaten

Vb. Werknemers (namen) en projecten waarop ze werken (namen) per departement, en ik elk departement alfabethisch geordend volgens familienaam en voornaam.

```
SELECT      Dname, Fname, Lname, Pname
FROM        DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE       Dnumber=Dno AND Ssn=Essn AND Pno=Pnumber
ORDER BY    Dname, Lname, Fname ;
```

Expliciet:

```
ORDER BY    Dname ASC, Lname ASC, Fname ASC ;
```

Departementsnamen in dalende orde:

```
ORDER BY    Dname DESC, Lname ASC, Fname ASC ;
```

3.4 SQL (Structured Query Language)

SQL is de standaard taal voor manipulatie in RDBMSs en heeft een groot aandeel in het success van relationele databanken.
Hoofdstuk 8 – Oefenzitting 2/3

HC3 (Deel 1):
* Inleiding
* SQL als vraagtaal (DML): basis

HC4 (Deel 2):
* **SQL als vraagtaal (DML): geavanceerd**
* SQL als DDL en VDL

SQL – Geneste queries

Doe iets voor tupels in een verzameling.

Vb. Alle projectnummers waar Smith aan meewerkt of waar hij manager is van het controllerende departement:

```
SELECT      DISTINCT Pnumber
FROM        PROJECT
WHERE       Pnumber IN
        (   SELECT   Pnumber
            FROM     PROJECT, DEPARTMENT, EMPLOYEE
            WHERE    Dnum=Dnumber AND Mgr_ssn=ssn AND Lname='Smith' )
                OR Pnumber IN
        (   SELECT   Pno
            FROM     WORKS_ON, EMPLOYEE
            WHERE    Essn=ssn AND Lname='Smith' ) ;
```

SQL – Geneste queries

Doe iets voor tupels in een verzameling.

Vb. Bekom ssn nummers van alle werknemers die op hetzelfde project werken als de werknemer met ssn nummer '123456789' en er dezelfde aantal uren op presteren:

```
SELECT      DISTINCT Essn
FROM        WORKS_ON
WHERE       (Pno,Hours) IN
            (   SELECT      Pno,Hours
                FROM        WORKS_ON
                WHERE       Essn='123456789' ) ;
```

Opgelet: '(' en ')' voor tupel aan te geven, maar bij één attribuut hoeft dit niet

Opgelet: Als de geneste query één tupel met één attibuut teruggeeft kan '=' i.p.v. 'IN'

SQL – Geneste queries

Correlaties: Binnenste SELECT afhankelijk van buitenste

=> De binnenste SELECT moet voor ieder waarde van de buitenste uitgevoerd worden

Vb. Bekom de naam van elke werknemer die een persoon ten laste heeft met dezelfde voornaam en geslacht:

```
SELECT    E.Lname,E.Fname
FROM      EMPLOYEE AS E
WHERE     E.Ssn IN
          (   SELECT      Essn
              FROM        DEPENDENT
              WHERE       E.Fname=Dependent_name AND E.Sex=Sex ) ;
```

SQL – Geneste queries

All, Any voor vergelijking met verzameling waarden

* ALL: Alle waarden voldoen aan de conditie
* ANY: Ten minste één waarde voldoet aan de conditie
* Vergelijksoperatoren: >,<,>=,>,<,=,<>

Vb. Bekom de namen van werknemers die strikt meer verdienen dan de werknemers van departement 5:

```
SELECT      Lname,Fname
FROM        EMPLOYEE
WHERE       Salary > ALL
            (   SELECT      Salary
                FROM        EMPLOYEE
                WHERE       Dno=5 ) ;
```

SQL – Geneste queries

Exists: Bestaat iets al dan niet in een verzameling

Vb. Bekom voor -en achternamen van werknemers met personen ten laste

```
SELECT      Lname,Fname
FROM        EMPLOYEE
WHERE       EXISTS
            (    SELECT  *  FROM DEPENDENT WHERE Ssn=Essn ) ;
```

Vb. Bekom voor -en achternamen van werknemers zonder personen ten laste

```
SELECT      Lname,Fname
FROM        EMPLOYEE
WHERE       NOT EXISTS
            (    SELECT  * FROM DEPENDENT WHERE Ssn=Essn ) ;
```

SQL – Geneste queries

Vb. Bekom de naam van werknemers die op alle projecten gecontrolleerd door departement 5 werken (zoals in boek):

```
SELECT      Lname,Fname
FROM        EMPLOYEE
WHERE       NOT EXISTS
    (
        (   SELECT Pnumber FROM PROJECT WHERE Dno=5 )
        EXCEPT
        (   SELECT Pno FROM WORKS_ON WHERE C.Essn=Ssn )
    ) ;
```

=> Selecteer werknemers zodat "de projecten gecontrolleerd door departement 5" zonder "de projecten waarop de werknemer werkt" een lege verzameling is.

SQL – Geneste queries

Vb. Bekom de naam van werknemers die op alle projecten gecontrolleerd door departement 5 werken (zoals in boek):

```
SELECT      Lname,Fname
FROM        EMPLOYEE
WHERE       NOT EXISTS
    (   SELECT * FROM WORKS_ON B
        WHERE ( B.Pno IN (SELECT Pnumber FROM PROJECT WHERE Dno=5 )
                    AND NOT EXISTS (
                        SELECT * FROM WORKS_ON C
                        WHERE C.Essn=Ssn AND C.Pno=B.Pno ) )
    ) ;
```

=> Selecteer werknemers zodat er geen project (of althans gepreseteerd werk op een project) bestaat gecontrolleerd door departement 5 waarop de werknemer niet werkt.

SQL – Geneste queries

Vb. Bekom de naam van werknemers die op alle projecten gecontrolleerd door departement 5 werken (zoals moet):

```
SELECT      Lname,Fname
FROM        EMPLOYEE
WHERE       NOT EXISTS
    (   SELECT * FROM PROJECT
        WHERE ( Dno=5 AND NOT EXISTS (
                        SELECT * FROM WORKS_ON
                        WHERE Essn=Ssn AND Pno=Pnumber ) )
    ) ;
```

=> Selecteer werknemers zodat er geen project bestaat gecontrolleerd door departement 5 waarop de werknemer niet werkt.

