BIOINFORMATICA

INFORMATIESYSTEMEN MET PYTHON_

JOINS

HAN_UNIVERSITY
OF APPLIED SCIENCES

TERUGBLIK FUNCTIES

- Functies zijn te verdelen in
  - Tekst functies
  - Nummerieke functies
  - Datum functies
  - Algemene functies

- https://dev.mysql.com/doc/refman/8.0/en/functions.html

STUDIEWIJZER

Les	Onderwerp	Theorie
1	Introductie tot Informatie Systemen	H1 Introduction to Database Development H2 Entity Relationships H3 Complex Relationships H4 Logical Database Design
2	Database ontwerp Constraints en normaliseren	H5 Normalisation H6 Introduction to Oracle SQL H7 Foreign Keys
3	SQL, operatoren en functies	H8 Selecting data from a Table
4	Rijen ophalen uit meerdere tabellen	H9 Selecting data from multiple tables
5	Subquery's en groepfuncties	H10 Subqueries and Group Functions
6	SQL Embedden in Python	
7	Python en SQL: mogelijkheden en risico's	

DOELEN

Aan het eind van deze week heb je kennis over

- Rijen uit verschillende tabellen aan elkaar verbinden met joins
- Het Cartesisch product herkennen
- Equi join en outer join
- ISO Join Syntax

AGENDA

Cartesisch product

Equi join

ISO syntax

Outer join

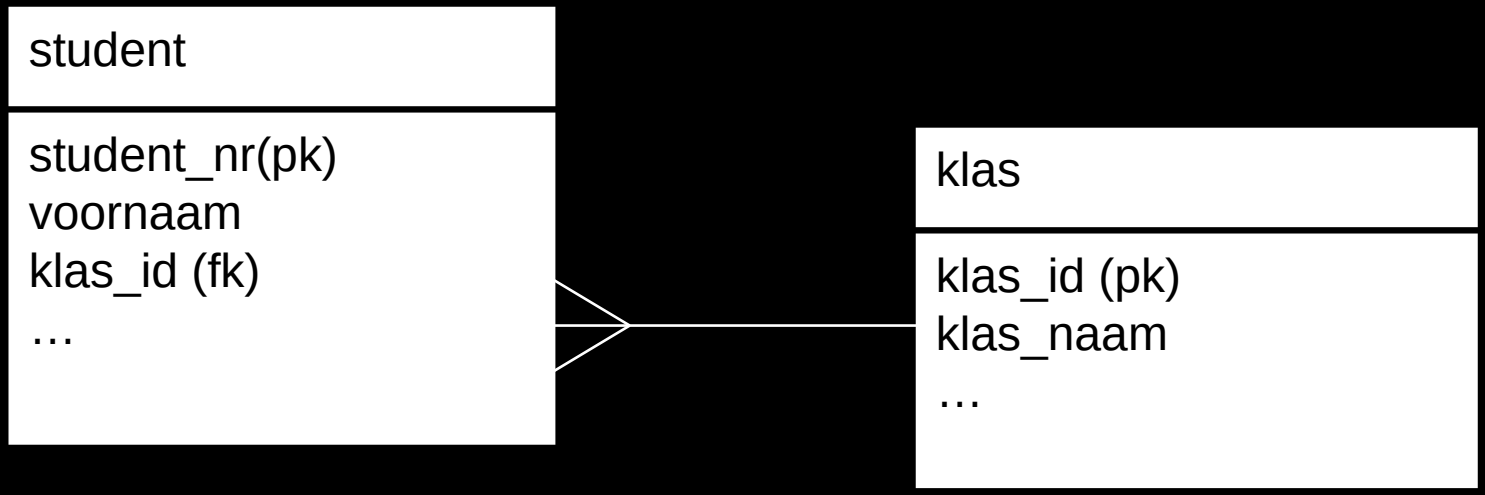
HET GENERIEKE SELECT STATEMENT

```
select <kolomnamen|*>
from   <tabel>
where  <conditie>
order by <kolomnaam>
```

Door meerdere tabellen op te nemen in de from clausule is het mogelijk om tabellen aan elkaar te knopen

CARTESISCH PRODUCT

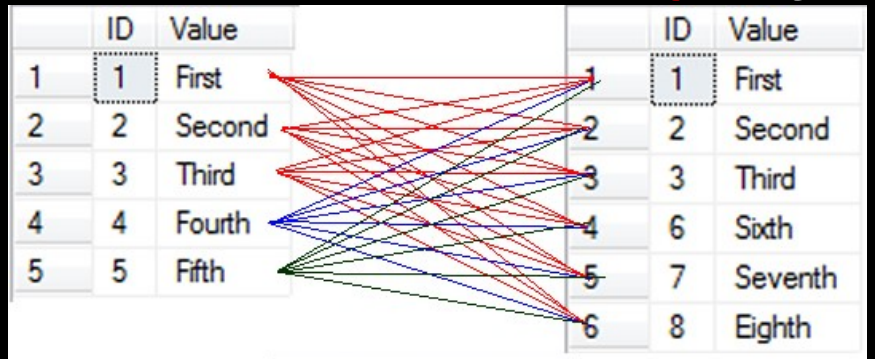
```
select *
from   student
,      klas;
```



CARTESISCH PRODUCT

Het cartesisch product is een koppeling van alle rijen uit de een tabel aan alle rijen uit de andere tabel

