## Inleidende Begrippen

## Integriteitsregels

`Key Constraint` ..... PK moet uniek zijn, en uniek blijven
`Entity Integrity Constraint` ...PK moet geldige waarde hebben
`Referential Integrity Constraint` .De populatie van FK en PK moet overeenkomen

## Bewerkingen met relaties

`UNION` .....RELATIE3 = RELATIE1 ∪ RELATIE2
`INTERSECTION` ..... RELATIE3 = RELATIE1 ∩ RELATIE2
`VERSCHIL` ..... RELATIE3 = RELATIE1 \ RELATIE2
`PRODUCT` .....RELATIE3 = RELATIE1 Π RELATIE2

## Eigenschappen van SQL

`Declaratief` .....Instructies wat je wil bereiken
`Interactief` ..... Instrcuties en krijg gelijk antwoord
`Embedded` .....Kan het in een programmeertaal gebruiken

## Subcategorieën

`DQL` ..... *SELECT*
`DML` .....*UPDATE*, *DELETE*, *INSERT*
`DDL` .....*CREATE*, *ALTER*, *DROP*
`DCL` ..... *GRANT*, *REVOKE*
`TC` .....*COMMIT*, *ROLLBACK*, *SAVEPOINT*

## Multipliciteiten

0..1 ..... Geen of een instantie
1 ..... Exact een instantie
∗,0..∗ ..... Geen of meerdere instanties
1..∗ ..... Een of meerdere instanties

## Select & Fetch

| Relationship | Example | Left | Right | Narrative |
|---|---|---|---|---|
| One-to-one | person ⟷ birth certificate | 1 | 1 | A person must have its own birth certificate |
| One-to-one *(optional on one side)* | person ⟷ driving license | 1 | 0..1 or ? | A person may have a driving license |
| Many-to-one | person ⟷ birthplace | 1..* or + | 1 | Many people can be born at the same place |
| Many-to-many *(optional on both sides)* | person ⟷ book | 0..* or * | 0..* or * | A person may own books |
| One-to-many | order ⟷ line item | 1 | 1..* or + | An order contains at least one item |
| Many-to-many | course ⟷ student | 1..* or + | 1..* or + | Students follow various courses |

```
SELECT [DISTINCT] select lijst
FROM tabelnaam
[WHERE conditie]
[ORDER BY clausule]
[OFFSET offset ROWS]
[FETCH row limiting clause]
```

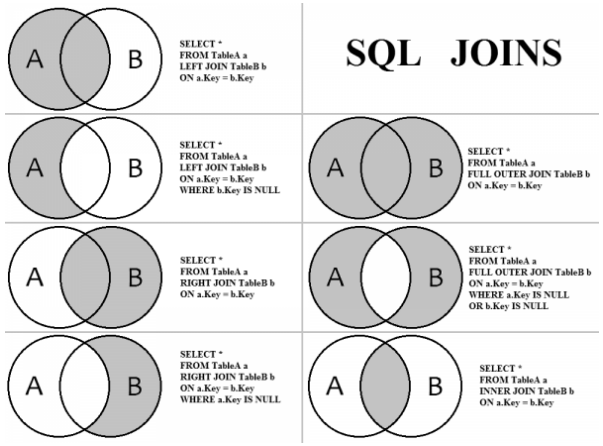
```
SELECT * FROM employees
WHERE dob
BETWEEN '1-JAN-1980'
AND '31-DEC-1989';
```

```
FETCH WITH TIES -- Wanneer je meerdere rijen wilt retourneren die gelijk staan
NULLS FIRST -- Wanneer je NULL waarde als eerst wil tonen
```

```
[OFFSET offset ROWS]
FETCH NEXT [row_count]
ROWS [ONLY | WITH TIES]
```

## Joins

"*Non*-Equi-joins, Auto-joins, Date&Time Functies"




## Gewone JOIN

Resulteert enkel gematchte records.

```
SELECT * FROM foo f
[INNER] JOIN bar b -- INNER is default
ON f.id = b.id
```

```
SELECT * FROM foo
JOIN bar USING(id) -- kolomnamen hebben dezelfde naam
```

## Non-equi JOIN

```
SELECT first_name, last_name, schaalnr
FROM employees
JOIN pay_grades ON (salary/12 BETWEEN lower_limit AND upper_limit);

-- De join is niet gebaseerd op een gelijkheid tussen attribuutwaarden
```

## Outer JOIN

Resulteert in alle gematchte records + null lijnen.