SQL – Aggregaatfuncties

Ingebouwde functies: AVG, SUM, MAX, MIN, COUNT

Vb. Zoek de som, het maximum en minimum en het gemiddelde van alle salarissen van werknemers:

```
SELECT    SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROM      EMPLOYEE ;
```

In **RA**: $\Im_{\text{SUM Salary,MAX Salary,MIN Salary,AVERAGE Salary}}(\text{EMPLOYEE})$

Vb. Zoek de som, het maximum en minimum en het gemiddelde van alle salarissen van werknemers in het 'Research' departement:

```
SELECT    SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROM      (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE     Dname='Research'
```

SQL – Aggregaatfuncties

Ingebouwde functies: AVG, SUM, MAX, MIN, COUNT

Vb. Bepaal het totaal aantal werknemers:

```
SELECT      COUNT(*)
FROM        EMPLOYEE ;
```

Vb. Bepaal het aantal werknemers in het 'Research' departement:

```
SELECT      COUNT(*)
FROM        EMPLOYEE,DEPARTMENT
WHERE       Dno=Dnumber AND Dname='Research'
```

Vb. Tel het aantal verschillende salarisen

```
SELECT      COUNT(DISTINCT(Salary))
FROM        EMPLOYEE ;
```

SQL – Aggregaatfuncties

Ingebouwde functies: AVG, SUM, MAX, MIN, COUNT
Wanneer gebruikt binnen WHERE (=> geneste queries)

Vb. Geef alle werknemers die minstens 2 personen ten laste hebben:

```
SELECT    Lname,Fname
FROM      EMPLOYEE
WHERE     (   SELECT   COUNT(*)
              FROM     DEPENDENT
              WHERE    ssn=Essn ) >= 2 ;
```

SQL – Aggregaatfuncties

Vb. Bekom de naam van werknemers die op alle projecten gecontrolleerd door departement 5 werken:

```
SELECT      Lname,Fname
FROM        EMPLOYEE
WHERE       (    SELECT COUNT(*) FROM PROJECT WHERE Dnum=5 ) =
            (    SELECT COUNT(hours)  FROM WORKS_ON, PROJECT
                 WHERE Pnum=Pnumber AND Dnum=5 AND Essn=Ssn ) ;
```

=> Selecteer werknemers zodat het aantal projecten gecontrolleerd door departement 5 gelijk is aan het aantal projecten gecontrolleerd door departement 5 waarop de werknemer werkt.

Opgelet: In aggregaatfuncties worden NULL waarden in het algemeen niet beschouwd als ze over specifieke attributen gespecifieerd zijn, anders wel!

SQL – Aggregaatfuncties & Groepering

Groeperingen via GROUP BY

=> Aggregaatfuncties worden voor elke groep toegepast

Vb. Geef voor elk departement, het nummer, het aantal werknemers en het gemiddelde salaris:

```
SELECT      Dno, COUNT(*), AVG(Salary)
FROM        EMPLOYEE
GROUP BY    Dno ;
```

RA: $_{\text{Dno}}\mathfrak{I}_{\text{COUNT Ssn,AVERAGE Salary}}(\text{EMPLOYEE})$

Opgelet: Groepering beschouwen NULL waarden als bron voor apparte groep.

SQL – Aggregaatfuncties & Groepering

Condities per groepering: HAVING

Vb. Geef voor elk project, waar strikt meer dan twee mensen op werken, de naam, het nummer, en het aantal werknemers die erop werken:

```
SELECT      Pname, Pnumber, COUNT(*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber
HAVING      COUNT(*) > 2 ;
```

SQL – Aggregaatfuncties & Groepering

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

Na WHERE

Pname	Pnumber	...	Essn	Pno	Hours	
ProductX	1		123456789	1	32.5	
ProductX	1		453453453	1	20.0	
ProductY	2		123456789	2	7.5	Na HAVING
ProductY	2		453453453	2	20.0	
ProductY	2		333445555	2	10.0	
ProductZ	3		666884444	3	40.0	
ProductZ	3		333445555	3	10.0	
Computerization	10	...	333445555	10	10.0	Na HAVING
Computerization	10		999887777	10	10.0	
Computerization	10		987987987	10	35.0	
Reorganization	20		333445555	20	10.0	Na HAVING
Reorganization	20		987654321	20	15.0	
Reorganization	20		888665555	20	NULL	
Newbenefits	30		987987987	30	5.0	Na HAVING
Newbenefits	30		987654321	30	20.0	
Newbenefits	30		999887777	30	30.0	

```
SELECT      Pname, Pnumber, COUNT(*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber
HAVING      COUNT(*) > 2 ;
```

RESULTAAT

Pname	Pnumber	Count(*)
ProductY	2	3
Computerization	10	3
Reorganization	20	3
Newbenefits	30	3

SQL – Gegevens opvragen

Algemene vorm:

```
SELECT      <attributenlijst>                (+ Rename, Veralg. projectie, Aggr.func.)
FROM        <tabellenlijst>                  (+ Rename, Expliciet en speciale joins)
WHERE       <condities op tupels>
GROUP BY    <groeperingsattributenlijst>
HAVING      <condities op groeperingen>
ORDER BY    <attributen voor ordening na selectie>
```

Deze vorm mag genest worden in de WHERE. ORDER BY heeft dan geen zin.
Meerdere nestings zijn mogelijk.
Verzamelingoperaties connecteren deze vorm wanneer hun resultaten unie-compatibel zijn. Het nesten vergt ook unie-compatibiliteit.