Cartesisch product wordt ook wel **cross join** genoemd

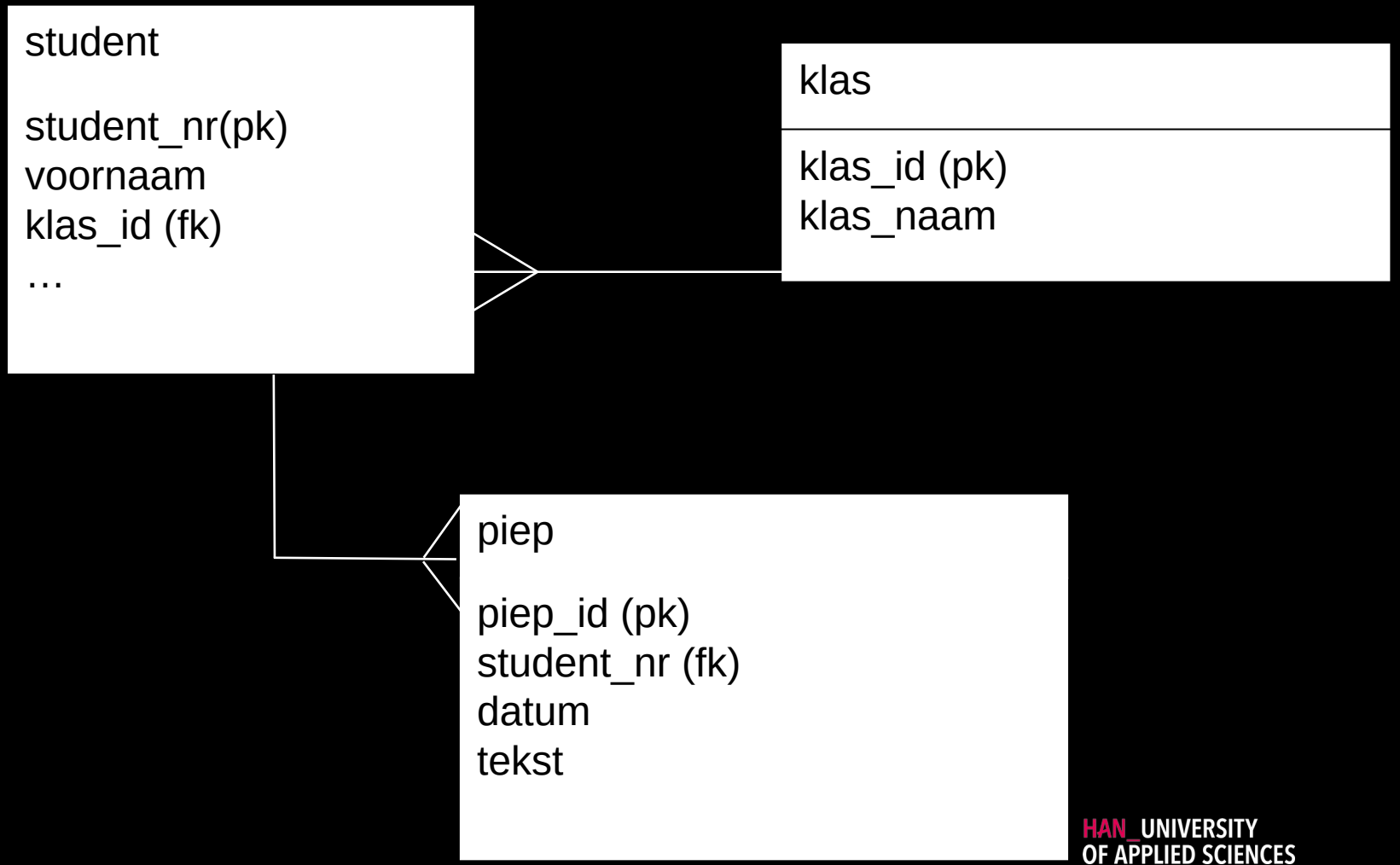


	ID	Value
1	1	First
2	2	Second
3	3	Third
4	4	Fourth
5	5	Fifth

	ID	Value
1	1	First
2	2	Second
3	3	Third
4	6	Sixth
5	7	Seventh
6	8	Eighth

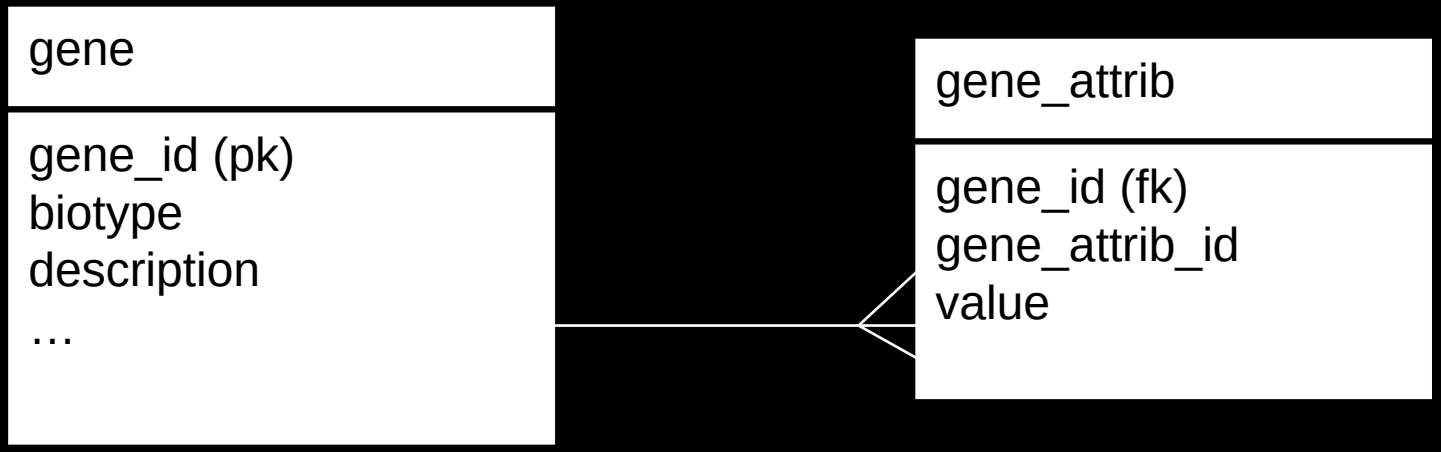
	ID	Value	ID	Value
1	1	First	1	First
2	1	First	2	Second
3	1	First	3	Third
4	1	First	6	Sixth
5	1	First	7	Seventh
6	1	First	8	Eighth
7	2	Second	1	First
8	2	Second	2	Second
9	2	Second	3	Third
10	2	Second	6	Sixth
11	2	Second	7	Seventh
12	2	Second	8	Eighth

SCHEMA STUDENTEN DATABASE



CARTESISCH PRODUCT

```
select *
from   gene
,      gene_attrib;
```



CARTESISCH PRODUCT

Wanneer een **join conditie** ontbreekt verkrijg je altijd het cartesisch product

De relatie tussen tabellen is terug te vinden in de tabellen

AGENDA

Cartesisch product

Equi join

ISO syntax

Outer join

EQUI JOIN

We kunnen tabellen aan elkaar koppelen op basis van een **foreign key** die verwijst naar een **primary key**

Dit noemen we een **equi join**: de waardes zijn gelijk

CARTESISCH PRODUCT

```
select student_nr
,      voornaam
,      tussenvoegsels
,      achternaam
,      klas_id
,      klas_id
,      klas_naam
from   student
,      klas
```

