KU LEUVEN

# Inleiding tot databanken

## 6. Queryverwerking (Deel 1)

*Prof. dr. Paolo Pilozzi*

# Overzicht

Queryverwerking bespreekt hoe een (R)DBMS queries behandelt en uitvoert op een zo optimaal mogelijke manier.
Hoofdstuk 15 – Oefenzitting 5

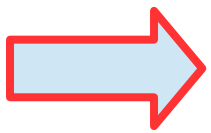
HC8 (Deel 1):
* **Inleiding & Herhaling**
* Queryverwerking -en optimalisatie
* Heuristische optimalisatie van querybomen

HC9 (Deel 2):
* Extern sorteren
* Implementaties van operatoren
* Queryuitvoering

# 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

# SQL

= **Wat** we willen uitvoeren op een databank.

Voorbeeld: Naam en adres van werknemers uit het "Research Departement".

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

Hoe zal een DBMS dit uitvoeren?

**DEPARTMENT**

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|---|---|---|---|

**EMPLOYEE**

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|---|---|---|

# Relationele Algebra

= **Hoe** we uitvoeren op een databank.

De fundamentele operatoren in relationele algebra zijn:

$\{\sigma, \pi, \rho, \cup, -, \times\}$

=> SELECT, PROJECT, RENAME, UNION, MINUS, CART.PROD.

De andere operatoren kunnen ervan afgeleid worden:

$R \cap S \equiv R \cup S - ((R - S) \cup (S - R))$

$R \bowtie_F S \equiv \sigma_F(R \times S)$

# Relationele Algebra

= **Hoe** we uitvoeren op een databank.

Voorbeeld: Naam en adres van werknemers uit "Research Departement".

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

$R1 \leftarrow DEPARTMENT \times EMPLOYEE$

$R2 \leftarrow \sigma_{Dname='Research'}(R1)$

$R3 \leftarrow \sigma_{Dnumber=Dno}(R2)$

$R4 \leftarrow \pi_{Fname,Lname,Address}(R3)$

**Optie 1**

# Relationele Algebra

= **Hoe** we uitvoeren op een databank.

Voorbeeld: Naam en adres van werknemers uit "Research Departement".

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

$$R1 \leftarrow \sigma_{Dname='Research'}(DEPARTMENT)$$
$$R2 \leftarrow R1 \bowtie_{Dnumber=Dno} EMPLOYEE$$
$$R3 \leftarrow \pi_{Fname,Lname,Address}(R2)$$

**Optie 2**

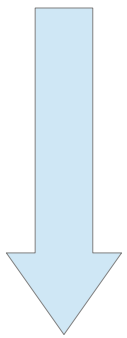
# Queryverwerking

= Omzetten, kiezen en optimaliseren.

Voorbeeld: Naam en adres van werknemers uit "Research Departement".

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

$R1 \leftarrow DEPARTMENT \times EMPLOYEE$
$R2 \leftarrow \sigma_{Dname='Research'}(R1)$
$R3 \leftarrow \sigma_{Dnumber=Dno}(R2)$
$R4 \leftarrow \pi_{Fname,Lname,Address}(R3)$



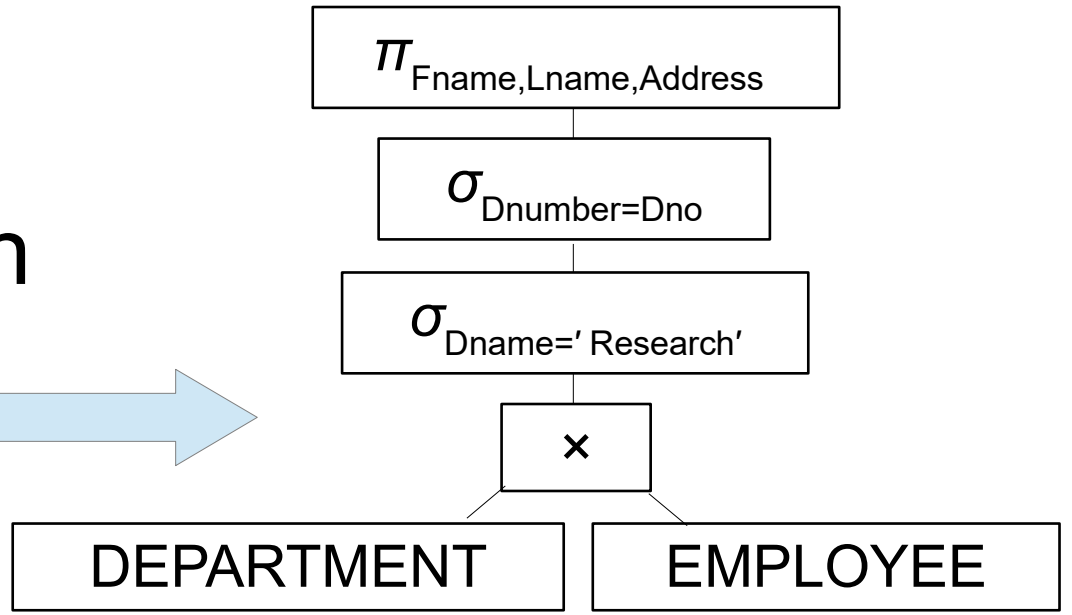
$R1 \leftarrow \sigma_{Dname='Research'}(DEPARTMENT)$
$R2 \leftarrow R1 \bowtie_{Dnumber=Dno} EMPLOYEE$
$R3 \leftarrow \pi_{Fname,Lname,Address}(R2)$

=> Bespreken van:
**Kiezen & Efficiënte Operatoren**

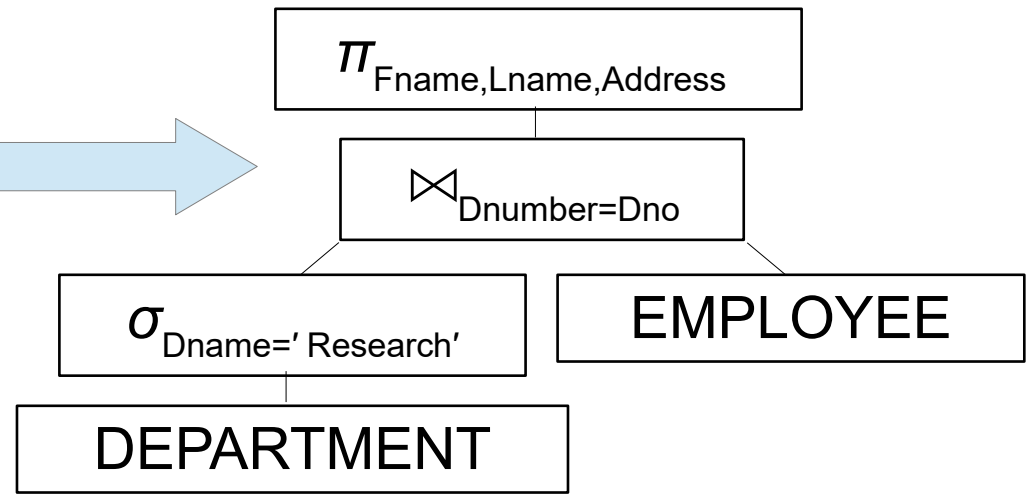
# Queryverwerking

= Kiezen adhv. Query-bomen

$R1 \leftarrow DEPARTMENT \times EMPLOYEE$
$R2 \leftarrow \sigma_{Dname='Research'}(R1)$
$R3 \leftarrow \sigma_{Dnumber=Dno}(R2)$
$R4 \leftarrow \pi_{Fname,Lname,Address}(R3)$



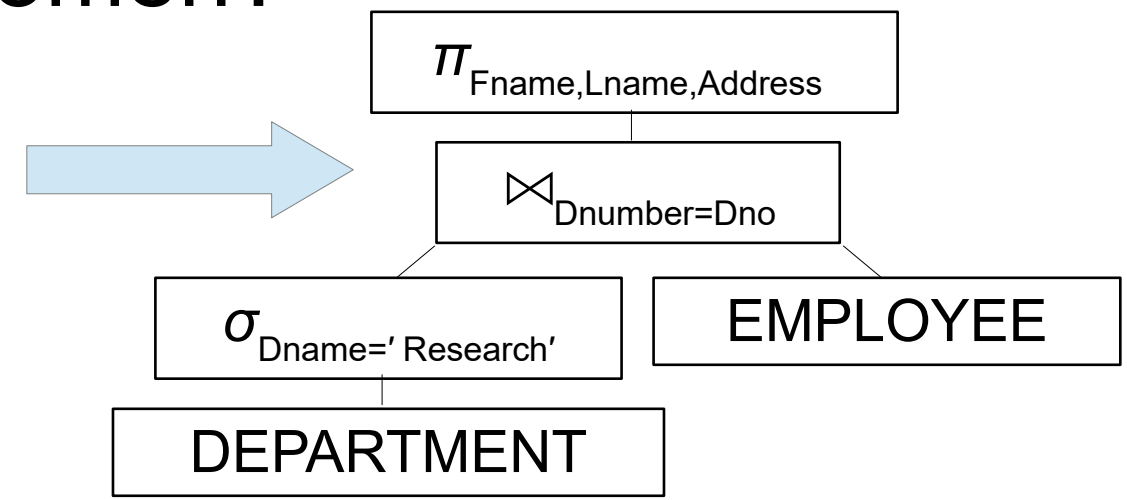

$R1 \leftarrow \sigma_{Dname='Research'}(DEPARTMENT)$
$R2 \leftarrow R1 \bowtie_{Dnumber=Dno} EMPLOYEE$
$R3 \leftarrow \pi_{Fname,Lname,Address}(R2)$



# Queryverwerking

## Hoe kiezen adhv. Query-bomen?

$R1 \leftarrow \sigma_{Dname='Research'}(DEPARTMENT)$
$R2 \leftarrow R1 \bowtie_{Dnumber=Dno} EMPLOYEE$
$R3 \leftarrow \pi_{Fname,Lname,Address}(R2)$




Belangrijkste maatstaf is
Blokkentransport!