3.4 SQL (Structured Query Language)

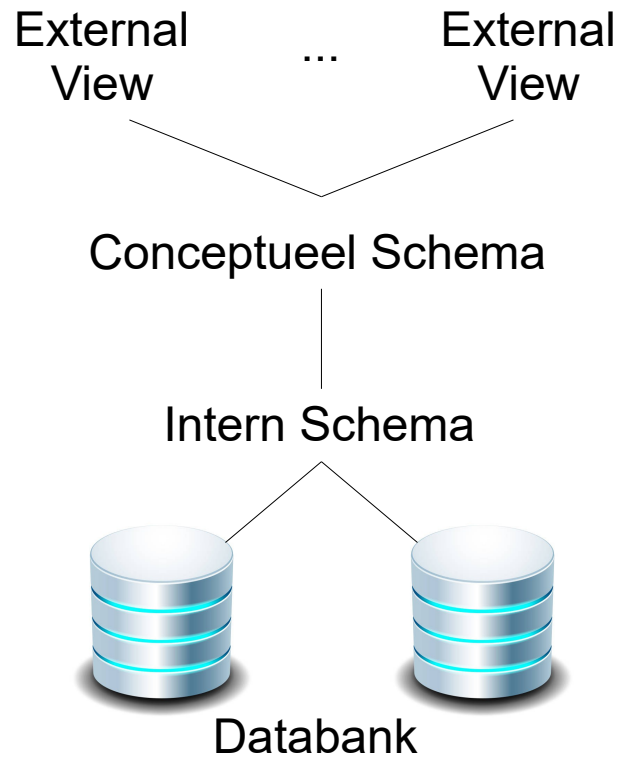
SQL is de standaard taal voor manipulatie in RDBMSs en heeft een groot aandeel in het success van relationele databanken.
Hoofdstuk 8 – Oefenzitting 2/3

HC3 (Deel 1):
* **Inleiding**
* SQL als vraagtaal (DML): basis

HC4 (Deel 2):
* SQL als vraagtaal (DML): geavanceerd
* SQL als DDL en VDL

3-S DBMS arch. & Talen

* 3-schema DBMS architectuur:
  - **SDL**: Storage Definition Language (intern)
  - **DDL**: Data Definition Language (conceptueel)
  - **VDL**: View Definition Language (extern)
* Daarnaast:
  - **DML**: Data Manipulation Language
  - Vb. ophalen, aanpassen, ...
* Relational DBMS:
  - **SQL** (= **DDL**+**VDL**+DML)
  - SQL is een declaratieve taal



(ANSI/SPARC, 1975)

Relationeel model

Relaties met attributen (domeinen) en hun tupels.
Sleutels: super, kandidaat, primair -en verwijssleutels
Restricties/Beperkingen/Constraints:

* Domeinrestricties
* Sleutelrestricties
* Integriteitsrestricties
  - Statische regels (toestand)
  - Dynamische regels (toestandsovergang)
* Authorisatierestricties
* ...

3.4 SQL (Structured Query Language)

SQL is de standaard taal voor manipulatie in RDBMSs en heeft een groot aandeel in het success van relationele databanken.
Hoofdstuk 8 – Oefenzitting 2/3

HC3 (Deel 1):
* Inleiding
* SQL als vraagtaal (DML): basis

HC4 (Deel 2):
* **SQL als vraagtaal (DML): geavanceerd**
* SQL als DDL en VDL

SQL – Gegevens aanpassen

Toevoegen (INSERT), verwijderen (DELETE) en wijzigen (UPDATE):

* Expliciet: Eigen gekozen volgorde.
  - *NULLs en DEFAULTs mogen weggelaten worden (zie later)*

```
INSERT INTO EMPLOYEE (Fname, Lname, Ssn)
VALUES ('Richard', 'Marini', '653298653') ;
```

* Impliciet: Volgorde volgens dat van schema.

```
INSERT INTO EMPLOYEE
VALUES ('Richard', 'K', 'Marini', '653298653', ‘1962-12-30',
'98 Oak Forest, Katy', TX', 'M', 37000, '987654321', 4) ;
```

SQL – Gegevens aanpassen

Toevoegen (INSERT), **verwijderen (DELETE)** en wijzigen (UPDATE):

* Alle tupels die aan conditie voldoen verwijderen.

```
DELETE FROM   EMPLOYEE
WHERE         Lname='Brown' ;

DELETE FROM   EMPLOYEE
WHERE         Ssn='123456789' ;

DELETE FROM   EMPLOYEE
WHERE         Dno IN
    ( SELECT Dnumber FROM DEPARTMENT WHERE Dname='Research' ) ;
```

* Geen WHERE (= WHERE TRUE): alles wordt verwijderd. Gevolg: lege tabel.

```
DELETE FROM   EMPLOYEE;
```

SQL – Gegevens aanpassen

Toevoegen (INSERT), verwijderen (DELETE) en **wijzigen (UPDATE)**:

* Alle tupels die aan conditie voldoen wijzigen.
* Met SET nieuwe waardes toekennen.

```
UPDATE   PROJECT
SET      Plocation='Bellaire', Dnum=5
WHERE    Pnumber=10 ;

UPDATE   EMPLOYEE
SET      Salary=Salary*1.5
WHERE    Dno IN
     ( SELECT Dnumber FROM DEPARTMENT WHERE Dname='Research' ) ;
```

3.4 SQL (Structured Query Language)

SQL is de standaard taal voor manipulatie in RDBMSs en heeft een groot aandeel in het success van relationele databanken.
Hoofdstuk 8 – Oefenzitting 2/3

HC3 (Deel 1):
* Inleiding
* SQL als vraagtaal (DML): basis

HC4 (Deel 2):
* SQL als vraagtaal (DML): geavanceerd
* **SQL als DDL** en VDL

SQL – Catalogus

= Verzameling gegevensbankschema's in "SQL omgeving"
=> Wordt vaak een "databank" genoemd op zich,
zelfs bij meerdere catalogi in éénzelfde RDBMS installatie
en bij gebruik zegt men toegang te krijgen m.b.t. die "databank"

* Bevat speciaal schema: INFORMATION_SCHEMA
* Schema's bevatten relaties (tabellen)
met een schema de logische opdeling m.b.t. databankapplicaties
* Integriteitsrestricties kunnen opgelegd worden over schema's heen,
d.w.z. tussen relaties (tabellen) in dezelfde catalogus
* Schema's binnen eenzelfde catalogus kunnen elementen delen,
bv. domeindefinities