## Auto JOIN

Een gewone join, waarbij de tafel met zichzelf samenvoegt. "Oftewel een recursieve relatie""

```
SELECT *
FROM employees e
JOIN employees mgr
ON (e.manager_id=mgr.employee_id);
```

## Impliciete conversies

`cast(foo AS INT)` ..... Omzetten van tekst naar numeriek
`to_char(123, '999')` ..Omzetten van numeriek naar tekst
`to_char(dob,'TMMonth')` ..... vertaalde datum naar tekst

## Afronden van nummers

`round(15251.675)` ..... afgerond op het geheel
`round(15251.675,1)` ..... afgerond op 1 decimaal
`round(15251.675,-1)` ..... afgerond tot een tiental

## Afkappen van nummers

`trunc(15251.675)` ..... afgekapt op 1 geheel
`trunc(15251.675,1)` ..... afgekapt op 1 decimaal
`trunc(15251.675,-1)` ..... afgekapt op 1 tiental

## Datum functies

`current_date` ..... huidige datum
`date_part` ..... stuk uit datum halen
`date_trunc` ..... datum afhakken

## Leeftijd

```
SELECT age(dob) "age" FROM employees;
SELECT date_part('year',(age(dob))) FROM employees;
```

## Rekenen met datum & tijd

```
SELECT date '2021-09-28' + 7;
SELECT date '2021-09-28' + interval '10 hour';
SELECT time '01:00' + interval '3 hours';
SELECT interval '1 hour' / 1.5;
```

## Group By & Outer Join

```
-- Geef het hoogste salaris per afdeling:

SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
ORDER BY department_id;


-- Geef het hoogste salaris per afdeling,
-- waarvan de waarden kleiner zijn dan 45000:

SELECT department\_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary) <45000
```

## Analystische functies

- Geeft 1 RIJ RESULTAAT per groep.
- Staan altijd in de $SELECT$ of in de $HAVING$ clausule.
- Staan NOOIT in de $WHERE$-clausule!
- Houden geen rekening met $NULL$ waarden.

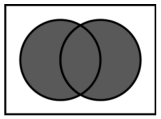
`AVG` ..... Bereken gemiddelde
`SUM` ..... Bereken de som
`MIN` ..... Laagste getal
`MAX` ..... Hoogste getal
`COUNT` ..... Telt attribuutwaarden

Volgorde van uitvoering:
$SELECT -> FROM -> WHERE -> GROUPBY -> HAVING -> ORDERBY$

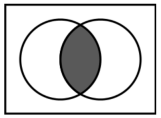
## Set Operators, Text & Conditional Functions

`UNION` ..... Samenstelling
`INTERSECT` ..... Doorsnede
`EXCEPT` ..... Verschil

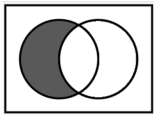
*Query 1* / **UNION** / *Query 2* } **neemt de UNIE van de rt van beide queries**



*Query 1* / ***INTERSECT*** / *Query 2* } **neemt de DOORSNEDE van de rt van beide queries**



*Query 1* / ***EXCEPT*** / *Query 2* } **neemt het VERSCHIL van de rt van beide queries**



## UNION

• Plaatst de rijen uit de resultatentabel van de query's samen in één resultatentabel.
• Haalt er afhankelijk van het al dan niet aanwezig zijn van de ALL optie, de dubbele rijen tussenuit

## INTERSECT

• Plaatst alle rijen van beiden tabellen samen in één resultatentabel.
• Haalt er de dubbele rijen tussenuit.