=> Waarbij we gebruik maken van indexeringen. Voorbeeld:
* Join op Dnumber en Dno met B+-bomen
* Selectie op basis van "Research" met Hash-index

# Overzicht

Queryverwerking bespreekt hoe een (R)DBMS queries behandelt en uitvoert op een zo optimaal mogelijke manier.
Hoofdstuk 15 – Oefenzitting 5

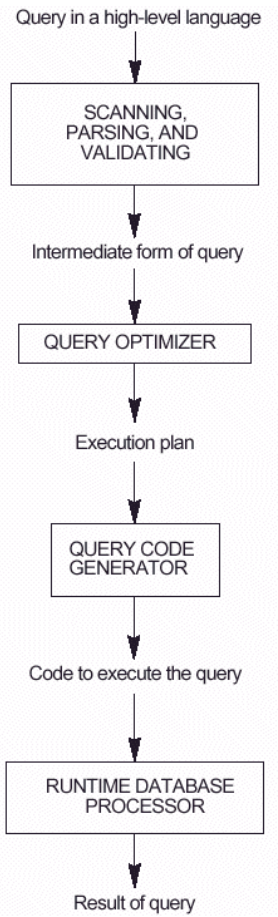
HC8 (Deel 1):
* Inleiding & Herhaling
* **Queryverwerking -en optimalisatie**
* Heuristische optimalisatie van querybomen

HC9 (Deel 2):
* Extern sorteren
* Implementaties van operatoren
* Queryuitvoering

# Queryverwerking – Overzicht

Stappen in verwerking van query:

1. Lezen en ontleden van query:
   - Scanner => Lezen v. Query
   - Parser => Syntax + Voorstelling
2. Query optimalisatie
   - Genereer opties/strategieën
   - Kies meest efficiënte
3. Codegeneratie
   - Maak gekozen strategie uitvoerbaar
4. Voer uit




Optie 1: Interpreted
=> Directe uitvoering

Optie 2: Compiled
=> Code opgeslagen voor herbruik

# Queryverwerking – Overzicht

**Doel**: Weinig en kleine records & Efficiënt blokkentransport

A. Zo klein mogelijke tussenrelaties
   1. Op schema-niveau Transformatieregels relationele algebra
      => Herwerken query-bomen: transversaal schuiven (boven-onder)
   2. Op gegevens-niveau: info rond records, selectiviteit, enz.
      => Herwerken query-bomen: lateraal schuiven (links-rechts)

B. Zo min mogelijk aantal lees, schrijf, -en vergelijkingsoperaties
   3. Op algoritme-niveau: Efficiënte implementaties
      => Gebruik maken van indexen
      => Efficiënte selectie, join, ... operaties

Merk op: A help bij B

# Queryverw. – SQL naar RA

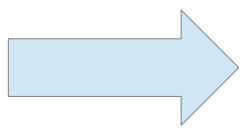
## Van SQL naar Relationele Algebra (RA)

Waarom?

* In SQL zijn volgorde van operaties minder strikt bepaald dan in RA.
  => RA geschikter voor uitvoeringsstrategieën dan SQL

\+ Makkelijk om te vormen.

```
SELECT    <Attributenlijst>
FROM      R1,...,Rn
WHERE     <Conditieformule>;
```



$$\pi_{\text{Attributenlijst}}(\sigma_{\text{Conditieformule}}(R1 \times \ldots \times Rn))$$

# Queryverw. – SQL naar RA

## Van SQL naar Relationele Algebra (RA)

Geneste queries? => Omzetten per blok, van binnen naar buiten.
Voorbeeld:

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

**A**: $c := \Im_{MAX\ Salary}(\sigma_{Dno=5}(EMPLOYEE))$

**B**: $\pi_{Lname,\ Fname}(\sigma_{Salary\ >\ c}(EMPLOYEE))$

# Queryverw. – Optimalisatie

Veel keuzes! Op basis van:

* Volgorde van RA operaties
* Volgorde relaties in product/join
* Implementatie van operaties
* Indexering

Optimalisatieprobleem:

* Elke strategie heeft een kost
  => Zoek minimale kost
* Meestal uitvoeringstijd als kost

# Queryverw. – Aanpak

Aanpak ter optimalisatie:

* Heuristische regels voor het ordenen van operaties (in query-bomen)
  - We bekomen zo verschillende strategieën
* Systematische schatting van de kosten over gekozen strategieën heen
  - Op basis van schatting aantal tupels in tussenresultaten
  - Op basis van blokkentransport
  - Na toepassing heuristieken
* Semantische optimalisatie
  - Gebruik kennis (over beperkingen) om queries te transformeren

# Overzicht

Queryverwerking bespreekt hoe een (R)DBMS queries behandelt en uitvoert op een zo optimaal mogelijke manier.
Hoofdstuk 15 – Oefenzitting 5

HC8 (Deel 1):
* Inleiding & Herhaling
* Queryverwerking -en optimalisatie
* **Heuristische optimalisatie van querybomen**

HC9 (Deel 2):
* Extern sorteren
* Implementaties van operatoren
* Queryuitvoering

# Heuristische optimalisatie

**Doel**: Weinig en kleine records & Efficiënt blokkentransport

A. Zo klein mogelijke tussenrelaties
   1. Op schema-niveau Transformatieregels relationele algebra
      => Herwerken query-bomen: transversaal schuiven (boven-onder)
   2. Op gegevens-niveau: info rond records, selectiviteit, enz.
      => Herwerken query-bomen: lateraal schuiven (links-rechts)

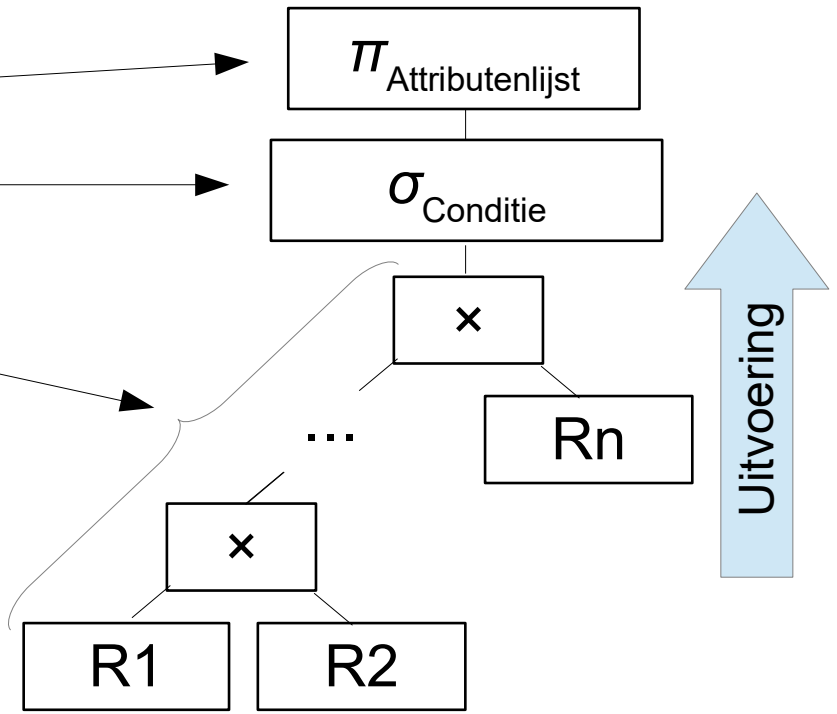
B. Zo min mogelijk aantal lees, schrijf, -en vergelijkingsoperaties
   3. Op algoritme-niveau: Efficiënte implementaties
      => Gebruik maken van indexen
      => Efficiënte selectie, join, ... operaties

Merk op: A help bij B

# Heuristische optimalisatie

Maak een boom in kanonieke vorm:

* SELECT: Projectie
* WHERE: Selectie
* FROM: Cartesisch product




Dan, herstructureer boom

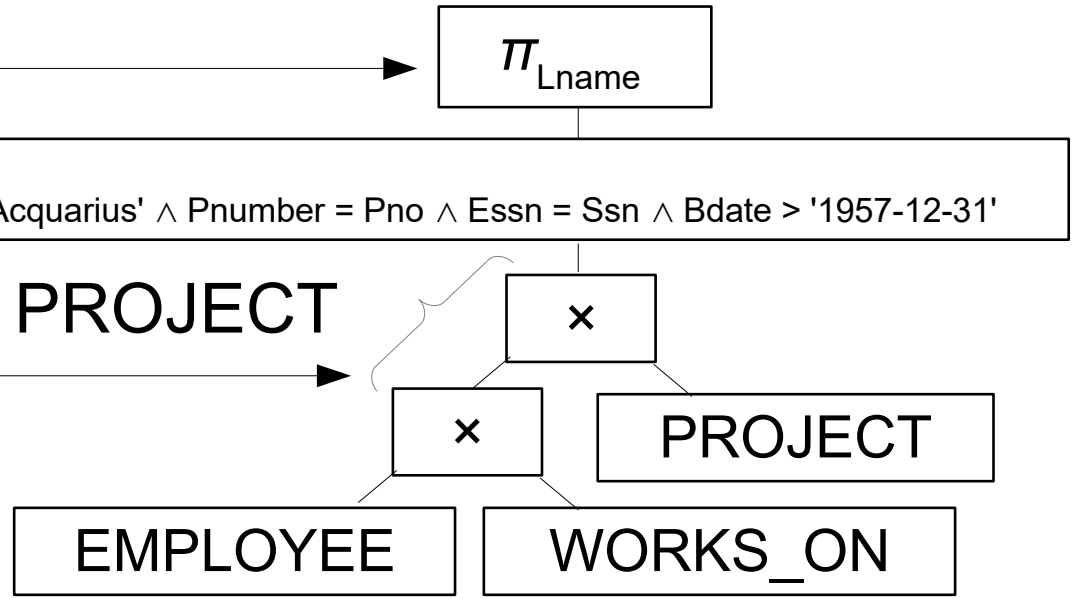
* volgens equivalentie-behoudende regels
* gedreven door heuristieken
  = keuzes die vaak werken

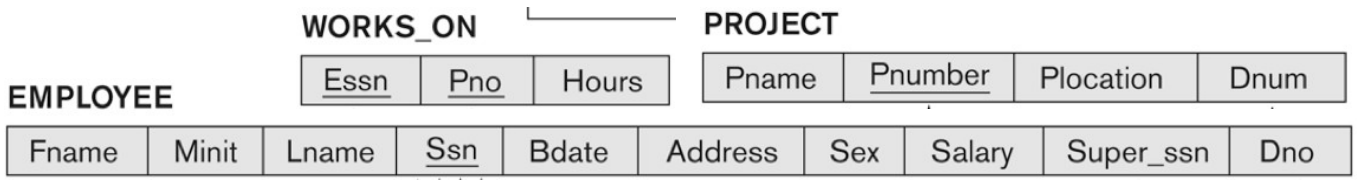
**Doel**: Maak de tussenrelaties zo klein mogelijk!