SQL – Schema

Componenten van een schema:

* Schemanaam + Eigenaar
* Beschrijving van elementen:
    - Tabellen
    - Restricties
    - Domeinen
    - Autorisaties
    - ...
    - Views

Aanmaak schema:

CREATE SCHEMA COMPANY **AUTHORIZATION** Jsmith ;

SQL – Tabel

Componenten van een tabel:

* Behoren tot schema (Er is een concept van een default schema)
* Tabelnaam (+ Eigenaar)
* Attributen: Naam, SQL-gegevenstype, attribuutrestricties
* Tabelrestricties

Aanmaak tabel:

CREATE TABLE [<schema name>.]<table name>
({ <column name> <column type> [<Attribute constraint>] }
{ <table constraint> } *
) ;

SQL – Tabel

Definitie attributen (domeinen) kan rechtstreeks met SQL-gegevenstype of via domeindefinitie voor herbruik:

```
CREATE DOMAIN SSN_TYPE AS CHAR(9) ;
```

* En met attribuutrestricties (zie later):

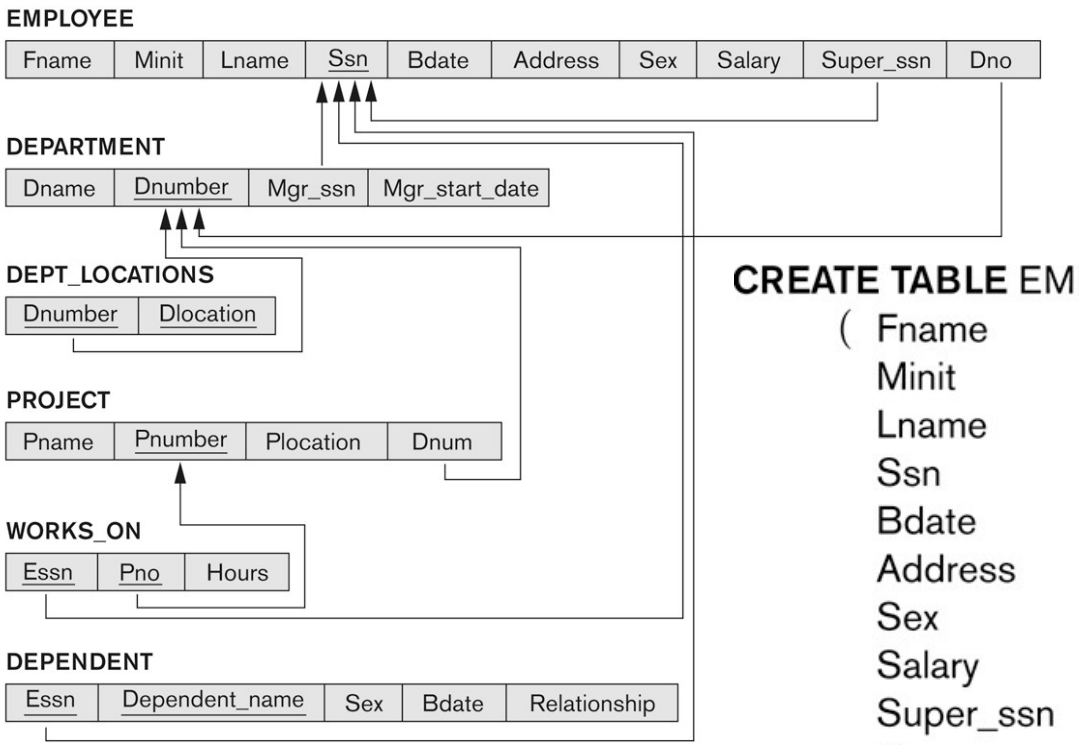
```
CREATE DOMAIN D_NUM AS INTEGER
CHECK (D_NUM > 0 AND D_NUM < 21)
```

Sleutelrestricties op tabel:

* Primaire sleutel: PRIMARY KEY <attribute list>
* Alternatieve sleutel: UNIQUE <attribute list>
* Verwijssleutel: FOREIGN KEY <attrlist> REFERENCES <table>.<attrlist>

SQL – Tabel

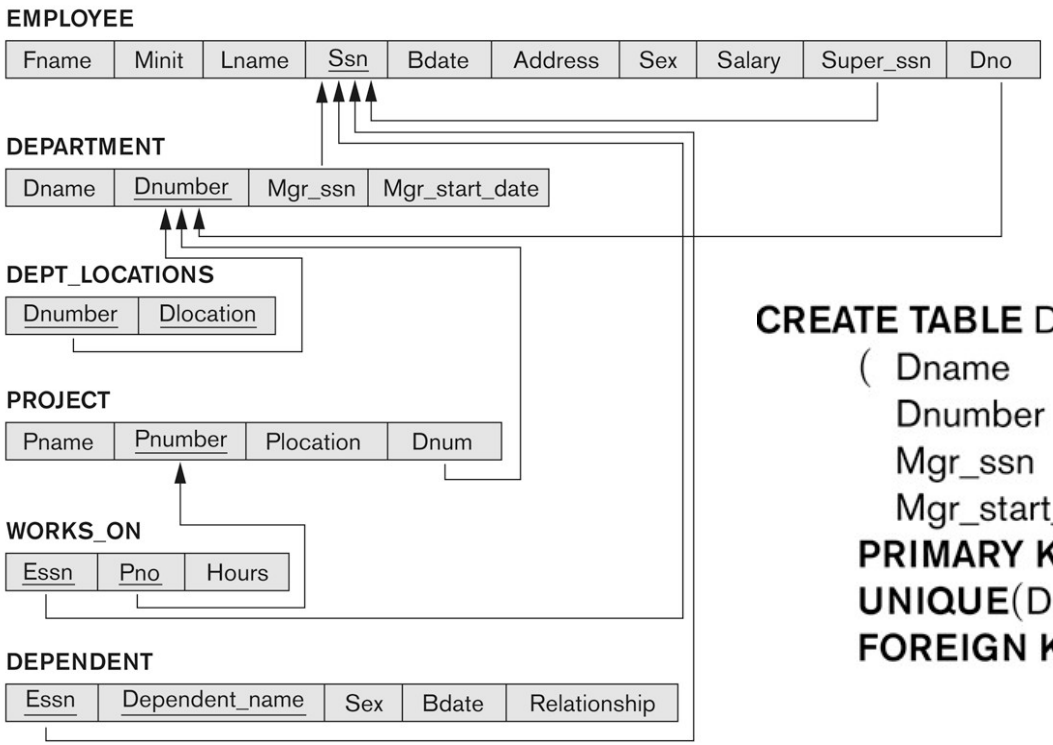
Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.




```
CREATE TABLE EMPLOYEE
      ( Fname          VARCHAR(15)      NOT NULL,
        Minit          CHAR,
        Lname          VARCHAR(15)      NOT NULL,
        Ssn            CHAR(9)          NOT NULL,
        Bdate          DATE,
        Address        VARCHAR(30),
        Sex            CHAR,
        Salary         DECIMAL(10,2),
        Super_ssn      CHAR(9),
        Dno            INT              NOT NULL,
      PRIMARY KEY (Ssn),
      FOREIGN KEY(Super_ssn) REFERENCES EMPLOYEE(Ssn),
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber) );
```