Ambiguous naming: de database weet niet uit welke tabel deze kolom moetkkomen

CARTESISCH PRODUCT

```
select s.student_nr
,      s.voornaam
,      s.tussenvoegsels
,      s.achternaam
,      s.klas_id
,      k.klas_id
,      k.klas_naam
from   student s
,      klas   k
```

Alliassing van een tabel

CARTESISCH PRODUCT

```
select s.student_nr
,      s.voornaam
,      s.tussenvoegsels
,      s.achternaam
,      s.klas_id
,      k.klas_id
,      k.klas_naam
from   student s
,      klas   k
```

VAN CARTESISCH PRODUCT NAAR JOIN

Wanneer het cartesisch product als uitgangspunt wordt genomen zien we combinaties van alle rijen

Een student koppelen we aan alle klassen

EQUI JOIN

Wanneer we de foreign key gelijkstellen aan de primary key koppelen we op basis van gelijkheid (equi join)

EQUI JOIN

```
select s.student_nr
,      s.voornaam
,      s.tussenvoegsels
,      s.achternaam
,      s.klas_id
,      k.klas_id
,      k.klas_naam
from   student s
,      klas   k
where  s.klas_id = k.klas_id
```

EQUI JOIN

```
where  s.klas_id = k.klas_id
```

student_nr	voornaam	tussenvoegsel	achternaam	s.klas_id	k.klas_id	klas_naam
687532	Pieter		Derks	1	1	bi1b
795132	Lucky		Luke	1	1	bi1b
326789	Lukas		Terlande	1	1	bi1b
791234	Heinz		Potter	2	2	bi1c
894523	Marco		Verhof	2	2	bi1c

