Databanken – SQL – Joins

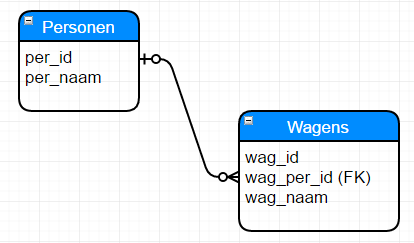
# Introductie

In databanken zit gerelateerde informatie verspreid over verschillende tabellen. Dat is het gevolg van het normalisatieproces.

Met joins kan deze informatie opnieuw samengevoegd worden.

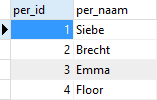
# Voorbeeld

In een databank bestaan de tabellen *personen* en *wagens*. Sommige personen hebben een wagen (of meer dan één). Sommige personen hebben geen wagen. Niet elke wagen hoort bij een persoon. Dat weerspiegelt zich in volgend ERD:



Laten we veronderstellen dat de tabellen volgende gegevens bevatten:

***Personen***



***Wagens***



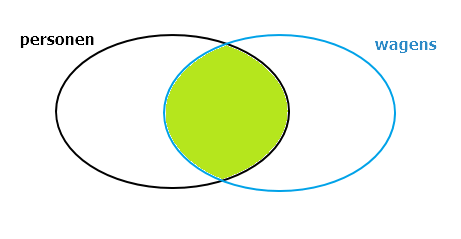
Dat betekent dus het volgende:

* Siebe heeft een Volvo
* Brecht heeft een Audi en een Mercedes
* Emma en Floor hebben geen auto
* De Fiat en de Citroën zijn van niemand

# Standaard Joins

## INNER JOIN

De INNER JOIN geeft als resultaat alle rijen waarvoor er in beide tabellen gegevens aanwezig zijn.

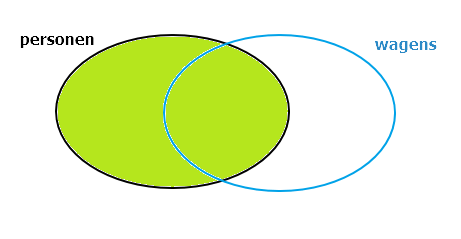


select \* from personen INNER JOIN wagens ON per\_id=wag\_per\_id

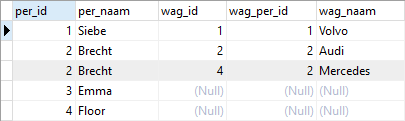


## LEFT JOIN

De LEFT JOIN geeft als resultaat alle rijen uit de linker tabel, aangevuld met informatie uit de rechter tabel. Ook rijen uit de linker tabel waarvoor er geen overeenkomstige gegevens in de rechter tabel bestaan, worden opgenomen in het resultaat.

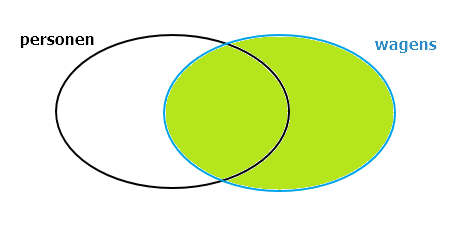


select \* from personen LEFT JOIN wagens ON per\_id=wag\_per\_id

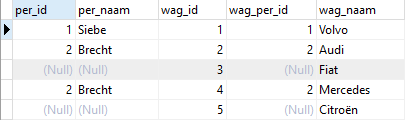


## RIGHT JOIN

De RIGHT JOIN geeft als resultaat alle rijen uit de rechter tabel, aangevuld met informatie uit de linker tabel. Ook rijen uit de rechter tabel waarvoor er geen overeenkomstige gegevens in de linker tabel bestaan, worden opgenomen in het resultaat.

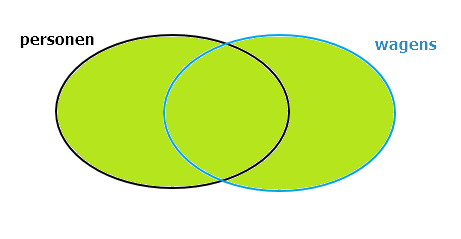


select \* from personen RIGHT JOIN wagens ON per\_id=wag\_per\_id

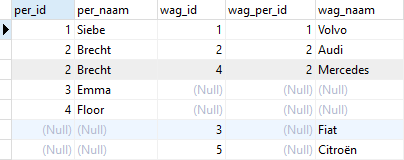


## FULL JOIN

De FULL JOIN geeft als resultaat alle rijen uit de linker tabel, aangevuld met informatie uit de rechter tabel én alle rijen uit de rechter tabel.



select \* from personen FULL JOIN wagens ON per\_id=wag\_per\_id



Helaas bestaat het commando FULL JOIN niet in MySQL. Het resultaat van hierboven werd bekomen met volgend statement:

select \* from personen LEFT JOIN wagens ON per\_id=wag\_per\_id

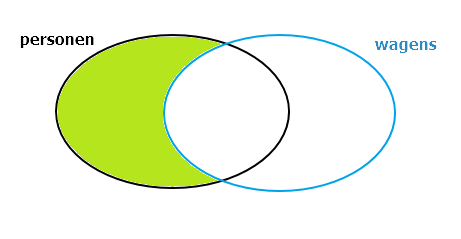
UNION

select \* from personen RIGHT JOIN wagens ON per\_id=wag\_per\_id

# Speciale gevallen

## LEFT JOIN (IF NULL)

Wanneer men enkel de lijnen wil hebben uit de linker tabel waarvoor er geen overeenkomstige gegevens zijn in de rechter tabel, kan dat als volgt:



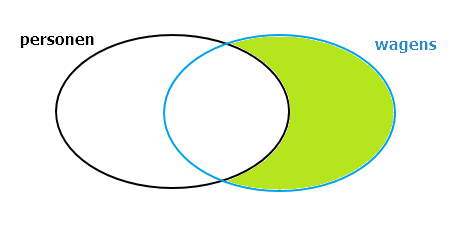
select \* from personen LEFT JOIN wagens ON per\_id=wag\_per\_id

where wag\_per\_id IS NULL



## RIGHT JOIN (IF NULL)

Op dezelfde manier kan men, wanneer men enkel de lijnen wil hebben uit de rechter tabel waarvoor er geen overeenkomstige gegevens zijn in de linker tabel, dat als volgt bekomen:



select \* from personen RIGHT JOIN wagens ON per\_id=wag\_per\_id

where wag\_per\_id IS NULL



Maar: dit kan ook eenvoudig met het volgende statement:

select \* from wagens

where wag\_per\_id IS NULL



Deze vereenvoudiging is enkel mogelijk aan de veel-kant van de relatie, omdat zich daar ook altijd de Foreign Key bevindt; in ons geval: ***wag\_per\_id***.

## CARTESIAANS PRODUCT of CROSS JOIN

Wanneer men geen voorwaarde opgeeft bij een join, worden voor elke lijn in de linker tabel, alle lijnen herhaald uit de rechter tabel (of andersom). Dit noemt men een *Cartesiaans product* ofwel een *cross join*.

select \* from personen INNER JOIN wagens



De personen en wagens die naast elkaar staan, zijn in dit geval dus niet noodzakelijk aan elkaar verbonden.

Toch kan het soms zinvol zijn gebruik te maken van deze optie. Bijvoorbeeld om een wedstrijdkalender samen te stellen:

select A.tea\_naam, B.tea\_naam from team A INNER JOIN team B

where A.tea\_id <> B.tea\_id