SQL – Tabel

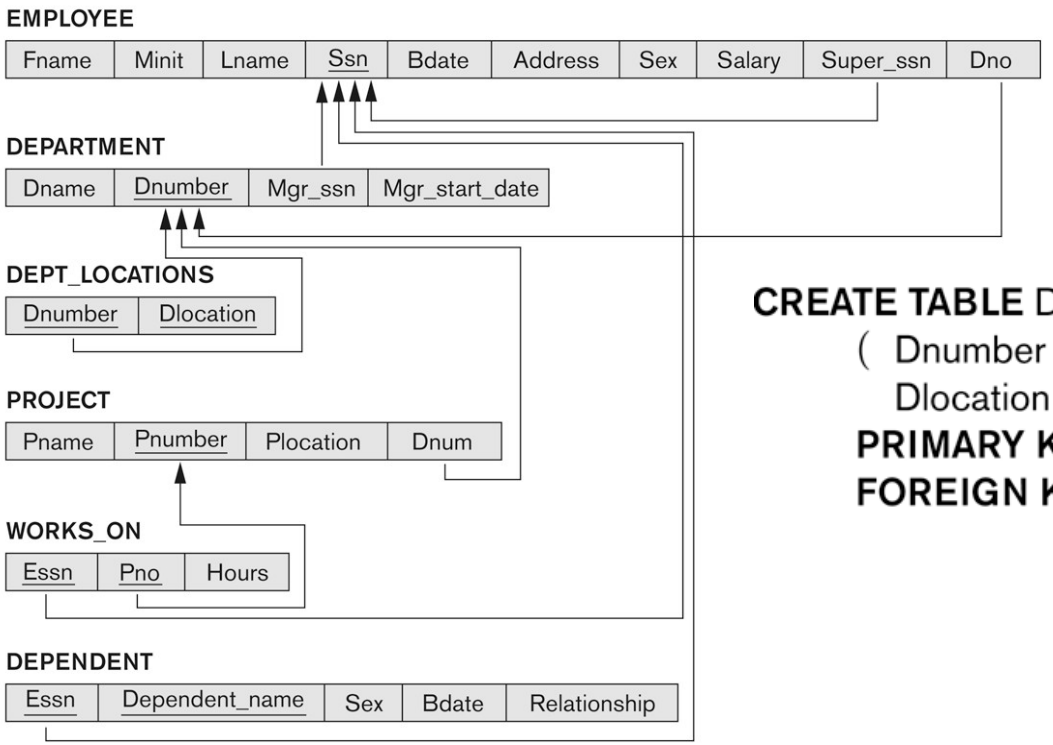
Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.




```
CREATE TABLE DEPARTMENT
    ( Dname          VARCHAR(15)    NOT NULL,
      Dnumber        INT            NOT NULL,
      Mgr_ssn        CHAR(9)        NOT NULL,
      Mgr_start_date DATE,
    PRIMARY KEY(Dnumber),
    UNIQUE(Dname),
    FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

SQL – Tabel

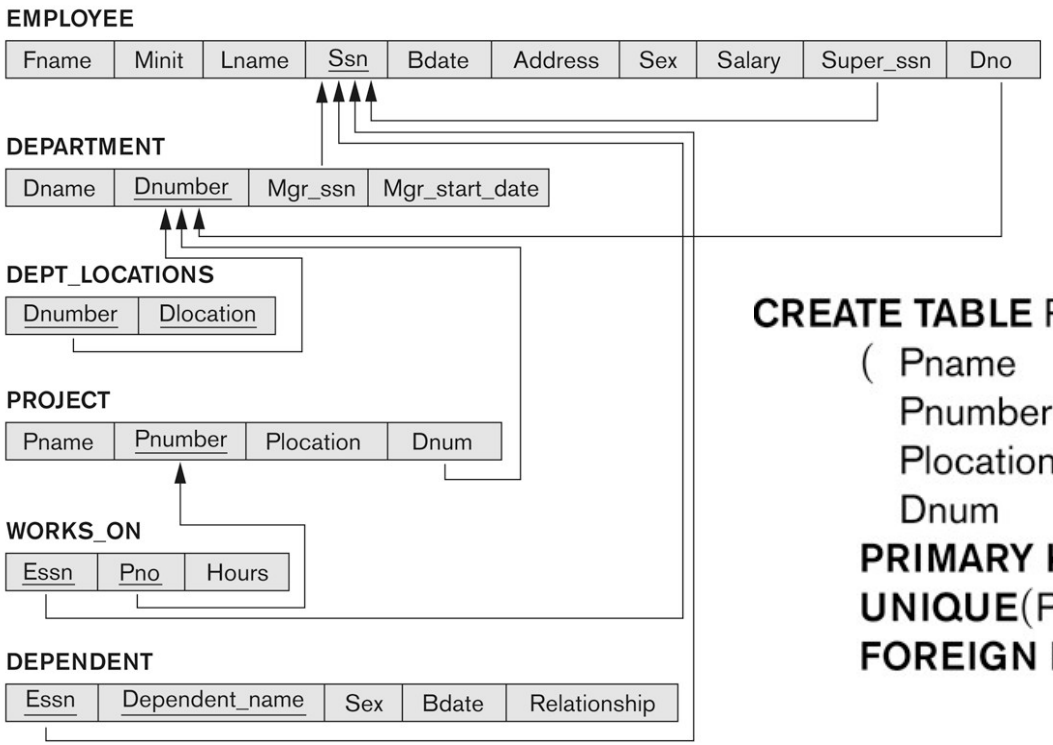
Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.