# Heur. optim. – Voorbeeld

Initiële query-boom maken:

```
SELECT   Lname
FROM     EMPLOYEE, WORKS_ON, PROJECT
WHERE    Pname = 'Acquarius' AND
         Pnumber = Pno AND
         Essn = Ssn AND
         Bdate > '1957-12-31';
```

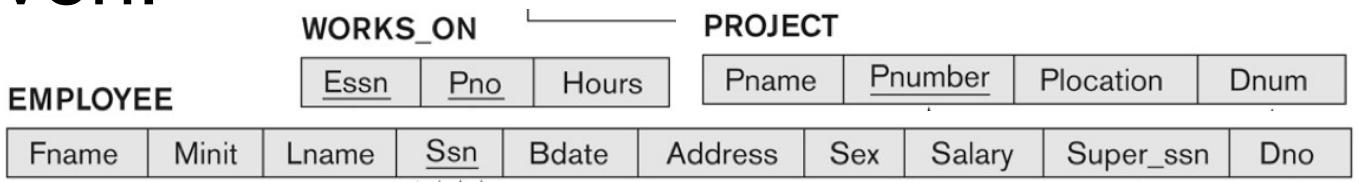


# Heur. optim. – Voorbeeld

Initiële query-boom. Gegeven:

* e aantal tupels in EMPLOYEE
* w aantal tupels in WORKS_ON
* p aantal tupels in PROJECT