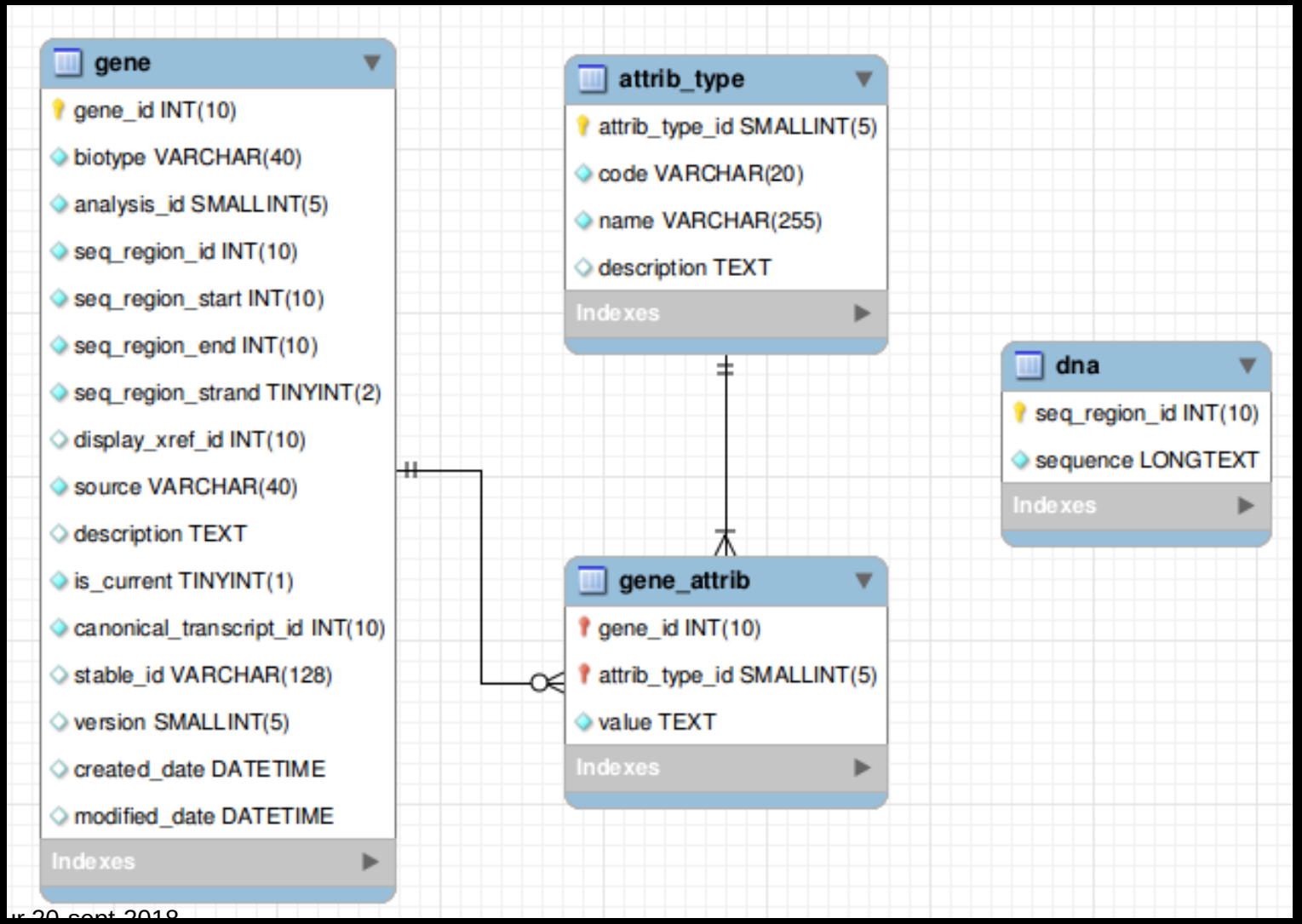
VAN CARTESISCH PRODUCT NAAR JOIN

Door de conditie er op te leggen dat klas_id = klas_id filteren we op rijen waarbij de fk = pk

FK PK

student_nr	voornaam	tussenvoegsel	achternaam	s.klas_id	k.klas_id	klas_naam
687532	Pieter		Derks	1	1	bi1b
795132	Lucky		Luke	1	1	bi1b
326789	Lukas		Terlande	1	1	bi1b
791234	Heinz		Potter	2	2	bi1c
894523	Marco		Verhof	2	2	bi1c

FRAGMENT ERD ENSEMBLDB



VOORBEELD CARTESISCH PRODUCT

```
select g.gene_id
,      g.biotype
,      a.attrib_type_id
,      a.value
from   gene g
,      gene_attrib a;
```

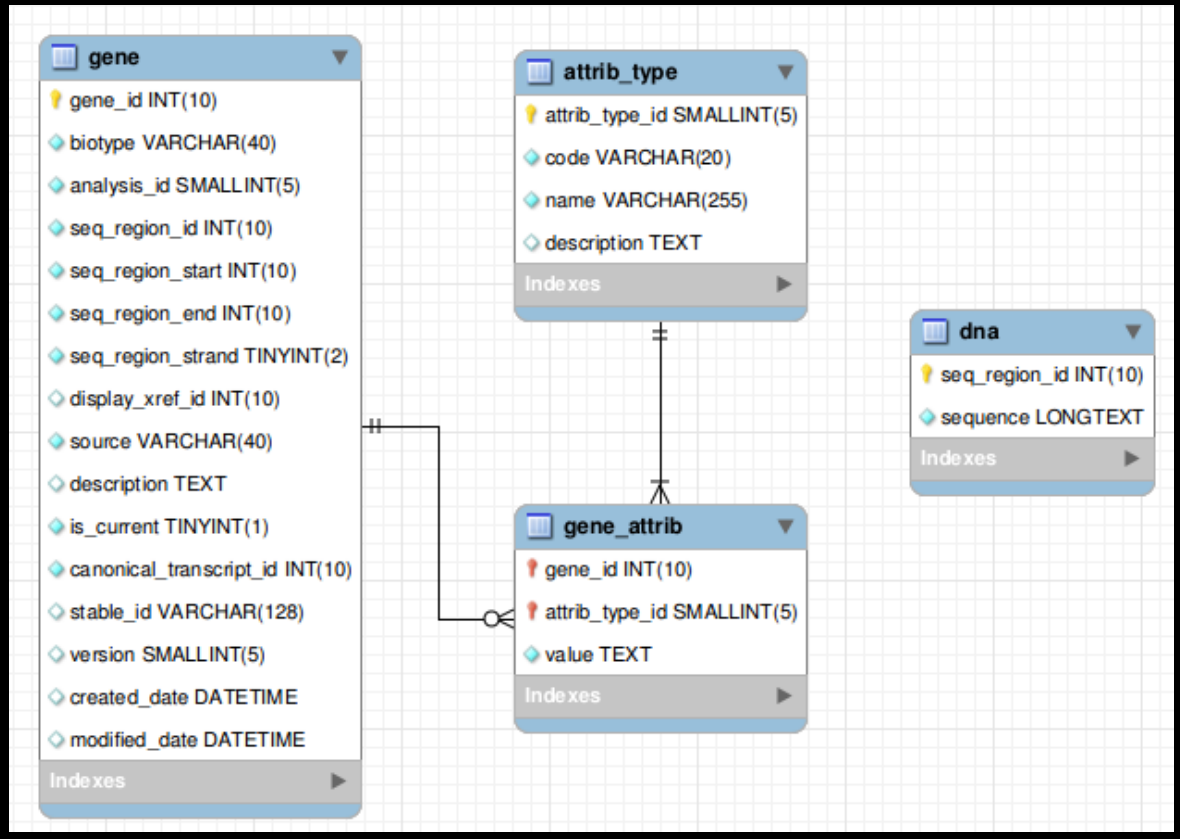
VAN CARTESISCH PRODUCT NAAR EQUI JOIN

```
select g.gene_id
,      g.biotype
,      a.attrib_type_id
,      a.value
from   gene g
,      gene_attrib a;
```