```
CREATE TABLE DEPT_LOCATIONS
      ( Dnumber          INT             NOT NULL,
        Dlocation        VARCHAR(15)     NOT NULL,
      PRIMARY KEY(Dnumber, Dlocation),
      FOREIGN KEY(Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

SQL – Tabel

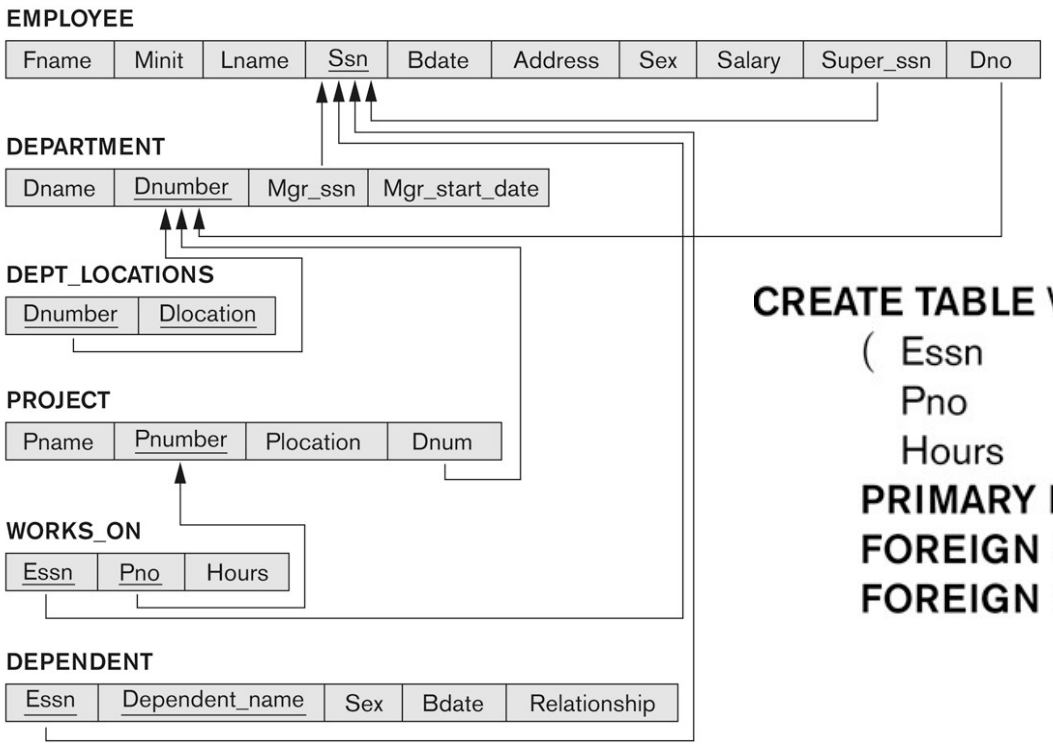
Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.




```
CREATE TABLE PROJECT
    ( Pname          VARCHAR(15)     NOT NULL,
      Pnumber        INT             NOT NULL,
      Plocation      VARCHAR(15),
      Dnum           INT             NOT NULL,
    PRIMARY KEY(Pnumber),
    UNIQUE(Pname),
    FOREIGN KEY(Dnum) REFERENCES DEPARTMENT(Dnumber) );
```

SQL – Tabel

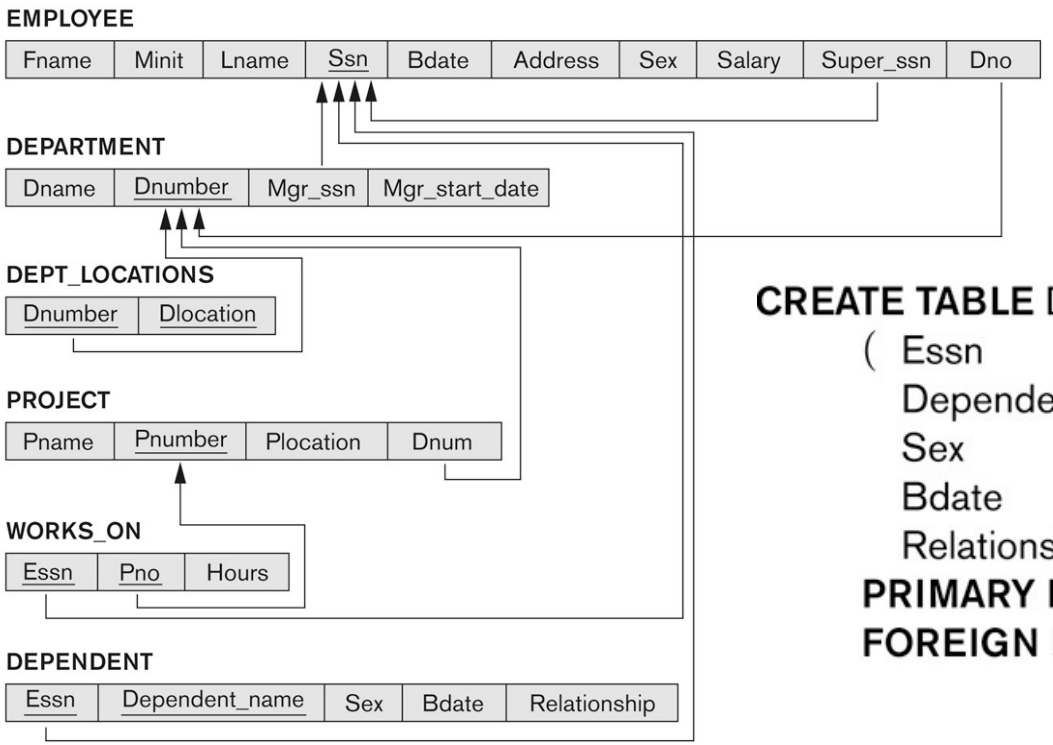
Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.




```
CREATE TABLE WORKS_ON
    ( Essn          CHAR(9)       NOT NULL,
      Pno           INT           NOT NULL,
      Hours         DECIMAL(3,1)  NOT NULL,
    PRIMARY KEY(Essn, Pno),
    FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY(Pno)  REFERENCES PROJECT(Pnumber) );
```