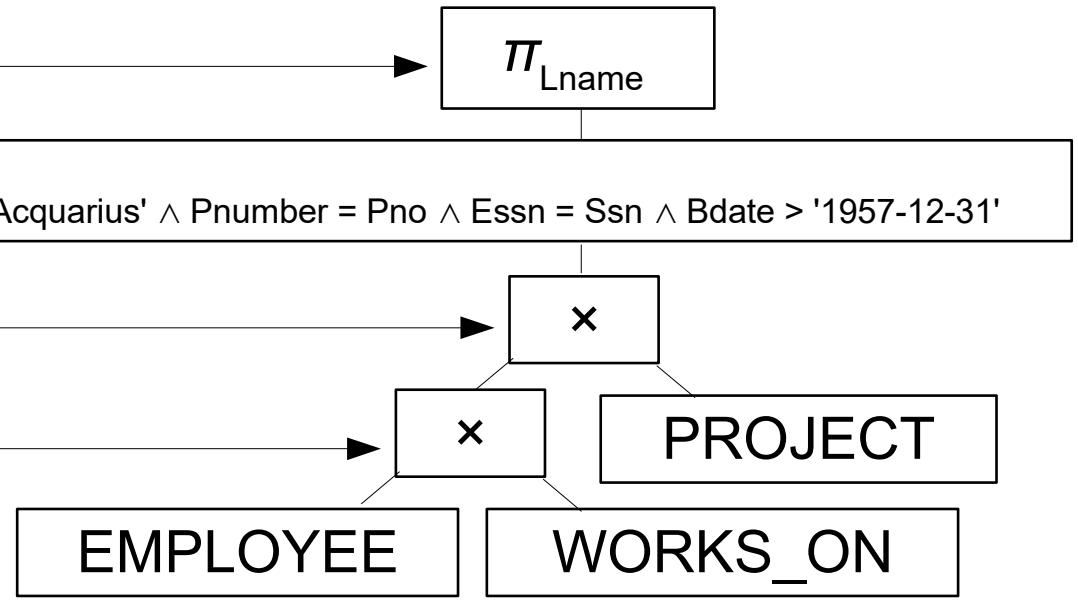



**y ≤ x ≤ e*w*p** met **1 attribuut**

**x ≤ e*w*p** met **17 attributen**

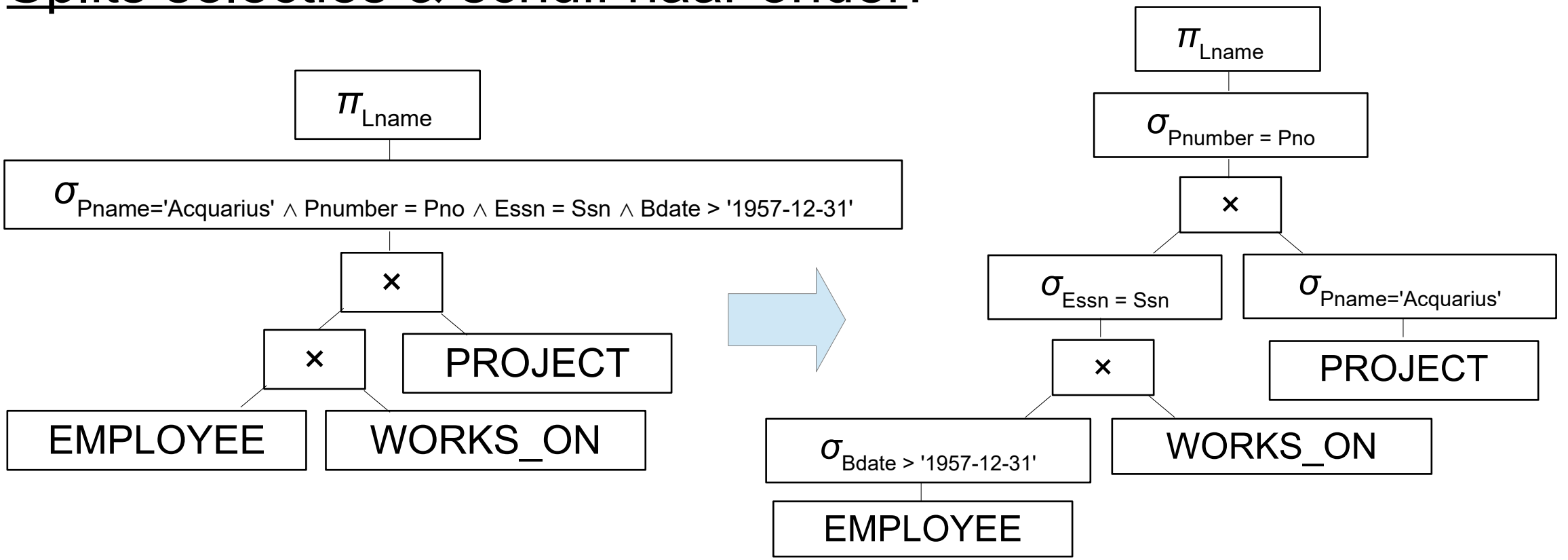
**e*w*p** met **17 attributen**

**e*w** met **13 attributen**

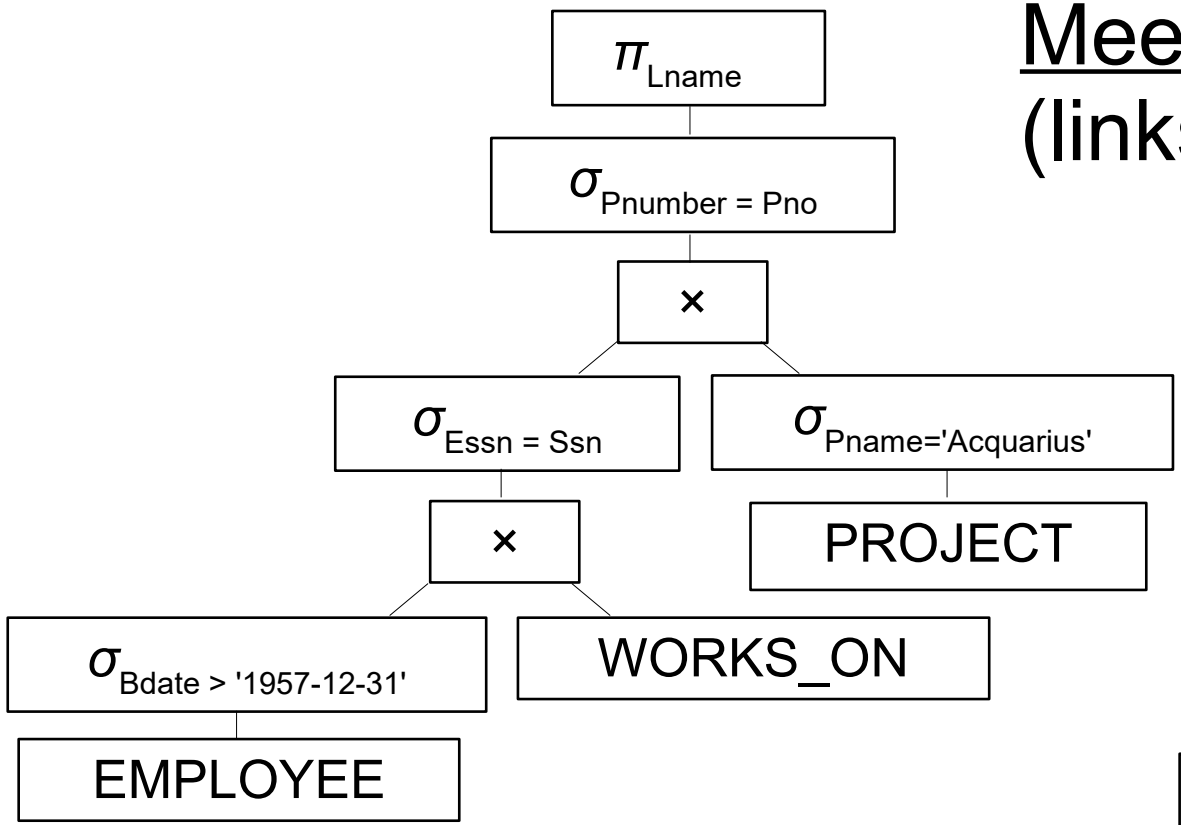


# Heur. optim. – Voorbeeld

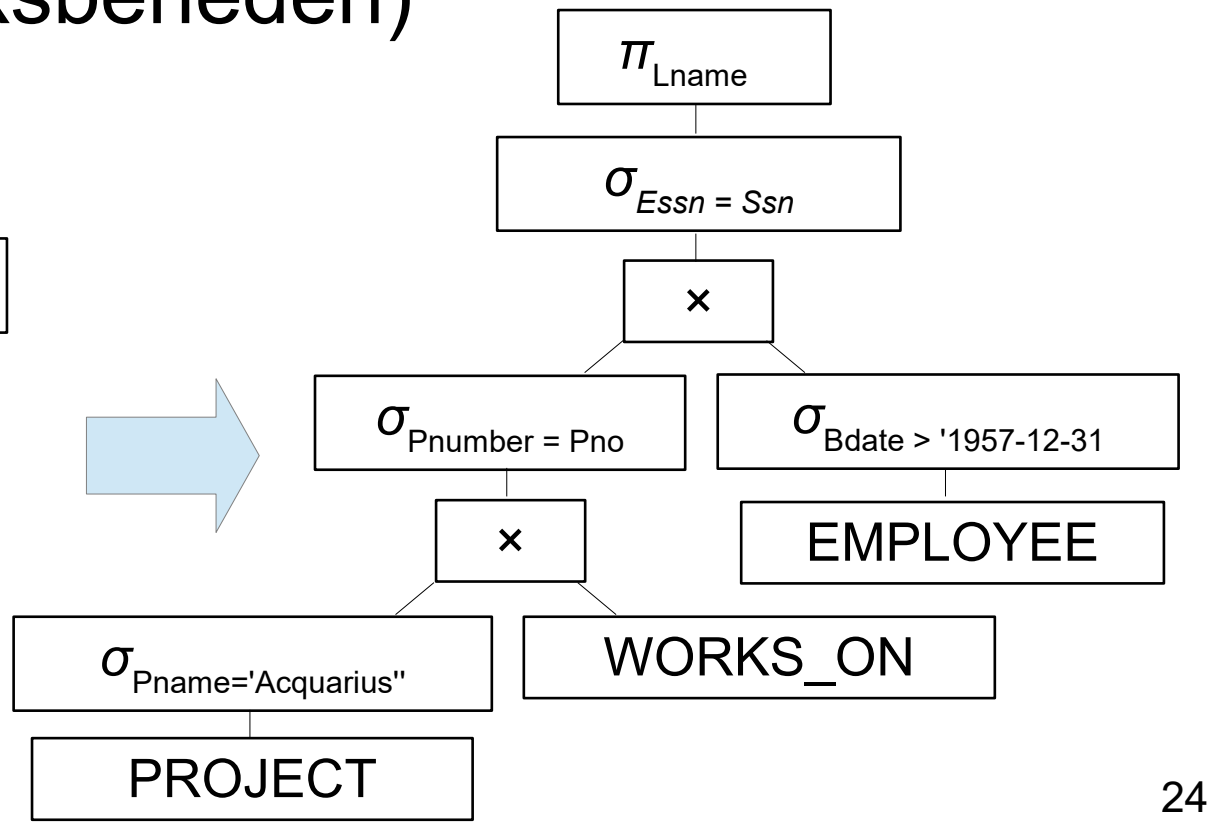
Splits selecties & schuif naar onder:



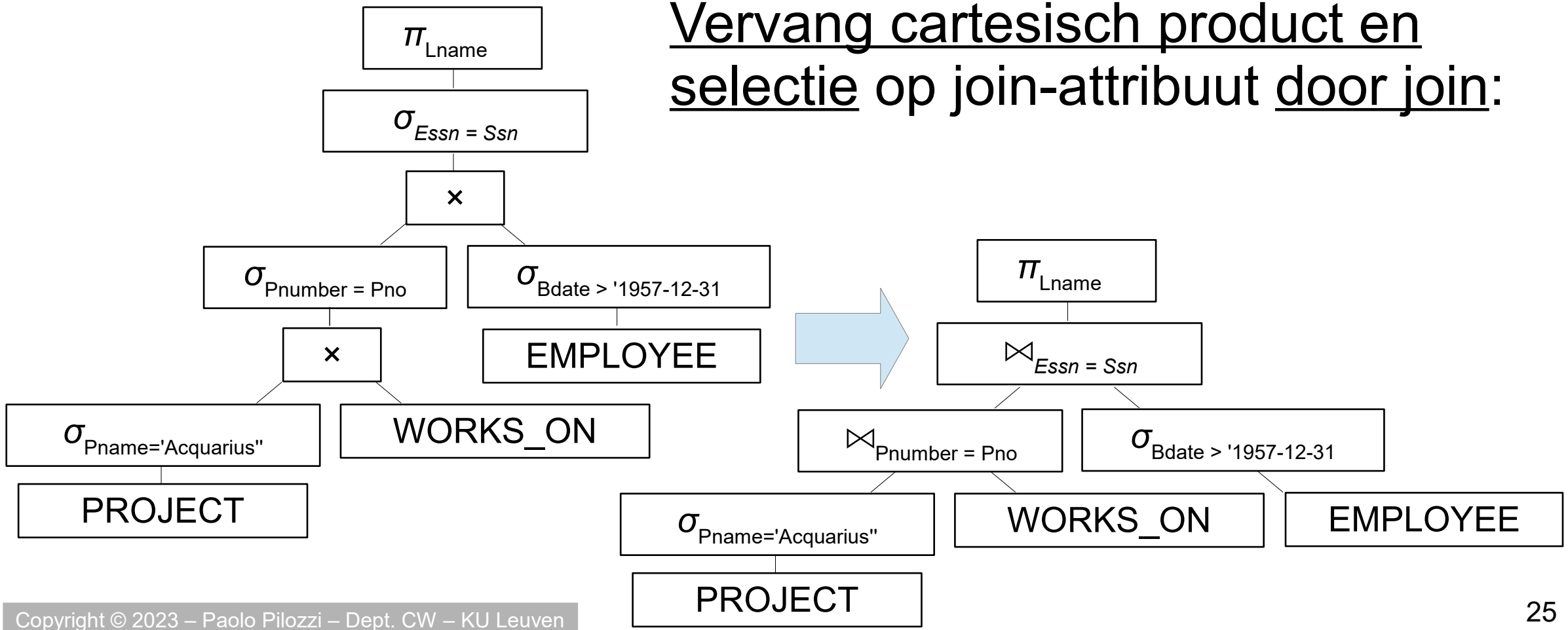
# Heur. optim. – Voorbeeld




Meest restrictieve selectie eerst:
(linksbeneden)