Pas deze query zo aan dat je genen koppelt aan attributen

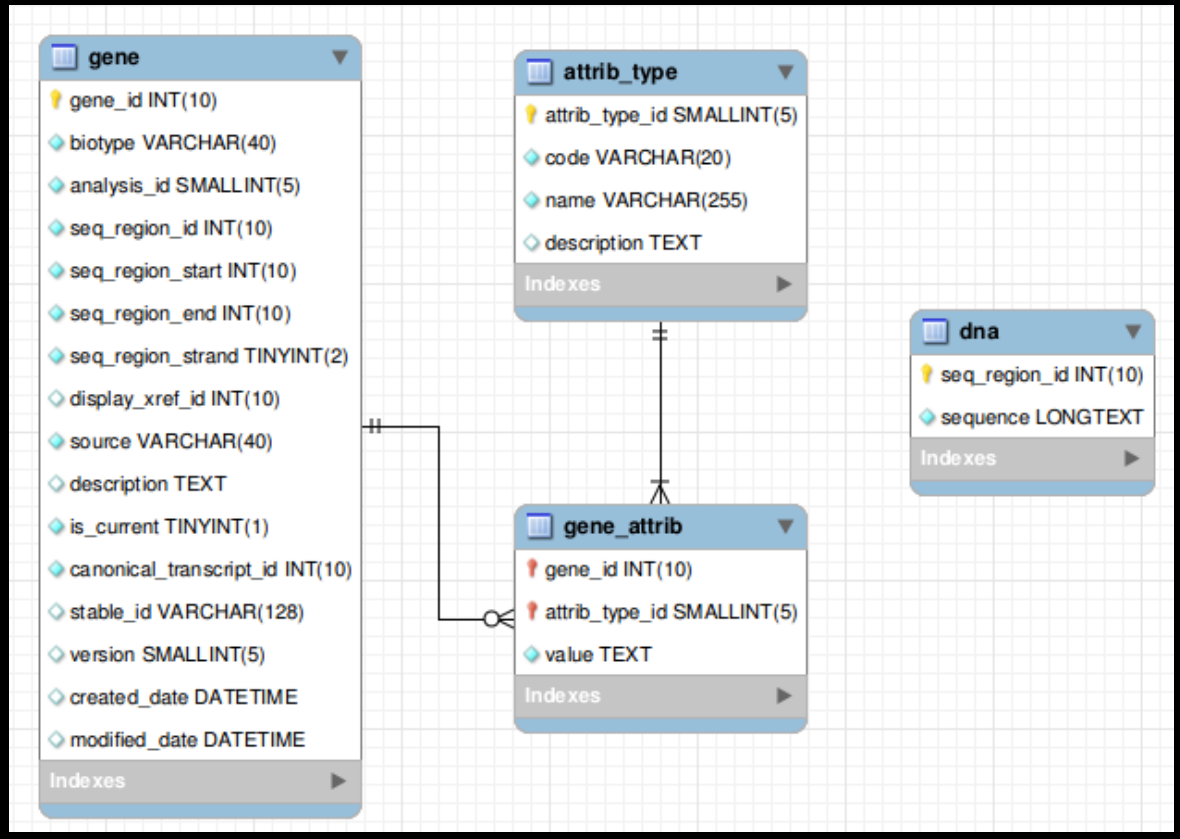
QUERY

Schrijf de query om genen te koppelen aan de attributen met het attrib_type type (code en name)



QUERY

Welk gen heeft het hoogste GC percentage?
Tip: GC percentage is “attrib_type_id” 142



EQUI JOIN SAMENVATTING

Meerdere tabellen zijn te koppelen door in de where clausule de tabellen aan elkaar te knopen

Het aantal join condities is het aantal tabellen min 1

AGENDA

Cartesisch product

Equi join

ISO syntax

Outer Join

ISO SQL SYNTAX

ISO: Internationale Organisatie voor Standaardisatie

Op internationaal niveau is een syntax afgesproken voor het schrijven van query's zodat alle relationele databases dezelfde syntax begrijpen

ISO STANDAARD

International Standards Organisation

Wereldwijde standaard afspraak voor de schrijfwijze (syntax) van SQL statements

Hierdoor *luisteren* alle databases naar hetzelfde SQL statement

DE EQUI JOIN

De equi join kunnen we herschrijven:

- natural join
- join using
- join on

NATURAL JOIN

```
select voornaam
,      achternaam
,      klas_naam
from   student
natural join klas
```