## EXCEPT

• Plaatst de rijen uit de resultatentabel van de 1e query, die niet voorkomen in de resultatentabel van de 2e query in één resultatentabel.
• Haalt er de dubbele rijen tussenuit.

## SET operators combineren

SELECT1 **UNION**
(SELECT2 **EXCEPT** SELECT3)

## Tekstfuncties

`UPPER` ..... Omzetten naar hoofdletters
`LOWER` ..... Omzetten naar kleineletters
`INITCAP` ..... Beginletter omzetten naar hoofdletter
`SUBSTR(attr, start, count)` ..... stuk van tekst
`POSITION('n' IN attr)` ..... positie van karakter
`CONCAT` ..... tekst aanelkaar plakken
`LPAD(attr, lenght, fill)` ..... links aanvullen
`RPAD(attr, length, fill)` ..... rechts aanvullen
`REPLACE(text, from, to)` ..... karakters vervangen
`TRIM(TRALING|LEADING|BOTH 'a' FROM attr)` .. karakters verwijderen

## Condtionele functies

```
SELECT employee\_id, manager\_id,
GREATEST(employee\_id, manager\_id),
LEAST(employee\_id, manager\_id)
FROM employees;


SELECT employee_id, province,
  CASE province
    WHEN 'NB' THEN 'Noord_Brabant'
    WHEN 'LI' THEN 'Limburg'
    ELSE province
  END "Full_name"
FROM employees;
```

## COALSCE

• De functie geeft de eerste $NOTNULL$ parameter van de parameterlijst terug.
• De $COALESCE$ functie heeft minstens 2 parameters
• Wanneer alle parameters een $NULL$ waarde bevatten, geeft de functie $NULL$ terug.

```
SELECT COALESCE(hours, 0) FROM tasks;


SELECT name,
  COALESCE(email, phone, cellphone) contact
FROM contact_info;
```

## NULLIF

• Geeft NULL terug als beide expressies hetzelfde resultaat geven.
• Bij ongelijkheid wordt de eerste parameter teruggegeven.

```
NULLIF(value1, value2)
```

## DDL - Create Table

```
CREATE TABLE tabelnaam(
    attribuutnaam gegevenstype
                  [default waarde]
                  [column constraint],
    ...,
    [table constraint]
);
```

## Constraints

`PRIMARY KEY` ... key constraint + entity integrity constraint
`NOT NULL` ..... Mag geen nullwaarde bevatten
`CHECK` ..... Moet aan conditie voldoen
`UNIQUE` ..... Moet uniek zijn
`FOREIGN KEY` ..... referential integrity constraint
`ON DELETE` ..... CASCADE / SET NULL

## Gegevenstype

`CHAR|VARCHAR` ..... Alfanumerieke attributen
`NUMERIC(n,m)` ..... Getallen
`DATE` ..... Datum zonder tijd
`TIME` ..... Tijd zonder datum
`TIMESTAMP` ..... Datum en tijd
`INTERVAL` ..... Tijd interval

"Gebruik het datatype CHAR(n) enkel wanneer je zeker weet dat de attribuutwaarden een vaste lengte hebben"

## DDL - Alter, Drop, Sequence

`INSERT` ..... Gegevens invoegen
`UPDATE` ..... Gegevens wijzigen
`DELETE` ..... Gegevens verwijderen
`TRUNCATE` ..... Tabellen leegmaken
`DROP` ..... Objecten verwijderen
`ALTER` ..... Objecten wijzigen
`ALTER ADD COLUMN|CONSTRAINT` ..... aan tabel toevoegen
`ALTER DROP COLUMN|CONSTRAINT` .... van tabel verwijderen
`ALTER RENAME COLUMN|CONSTRAINT` .. van tabel hernoemen

```
INSERT INTO foo (foo, bar) VALUES (a, b);
```

```
UPDATE tabelnaam
SET attribuutnaam=nieuwe waarde
WHERE conditie;
```

```
DELETE FROM deps USING emp
WHERE deps.emp_id=emp.id
AND lower(name)='bob';
```