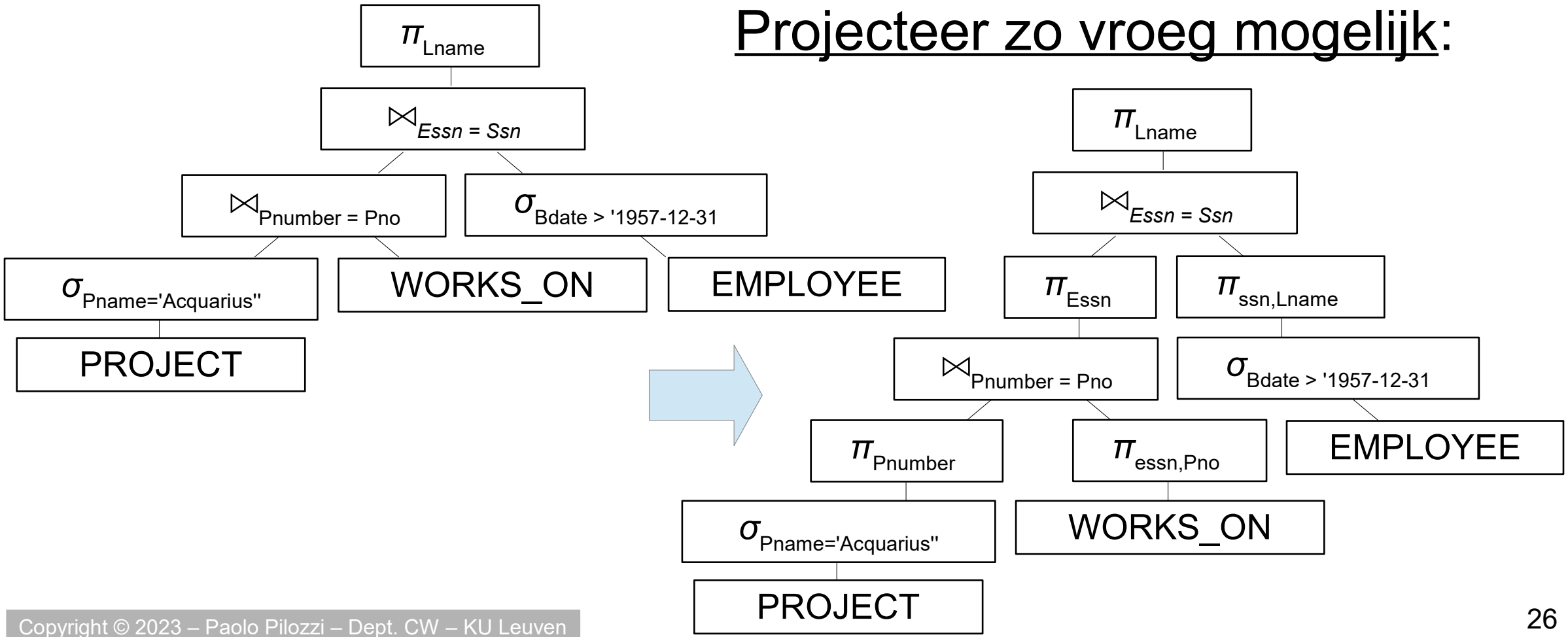


# Heur. optim. – Voorbeeld

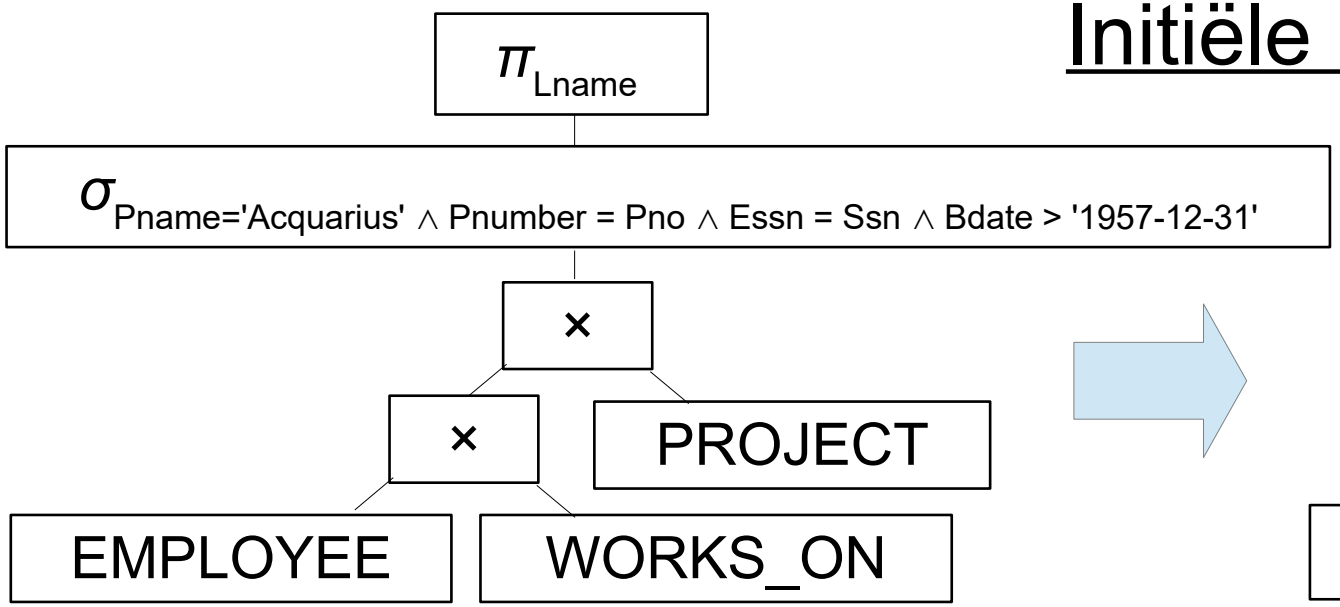



Vervang cartesisch product en selectie op join-attribuut door join:

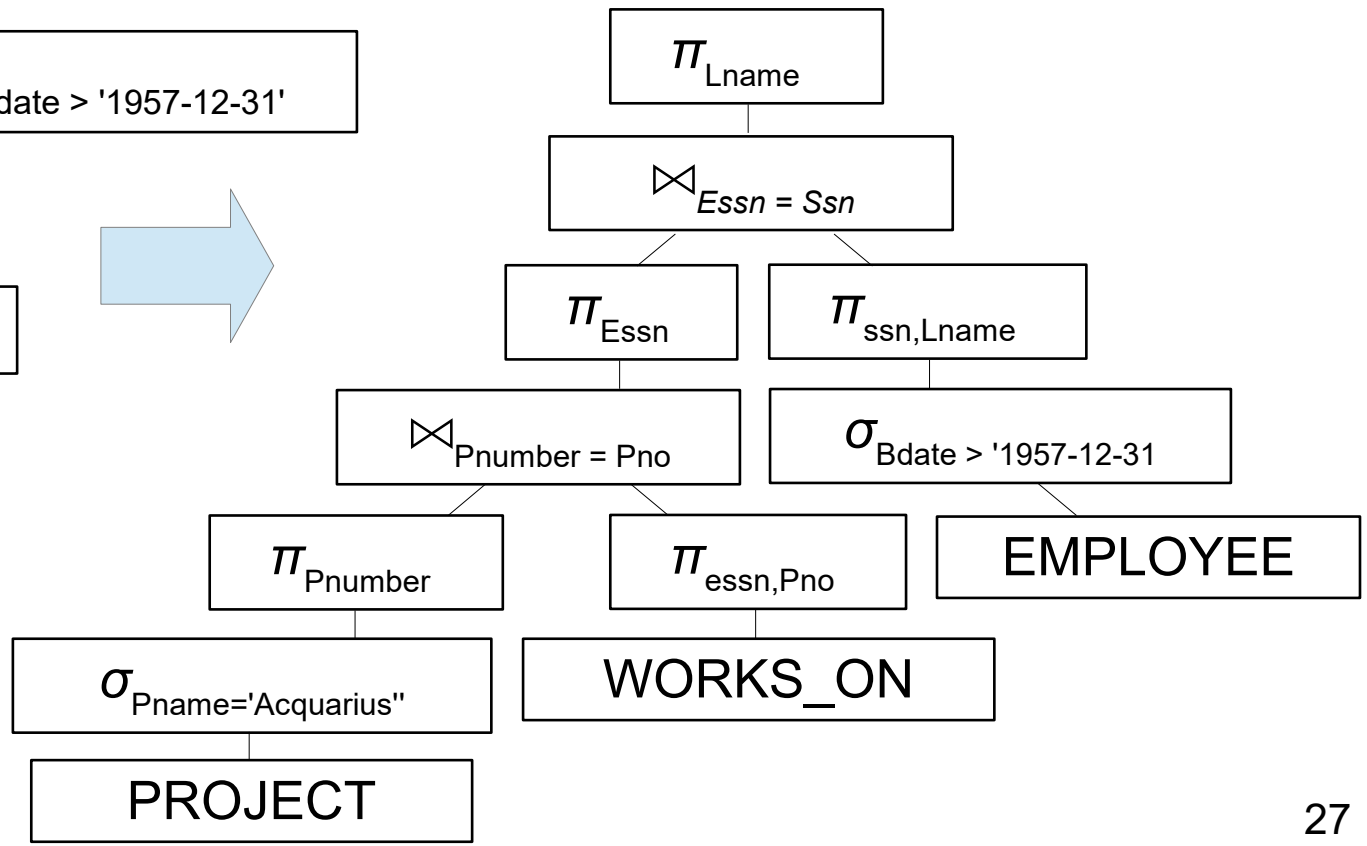
# Heur. optim. – Voorbeeld



# Heur. optim. – Voorbeeld



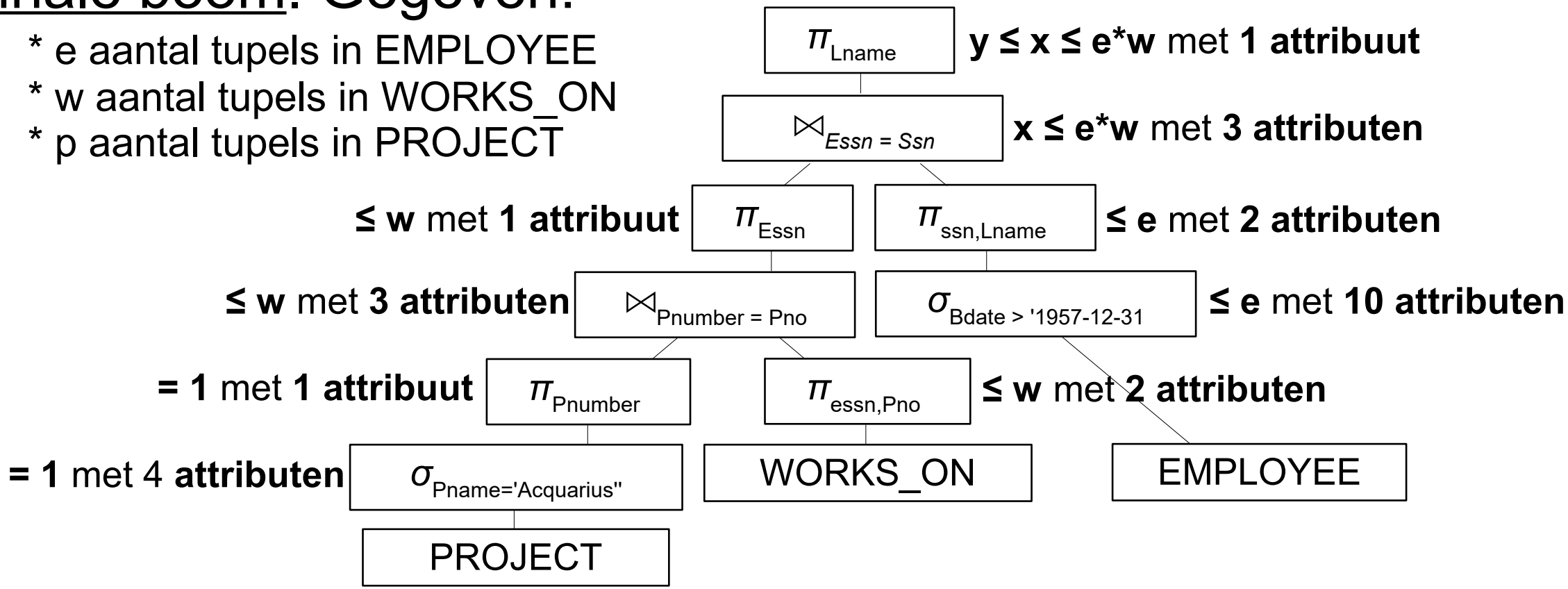

Initiële boom naar finale boom:



# Heur. optim. – Voorbeeld

Finale boom. Gegeven:

* e aantal tupels in EMPLOYEE
* w aantal tupels in WORKS_ON
* p aantal tupels in PROJECT

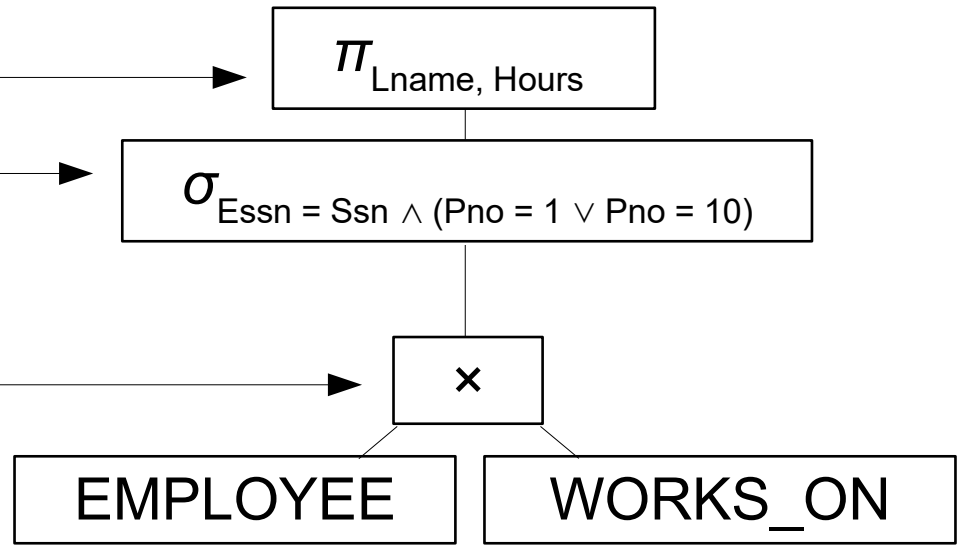


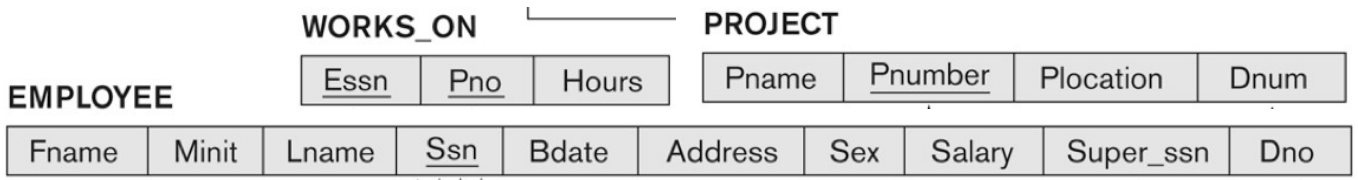

*Merk op: Minder attributen met Natural JOIN!*

# Heur. optim. – Voorbeeld

Initiële query-boom maken:

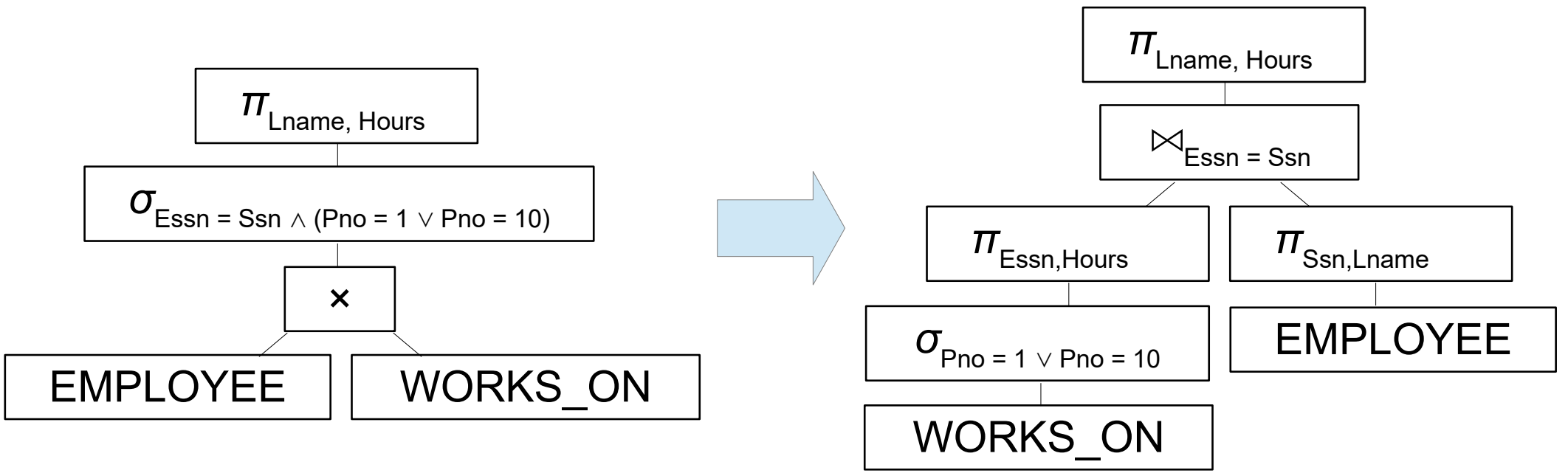
```
SELECT   Lname, Hours
FROM     EMPLOYEE, WORKS_ON
WHERE    Essn = Ssn AND
         (   Pno = 1 OR
             Pno = 10 );
```



# Heur. optim. – Voorbeeld

Initiële naar finale query-boom:



# Algemene Transformatiereges

**σ-cascade**: $\sigma_{c1 \wedge c2 \wedge \cdots \wedge cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$

**σ-commutativiteit:** $\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$

***π-cascade:*** $\pi_{l1}(\pi_{l2}(\ldots(\pi_{ln}(R))\ldots)) \equiv \pi_{l1}(R)$

***σ-π-switch:*** $\pi_{A1,A2,\ldots,An}(\sigma_c(R)) \equiv \sigma_c(\pi_{A1,A2,\ldots,An}(R))$
Met c bevat enkel attributen in {A1,A2,…,An}

# Algemene Transformatiereges

**⋈/×-commutativiteit:**

$$R \bowtie S \equiv S \bowtie R$$
$$R \times S \equiv S \times R$$

**σ-⋈/×-switch:**

$$\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S$$
$$\sigma_c(R \times S) \equiv \sigma_c(R) \times S$$

Met c bevat enkel attributen in R.

$$\sigma_{c1 \wedge c2}(R \bowtie S) \equiv \sigma_{c1}(R) \bowtie \sigma_{c2}(S)$$
$$\sigma_{c1 \wedge c2}(R \times S) \equiv \sigma_{c1}(R) \times \sigma_{c2}(S)$$

Met c1 bevat enkel attributen in R en c2 in S.

# Algemene Transformatiereges

**$\pi$-×-switch:** $\pi_{I}(R \times S) \equiv \pi_{I(R)}(R) \times \pi_{I(S)}(S)$

Met I(R) de attributen van I in R en I(S) die in S.

**$\pi$-⋈-switch:** $\pi_{I}(R \bowtie_{c} S) \equiv \pi_{I(R)}(R) \bowtie_{c} \pi_{I(S)}(S)$

Met c bevat enkel attributen in I.

Met I(R) de attributen van I in R en I(S) die in S.

Anders, Algemeen: $\pi_{I}(R \bowtie_{c} S) \equiv \pi_{I}(\pi_{I(R),c(R)}(R) \bowtie_{c} \pi_{I(S),c(S)}(S))$

Met I(R) de attributen van I in R en I(S) die in S.

Met c(R) de attributen van c in R en c(S) die in S.

Voorbeeld: $\pi_{R.A,S.D}(R \bowtie_{B=C} S) \equiv \pi_{R.A,S.D}(\pi_{A,B}(R) \bowtie_{B=C} \pi_{C,D}(S))$

# Algemene Transformatiereges

**∩-Commutativiteit**: $R \cap S \equiv S \cap R$

**∪-Commutativiteit**: $R \cup S \equiv S \cup R$

**∩/∪/⋈/×-Associativiteit**:

$(R \cap S) \cap T \equiv R \cap (S \cap T)$

$(R \cup S) \cup T \equiv R \cup (S \cup T)$

$(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$

$(R \times S) \times T \equiv R \times (S \times T)$

**σ-∩/–/∪-distributiviteit:**

$\sigma_c(R \cap S) \equiv \sigma_c(R) \cap \sigma_c(S)$

$\sigma_c(R - S) \equiv \sigma_c(R) - \sigma_c(S)$

$\sigma_c(R \cup S) \equiv \sigma_c(R) \cup \sigma_c(S)$

# Algemene Transformatiereges

**π-∪-distributiviteit:** $\pi_l(R \cup S) \equiv \pi_l(R) \cup \pi_l(S)$

**σ-×-naar-⋈:** $\sigma_c(R \times S) \equiv R \bowtie_c S$

**Condities (join & selectie) transformeren met wetten van De Morgan.**

$\neg(A \wedge B) \equiv (\neg A) \vee (\neg B)$
$\neg(A \vee B) \equiv (\neg A) \wedge (\neg B)$

**...**

=> Wat hebben we van deze regels al toegepast?