NATURAL JOIN

Een natural join zal op basis van gelijke kolomnamen tabellen koppelen

VRAAG

Wat is de reden dat deze query geen rijen zal retourneren?

```
select *
from   student
natural join docent
```

JOIN USING

Als in een tabel meerdere kolommen voor komen met dezelfde naam kun je met **join using** aangeven welke kolom je daar uit wilt gebruiken

JOIN USING

```
select *
from   student
join docent using(slb_id)
```

JOIN ON

Om tabellen te koppelen die **geen overeenkomstige namen** hebben gebruik je **join on**

JOIN ON

```
select *
from   student s
join   docent d
on(s.slb_id = d.slb_id)
```

AGENDA

Cartesisch product

Equi join

ISO syntax

Outer Join

JOINS

Join using klas_id

Student

studnr	voornaam	klas_id
178	Jan	7
234	Kees	2
576	Piet	

Klas

klas_id	klasnaam
7	Bin1a
2	Bin1b
4	Chemie2a

(inner) join

voornaam	klasnaam
Jan	Bin1a
Kees	Bin1b

left outer join

voornaam	klasnaam
Jan	Bin1a
Kees	Bin1b
Piet	

full outer join

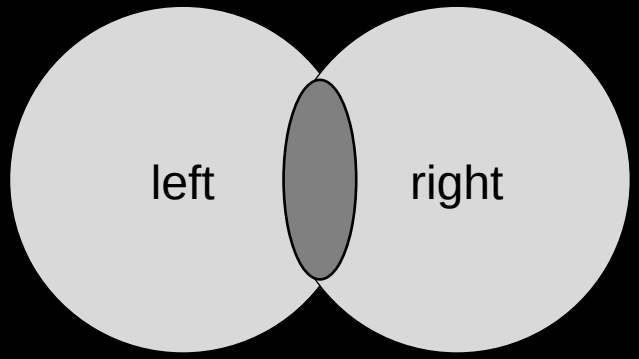
≠

Cartesisch product

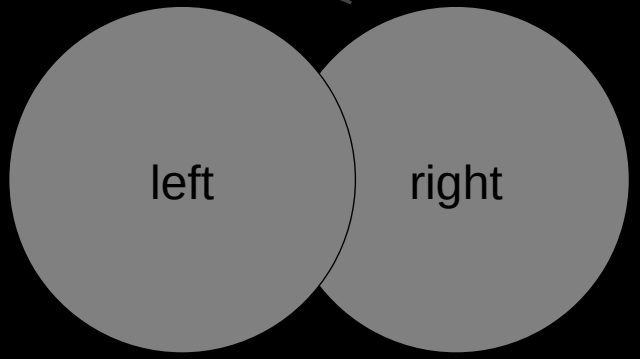
voornaam	klasnaam
Jan	Bin1a
Kees	Bin1b
Piet	
	Chemie2a

JOINS

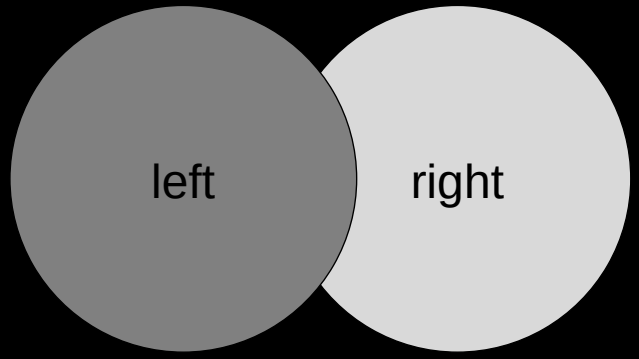
Full outer join is niet mogelijk in MySQL