SQL – Tabel

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.




```
CREATE TABLE DEPENDENT
    ( Essn                  CHAR(9)          NOT NULL,
      Dependent_name        VARCHAR(15)      NOT NULL,
      Sex                   CHAR,
      Bdate                 DATE,
      Relationship          VARCHAR(8),
    PRIMARY KEY(Essn, Dependent_name),
    FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn) );
```

SQL – Domeinen

Numerisch:
INT (or INTEGER), SMALLINT, *(BIGINT)*, FLOAT (or REAL), DOUBLE PRECISION *(or DOUBLE/BIGFLOAT)*, DECIMAL(i,j) or DEC/NUMERIC met i de precisie en j de schaal

* vaak zijn de byte representaties machine specifiek (soms ook BYTE voor 1byte)
* Meestal: INT en FLOAT (4byte), SMALLINT (2byte), BIGINT en DOUBLE (8byte)

Karakters:
CHAR(n), VARCHAR(n), CHAR, VARCHAR, *(CLOB/TEXT)* met alternatieven voor VARCHAR zijnde CHAR VARYING or CHARACTER VARYING

* CHAR voor vaste lengte (wel padding: spaties)
* VARCHAR voor variabele lengte (extra bytes voor lengte, als langer: truncate)
* n stelt bytes voor, niet aantal karakters, is enkel hetzelfde in 'Latin-1' (≠ E-ASCII)

Bit-string:
BIT(n) (geen padding), BIT VARYING(n) (geen truncate), BIT, BIT VARYING, *(BLOB)*

* *Als literal: B'100101' => String van bits met prefix 'B'*

SQL – Domeinen

Booleaans: BOOLEAN
* TRUE (1), FALSE (0), UNKNOWN (NULL)

Date & Time: DATE, TIME, TIME(n)
* DATE (YEAR, MONTH, DAY) vorm 'YYYY-MM-DD', als literal DATE'2023-10-19'
* TIME (HOUR, MINUTE, SECOND) vorm 'HH:MM:SS', als literal TIME'11:44:17'
* TIME(i) voor tijd in fracties van seconden (met i aantal getallen na de komma)
* TIME WITH TIMEZONE (+ 6 karakters voor '+'/'-' en HH:MM)
* Default: lokale tijdszone

Timestamp: TIMESTAMP
* DATE+TIME+FRACTION(+TIMEZONE)
* Als literal: TIMESTAMP'2023-10-19 11:44:17 648302'
* Unix time = ms sinds epoch: 1970-01-01 00:00:00+00:00

Interval: INTERVAL
* Om DATE, TIME, TIMESTAMP te de/incrementeren

SQL – Referentiële integriteit

Verwijssleutel kan enkel naar bestaande primaire sleutel verwijzen of mag NULL zijn tenzij attribuutrestrictie.

Vb. Dno in EMPLOYEE verwijst naar bestaand Dnumber in DEPARTMENT

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

SQL – Attribuutrestricties & Referentiële integriteitsrestricties

Attribuutrestricties:
* NOT NULL: Automatisch voor
  Primaire Sleutel en UNIQUE
* DEFAULT <waarde>
* CHECK <condities>
  - Dnumber INT NOT NULL CHECK
    (Dnumber > 0 AND Dnumber < 21)

Ref. integriteitsrestricties:
* Wanneer: ON DELETE / ON UPDATE
* Wat: SET NULL / CASCADE / SET DEFAULT

```
CREATE TABLE EMPLOYEE
    ( ...,
      Dno          INT          NOT NULL          DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY(Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY(Super_ssn) REFERENCES EMPLOYEE(Ssn)
                   ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                   ON DELETE SET DEFAULT       ON UPDATE CASCADE );
CREATE TABLE DEPARTMENT
    ( ...,
      Mgr_ssn      CHAR(9)      NOT NULL          DEFAULT '888665555',
      ...,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE(Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                   ON DELETE SET DEFAULT       ON UPDATE CASCADE );
CREATE TABLE DEPT_LOCATIONS
    ( ...,
    PRIMARY KEY(Dnumber, Dlocation),
    FOREIGN KEY(Dnumber) REFERENCES DEPARTMENT(Dnumber)
                   ON DELETE CASCADE           ON UPDATE CASCADE );
```

SQL – Referentiële integriteitsrestricties

Vb. Bij wijzigen Dnumber van Research departement:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	~~5~~ 6
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	~~5~~ 6
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	~~5~~ 6
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	~~5~~ 6
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	~~5~~ 6	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

SQL – Referentiële integriteitsrestricties

Vb. Bij verwijderen van Research departement:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	~~5~~
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	~~5~~
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	~~5~~
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	~~5~~
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

1
1
1
1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
~~Research~~	~~5~~	~~333445555~~	~~1988-05-22~~
Administration	4	987654321	1995-01-01
~~Headquarters~~	1	888665555	1981-06-19

SQL – Schema aanpassen

* Verwijderen van schema:
  DROP SCHEMA <schema name> ;
* Verwijderen van tabel:
  DROP TABLE [<schema name>.]<table name>;
  => Tabel en waarden worden verwijderd!
* CASCADE vs. RESTRICT (default)
  - **DROP TABLE** DEPENDENT **CASCADE** ;
    => altijd verwijderd alsook refererende constraints
  - **DROP TABLE** DEPENDENT **RESTRICT** ;
    => enkel wanneer de tabel geen referrerende constraints heeft, noch views
* DROP ook voor verwijderen van restricties en domeinen
* ALTER voor aanpassingen: Namen, attributen, beperkingen, etc.