## Sequence

"Wordt gebruikt om automatisch volgnummers te genereren."

```
CREATE SEQUENCE [IF NOT EXISTS] sequencenaam
INCREMENT [BY ] increment
[MINVALUE minvalue | NO MINVALUE]
[MAXVALUE maxvalue| NO MAXVALUE]
[START [WITH] startwaarde
[CACHE cache]
[[NO] CYCLE]
```

```
CREATE SEQUENCE seq_project_id
START WITH 100 INCREMENT BY 5;
-- 100 105 110 115 120...

INSERT INTO projects
VALUES ( nextval('seq_project_id'), 'Foo_Bar');
```

## Subqueries

Een subquery is een query in een andere query.
- Wordt eerst de subquery uitgevoerd.
- Levert waarden aan de hoofdquery.

```
SELECT *
FROM emplyoees
WHERE salary = (
  SELECT MIN(salary) FROM employees
);
```

"Eerst wordt de binnenste $SELECT$ uitgevoerd, daarna de buitenste "

Waar kan je subquery schrijven?
$SELECT, FROM, WHERE, HAVING$

Subqueries die meer dan 1 rij opleveren
$< ANY$ ..... Minder dan het hoogste
$> ANY$ ..... Meer dan de laagste
$= ANY$ ..... gelijk aan één van de resultaten
$! = ANY$ ..... verschillend van één van de resultaten
$> ALL$ ..... Meer dan de hoogste
$< ALL$ ..... Minder dan de laagste
$<> ALL$ ..... verschillend van alle resultaten

Alternatief: $< MAX$ ..... Minder dan het hoogste
$> MIN$ ..... Meer dan de laagste
$IN$ ..... gelijk aan één van de resultaten
$> MAX$ ..... Meer dan de hoogste
$< MIN$ ..... Minder dan de laagste
$NOTIN$ ..... verschillend van alle resultaten

## Views

Via een view kan men gegevens van de onderliggende tabel afschermen.

```
CREATE OR REPLACE VIEW v_dept_view
  AS SELECT department_id,department_name
    FROM DEPARTMENTS
    ORDER BY department_id;
```

## Correlated Subqueries

Er is een 'correlatie' (= samenhang) tussen beide queries.
- Daarna levert de subquery informatie aan de hoofdquery.
- Levert de hoofdquery eerst informatie aan de subquery.

```
SELECT employee_id, last_name, department_id
FROM employees e
WHERE NOT EXISTS (
  SELECT 'x'
  FROM employees
  WHERE manager_id = e.employee_id
);
```

## Transacties, Locking & Beveiliging

Een transactie bestaat uit bij elkaar horende instructies, waarvoor de DBMS garantie geeft dat ze samen slagen of falen.

```
BEGIN;
BEGIN TRANSACTION;
START TRANSACTION;

SAVEPOINT foo;

-- Alle queries worden hierin vastgelegd.
-- LOCK wordt uitgevoerd per ROW.
COMMIT;

-- Alle statements worden hier teruggedraaid.
ROLLBACK;
ROLLBACK TO SAVEPOINT foo;

END;
END TRANSACTION;
END WORK;

BEGIN;
```

## Privileges

```
CREATE USER user PASSWORD 'secret123';
CREATE USER user SUPERUSER;

CREATE ROLE admin WITH SUPERUSER;
CREATE ROLE admin LOGIN INHERIT;

GRANT ALL PRIVILEGES ON table TO user WITH GRANT OPTION;
REVOKE ALL PRIVILEGES ON table FROM user CASCADE;

DROP USER if exists user;
DROP ROLE if exists admin;

select * from information_schema.enabled_roles;
select * from information_schema.role_column_grants;
select * from information_schema.role_table_grants;
```