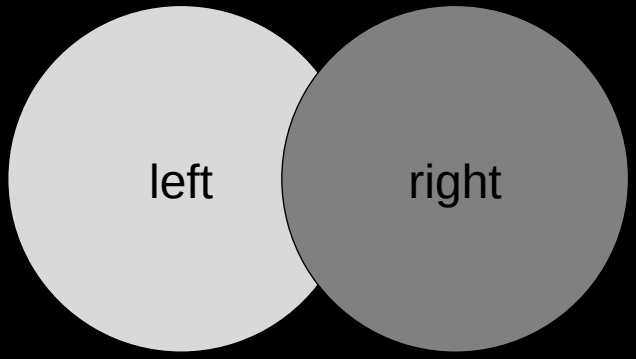
inner join



full outer join

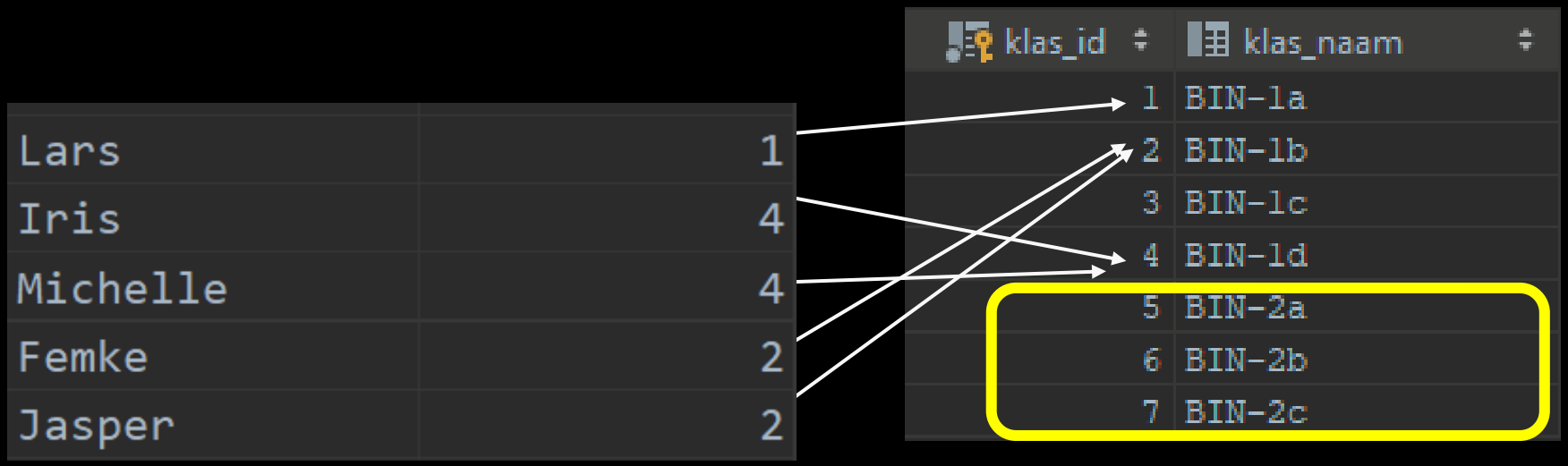


left outer join



right outer join

JOIN – KLASSEN DIE NIET TE ZIEN ZIJN



Lars	1
Iris	4
Michelle	4
Femke	2
Jasper	2

klas_id	klas_naam
1	BIN-1a
2	BIN-1b
3	BIN-1c
4	BIN-1d
5	BIN-2a
6	BIN-2b
7	BIN-2c

OUTER JOINS

Bij sommige rijen zul je geen match hebben met een andere rij

Bijvoorbeeld: er zijn geen studenten voor klas Bin-2a

Als je zo’n rij toch wilt zien gebruik je een outer join

OUTER JOIN – ISO SYNTAX

```
select voornaam
,      klas_naam
from   student
right outer join klas
using (klas_id)
```

OUTER JOIN – ISO SYNTAX

```
select s.voornaam
,      k.klas_naam
from   student s
right outer join klas k
on (s.klas_id = k.klas_id)
```