# Heuristische optimalisatie

= Het toepassen van transformatieregels die een query-boom omzetten naar een in vele gevallen betere.

1. Beperken op tupels in relaties
   * Splits conjunctie in selecties
   * Schuif selecties zoveel mogelijk naar beneden
     1. Selectie over één relatie => Net boven de relatie
     2. Selectie over twee relaties => Net boven hun cartesisch product
   * Kleinere relaties zoveel mogelijk naar links
     - Hou join condities bij cartesische producten!

# Heuristische optimalisatie

= Het toepassen van transformatieregels die een query-boom omzetten naar een in vele gevallen betere.

2. Beperken op grootte product relaties
   * Combineer selectie en join conditie in een join operatie

3. Beperken op attributen in relaties
   * Splits projecties op en projecteer zo vroeg mogelijk
     - Enkel attributen die verder boven nodig zijn bijhouden

# Heuristische optimalisatie

= Het toepassen van transformatieregels die een query-boom omzetten naar een in vele gevallen betere.

4. Beperken op hoeveelheid werk en bloktransfers
    * Identificeer deelbomen die
        - door één algoritme uitgevoerd kunnen worden
        - zonder tijdelijke bestanden

# Heuristische optimalisatie

Wat zijn "kleine" relaties?

=> Hoeveel tupels in een relatie?
=> Hoeveel tupels voldoen aan selectie-join conditie?
=> Hoeveel tupels in join?

Hoe beantwoorden?

* Info over tabellen: aantal tupels + domein attribuut (verdeling)
  => grootte resultaat van operatie inschatten
* Voorbeeld: 500 werknemers verdeeld over 20 departementen
  => Selectie op departement 5 (Dno=5)
  => Schatten met uniforme verdeling: 25 tupels na selectie

# Heuristische optimalisatie

Beter schatten met Histogrammen

=> Meer info over de verdeling van waardes in tupels.

Histogram = groepen van waardes + tupels per groep
met uniforme verdeling binnen elke groep

=> Uniforme verdeling vorige slide: Histogram met 1 groep

Alternatief aantal tupels per groep:

Aantal waardes binnen groep die effectief voorkomen

In praktijk: voor belangrijke attributen (veel in queries)

# Heuristische optimalisatie

Beter schatten met Histogrammen

=> Meer info over de verdeling van waardes in tupels.

Histogram = groepen van waardes + tupels per groep
met uniforme verdeling binnen elke groep

=> Uniforme verdeling vorige slide: Histogram met 1 groep

Twee soorten histogrammen:

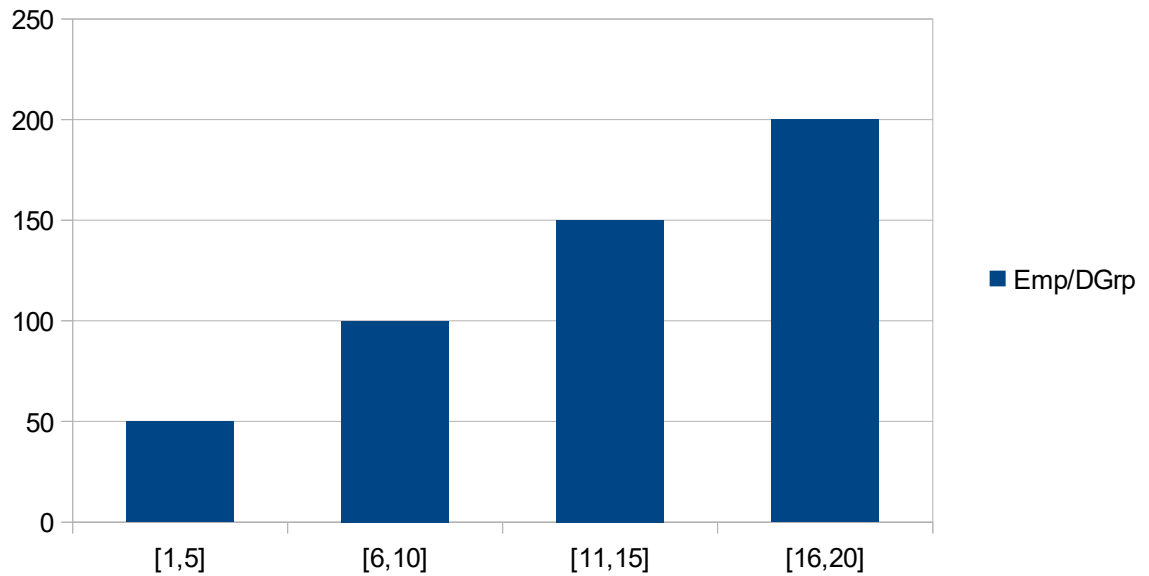
- Equi-depth: Groepen met ongeveer evenveel tupels
- Equi-width: Groepen met ongeveer evenveel waardes

# Heuristische optimalisatie

Voorbeeld Histogrammen: ... WHERE Pnumber = 17 ...

Volgens histogram:
17 in groep [16,20]
150 werknemers in [16,20]

=> 30 in departement 17




Equi-width histogram met aantal werknemers per departementsgroep.

# Heuristische optimalisatie

Schatten met volgende informatie uit de catalogus:

* Aantal tupels in R
* Histogrammen voor attributen van R
* Aantal verschillende waardes, V(A,R), v. attribuut A in R
* Maximale waarde, max(A,R), van A in R
* Minimale waarde, min(A,R), van A in R

=> We gebruiken dit om resultaat operaties in te schatten.

# Heuristische optimalisatie

Schatten grootte operatie – Enkelvoudige Selectie:

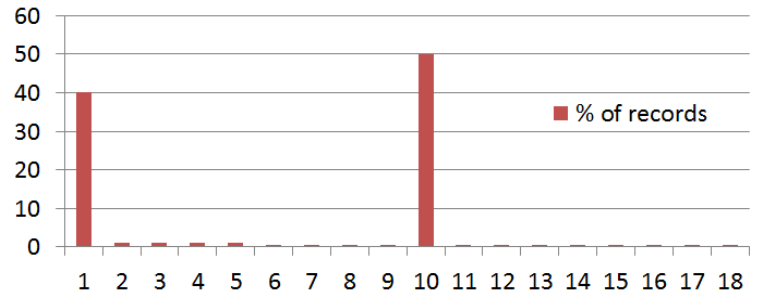
* Voor $\sigma_{A=v}(R)$:
  - Als A sleutel: 1 tupel in resultaat
  - Anders: $n_R/V(A,R)$ tupels in resultaat
* Voor $\sigma_{A\leq v}(R)$:
  => als min(A,R) en max(A,R) in catalogus: aantal tupels is
  - 0 als v < min(A,R)
  - $n_R$ * ((v-min(A,R))/(max(A,R)-min(A,R))) anders
  - geen informatie gegeven: gebruik $n_R/2$

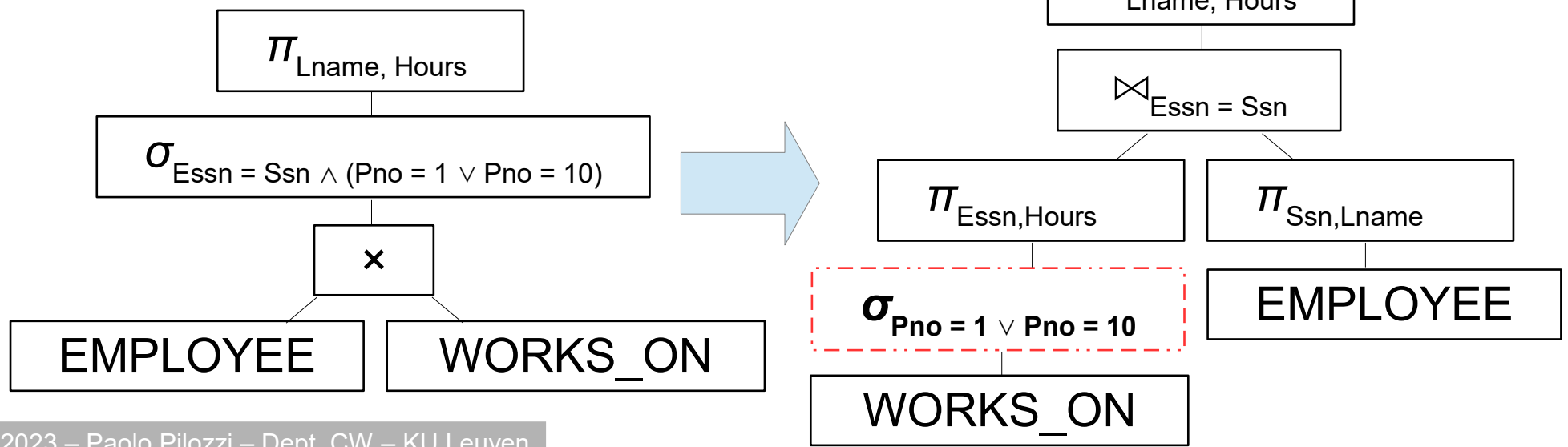
Merk op: Betere schattingen mogelijk met histogram!