3.4 SQL (Structured Query Language)

SQL is de standaard taal voor manipulatie in RDBMSs en heeft een groot aandeel in het success van relationele databanken.
Hoofdstuk 8 – Oefenzitting 2/3

HC3 (Deel 1):
* Inleiding
* SQL als vraagtaal (DML): basis

HC4 (Deel 2):
* SQL als vraagtaal (DML): geavanceerd
* **SQL als** DDL en **VDL**

SQL – VIEWs

Een View is een afgeleide relatie.

* Tupels worden niet expliciet opgeslagen.
* Tupels worden afgeleid/berekend uit relaties (=definiërende relaties).
* Een view is virtueel en wordt niet gematerialiseerd.
* Een view wordt bevraagd met SELECT zoals een tabel.

Een VIEW aanmaken:

CREATE VIEW <view name> [<attribute list>]
AS <query> ;

Een VIEW verwijderen (geen verwijdering van tupels:

DROP VIEW <view name>

SQL – VIEWs

Voorbeelden van aanmaken en verwijderen van view:

```
CREATE VIEW WORKS_ON1
AS      SELECT      Fname, Lname, Pname, Hours
        FROM        EMPLOYEE, PROJECT, WORKS_ON
        WHERE       Ssn = Essn AND Pno = Pnumber ;

DROP VIEW WORKS_ON1 ;

CREATE VIEW DEPT_INFO(Dept_name,No_of_emps,Total_sal)
AS      SELECT      Dname, COUNT(*), SUM(Salary)
        FROM        DEPARTMENT, EMPLOYEE
        WHERE       Dnumber = Dno
        GROUP BY    Dname ;
```

SQL – VIEWs

Queries op VIEWs geschieden zoals op een tabel.
vb. Aanmaak WORKS_ON1 view:

```
CREATE VIEW WORKS_ON1
AS    SELECT    Fname, Lname, Pname, Hours
      FROM      EMPLOYEE, PROJECT, WORKS_ON
      WHERE     Ssn = Essn AND Pno = Pnumber ;
```

Vb. Query op WORKS_ON1 view:

```
SELECT    Fname, Lname
FROM      WORKS_ON1
WHERE     Pname = 'ProductX' ;
```

SQL – VIEWs

Benaderingen voor onderliggende implementatie:

1. Query modification:
   - Query op view omvormen tot query op definiërende tabellen
   - Mogelijks complexe en tijdsintensieve queries die herhaalt worden
2. View materialization:
   - Tijdelijke aanmaak van een tabel (view) wanneer voor het eerst in een query gebruikt
   - Vereist incrementele updates op gematerializeerde tabel wanneer definiërende tabellen wijzigen
   - Gematerializeerde tabel (view) verwijderen wanneer in ongebruik

SQL – VIEWs

Voorbeeld "query modification":

De query

```
SELECT   Fname, Lname
FROM     WORKS_ON1
WHERE    Pname = ‘ProductX’;
```

op view WORKS_ON1 wordt automatisch omgevormd tot de query

```
SELECT   Fname, Lname
FROM     EMPLOYEE, PROJECT, WORKS_ON
WHERE    Ssn = Essn AND Pno = Pnumber AND Pname = ‘ProductX’ ;
```

SQL – Wijzigen van VIEWs

Aangezien wijziging VIEW moet doorgegeven worden naar de definiërende relaties: Niet altijd eenduidig mogelijk!

* Bij 1 definiërende relatie => Wijziging view is wijziging basisrelatie.
* Bij join van relaties => Welke relatie aanpassen?

Vb. Pname v. John Smith in WORKS_ON1 v. 'ProductX' n. 'ProductY'

```
UPDATE      WORKS_ON1
SET         Pname = 'ProductY'
WHERE       Lname = 'Smith' AND Fname = 'John' AND Pname = 'ProductX' ;
```

=> Ambigu: Project van naam veranderd of werkt John op ander project?

SQL – Wijzigen van VIEWs

=> Verander projectnaam

```
UPDATE      PROJECT
SET         Pname = 'ProductY'
WHERE       Pname = 'ProductX' ;
```

=> Verander projecten waarop John werkt:

```
UPDATE      WORKS_ON
SET         Pno = ( SELECT Pnumber FROM PROJECT
                    WHERE Name = 'ProductY')
WHERE       Essn = ( SELECT ssn FROM EMPLOYEE
                    WHERE Lname = 'Smith' AND Fname = 'John' )
            AND Pno IN ( SELECT Pnumber FROM PROJECT
                    WHERE Pname = 'ProjectX' );
```

SQL – Wijzigen van VIEWs

In het algemeen:

* Een view afgeleid uit 1 tabel kan aangepast worden als de view een primaire sleutel of kandidaatsleutel van die tabel bevat
  => Tupel dat gewijzigd moet worden is éénduidig bepaald
* een view afgeleid uit meerdere tabellen is meestal niet aanpasbaar
  => Er moet een procedure zijn om te kiezen
  => Gebruiker kiest (of DBMS kiest meest waarschijnlijke)
* Resultaten van aggregaatfuncties kunnen niet aangepast worden
  => Volgende voorbeeld heeft geen zin/slaagt op niets:

```
UPDATE   DEPTS_INFO
SET      TOTAL_SAL=100000
WHERE    DNAME='Research'
```

SQL – Wat meer?

* Permissies: Welke gebruiker mag wat doen?
* Verbinding met programmeertalen
* Nog niet vermeldde schemawijzigingen
* Assertions: inter-relationele -en dynamische restricties
* Triggers: Assertions met bijhorende acties
* Concepten i.v.m concurrentiecontrole en herstel