OUTER JOIN

In de ISO syntax hebben we

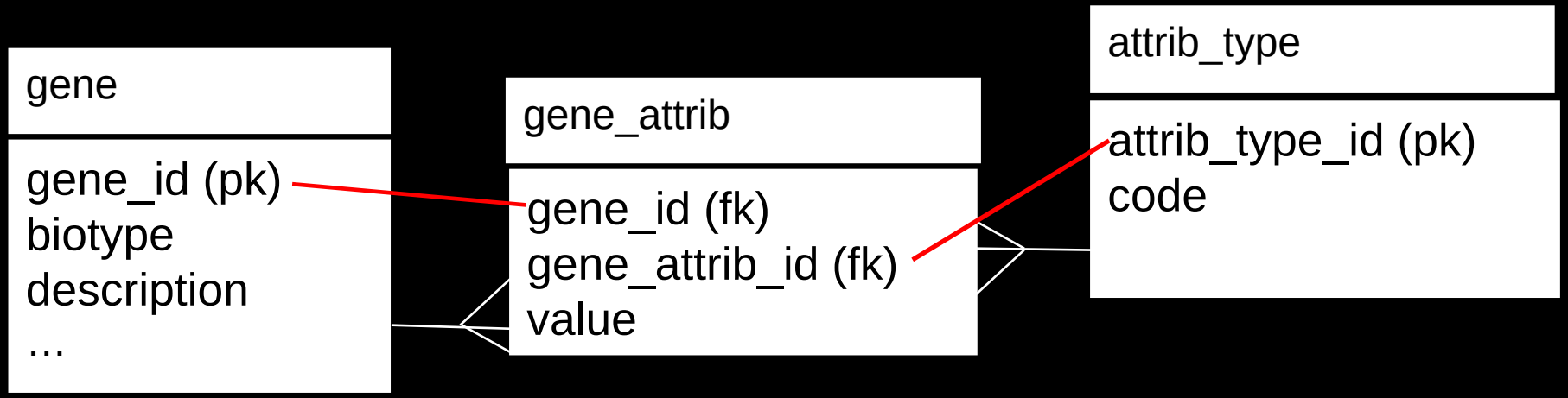
Left outer join

Right outer join

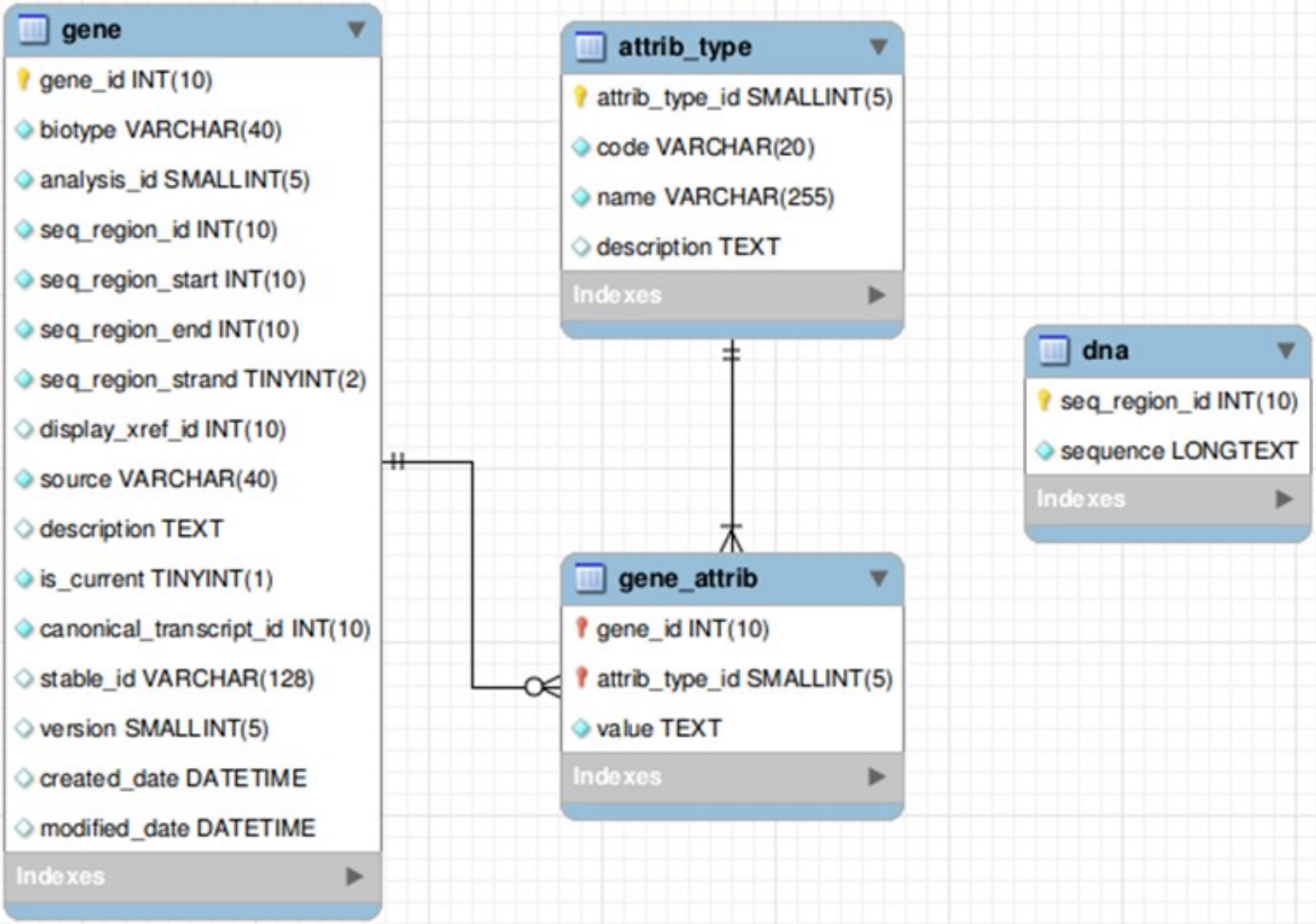
Full outer join (niet in MySQL!)

QUERY

Schrijf de query tegen de ensembl database waarmee je de beschrijving toont van de genen met een GC percentage hoger dan 85%



Voorbeeld ERD: eigen werk auteur


gene
gene_id INT(10)
biotype VARCHAR(40)
analysis_id SMALLINT(5)
seq_region_id INT(10)
seq_region_start INT(10)
seq_region_end INT(10)
seq_region_strand TINYINT(2)
display_xref_id INT(10)
source VARCHAR(40)
description TEXT
is_current TINYINT(1)
canonical_transcript_id INT(10)
stable_id VARCHAR(128)
version SMALLINT(5)
created_date DATETIME
modified_date DATETIME
Indexes
attrib_type
attrib_type_id SMALLINT(5)
code VARCHAR(20)
name VARCHAR(255)
description TEXT
Indexes
dna
seq_region_id INT(10)
sequence LONGTEXT
Indexes
gene_attrib
gene_id INT(10)
attrib_type_id SMALLINT(5)
value TEXT
Indexes

QUERY

```
select gene_id, biotype, gene.description, code,
value
from gene
       join gene_attrib using (gene_id)
       join attrib_type using (attrib_type_id)
where code = 'GeneGC'
and   value > 85
```

QUERY

Schrijf de query waarmee je de genen toont waar geen attributen bij gegeven zijn

QUERY

```
select gene_id, biotype, gene.description, value
from gene
       left outer join gene_attrib using (gene_id)
where value is null
```

```
select gene_id, biotype, gene.description, code, value
from gene
       left outer join gene_attrib using (gene_id)
       left outer join attrib_type using
(attrib_type_id)
where value is null
```

VRAAG

Schrijf de query om de studenten te tonen die niet in een klas zitten

QUERY

Toon de attributen waar geen gebruik van wordt gemaakt

QUERY

```
select gene_id, biotype, description, value
from gene
       right outer join gene_attrib using
(gene_id)
where gene_id is null
```

SAMENVATTING

- In de **from clausule** is het mogelijk meerdere tabellen op te nemen
- Om deze aan elkaar te knopen maken we gebruik van **join syntax**
- Join syntax is
  - een link middels een **where clausule**
  - of een koppeling via de **ISO syntax**

SAMENVATTING

Cartesisch product (cross join)

Equi join (inner join)

Outer joins

- **Left**
- **Right**
- **Full**