# Heuristische optimalisatie

## Voorbeeld – Enkelvoudige Selectie:

1000 records in EMPLOYEE, 5000 in WORKS_ON
18 unieke Pno-waarden in WORKS_ON
=> Zonder histogram: (5000/18)*2 = 556
=> Met histogram: 0.4*5000+0.5*5000 = 4500



# Heuristische optimalisatie

Schatten grootte operatie – Join:

* Voor $\mathbf{R} \bowtie_{\mathbf{R.A=S.B}} \mathbf{S}$:
  - m := min(V(A,R),V(B,S)) bepaalt wat recombineert (= overlap)
    => $n_{R'} := n_R * (m/V(A,R))$ tupels in R die deelnemen in join
    => $n_{S'} := n_S * (m/V(B,S))$ tupels in S die deelnemen in join
  - Voor elk nuttig tupel in R, met a voor A: er zijn $n_{S'}/m$ a's in S
    => Er zijn $n_{R'} * n_{S'}/m$ tupels in de join
  - Hetzij voor elk nuttig tupel in S, met b voor B: Er zijn $n_{R'}/m$ b's in R
    => Er zijn $n_{S'} * n_{R'}/m$ tupels in de join

  <=> $n_R * (m/V(A,R)) * n_S * (m/V(B,S))/m$ <=> $n_R * n_S * (m/(V(A,R)*V(B,S))$

# Heuristische optimalisatie

Schatten grootte operatie – Join:

* Voor $\mathbf{R} \bowtie_{\mathbf{R.A=S.B}} \mathbf{S}$:
  - Overlap m := min(V(A,R),V(B,S))
  - Er zijn $n_R * n_S * (m/(V(A,R)*V(B,S))$ tupels in de join
    => Als enkel A sleutel: $V(A,R) = n_R$ =>
    $n_S * (\min(n_R, V(B,S))/(V(B,S))$ => Veilige schatting: $n_S$
    => Als enkel B sleutel: $V(B,S) = n_S$ =>
    $n_R * (\min(V(A,R), n_S)/(V(A,R))$ => Veilige schatting: $n_R$
    => Als A en B sleutels: $V(A,R) = n_R$ en $V(B,S) = n_S$
    $\min(n_R, n_S)$

# Heuristische optimalisatie

Schatten grootte operatie – Join:

* Voor $\mathbf{R} \bowtie_{\mathbf{R.A=S.B}} \mathbf{S}$:
  - Overlap m := min(V(A,R),V(B,S))
  - Er zijn $n_R * n_S * (m/(V(A,R)*V(B,S))$ tupels in de join
    => A sleutel: $n_S$; B sleutel: $n_R$; A en B sleutels: $\min(n_R,n_S)$.

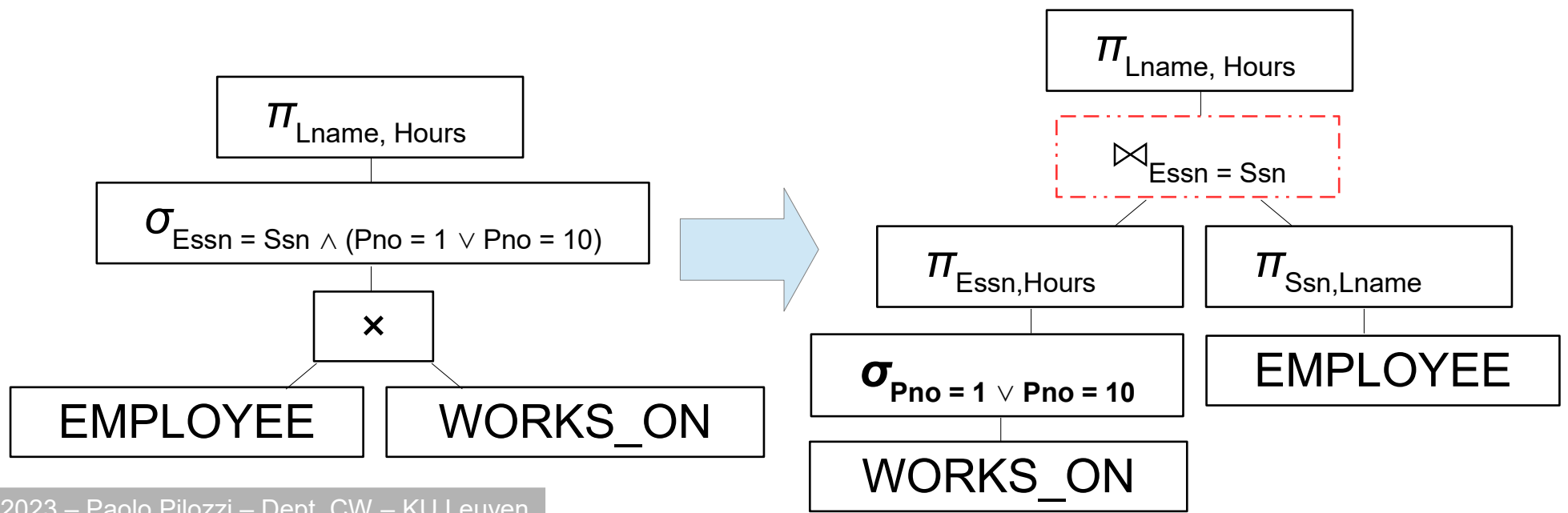
* Voorbeeld: 10000 tupels in R, 5000 tupels in S
  met 100 mogelijke waardes voor R.A en S.B.
  A noch B zijn sleutels. Hoeveel tupels na join, $R \bowtie_{R.A=S.B} S$?

  => 10000*5000*(min(100,100)/(100*100)) = 500000

# Heuristische optimalisatie

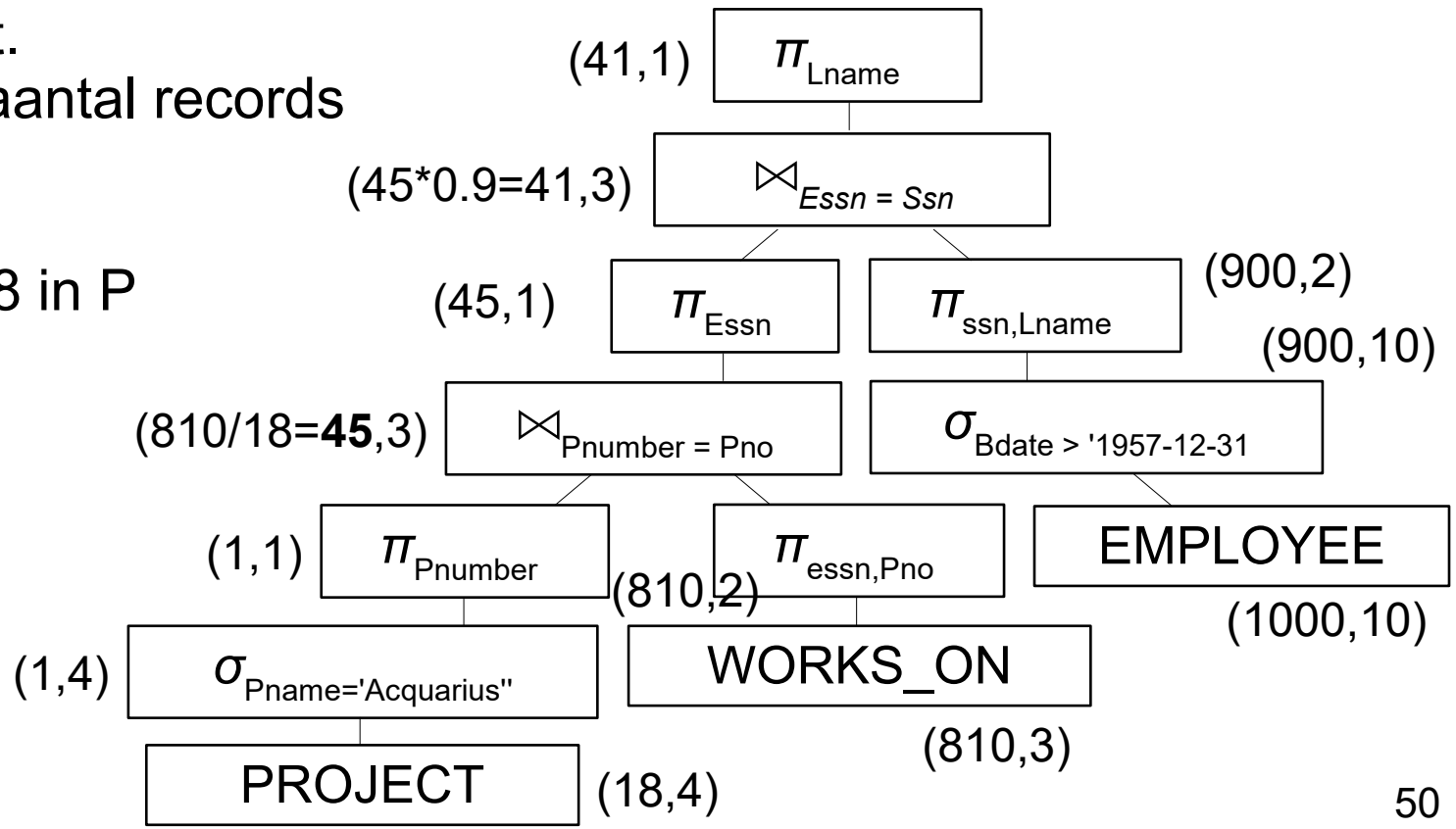
## Voorbeeld – Join:

1000 records in EMPLOYEE, 5000 in WORKS_ON
Na selectie, 4500 in WORKS_ON (zie eerdere slide)
Ssn is een sleutel in EMPLOYEE => 4500 na join



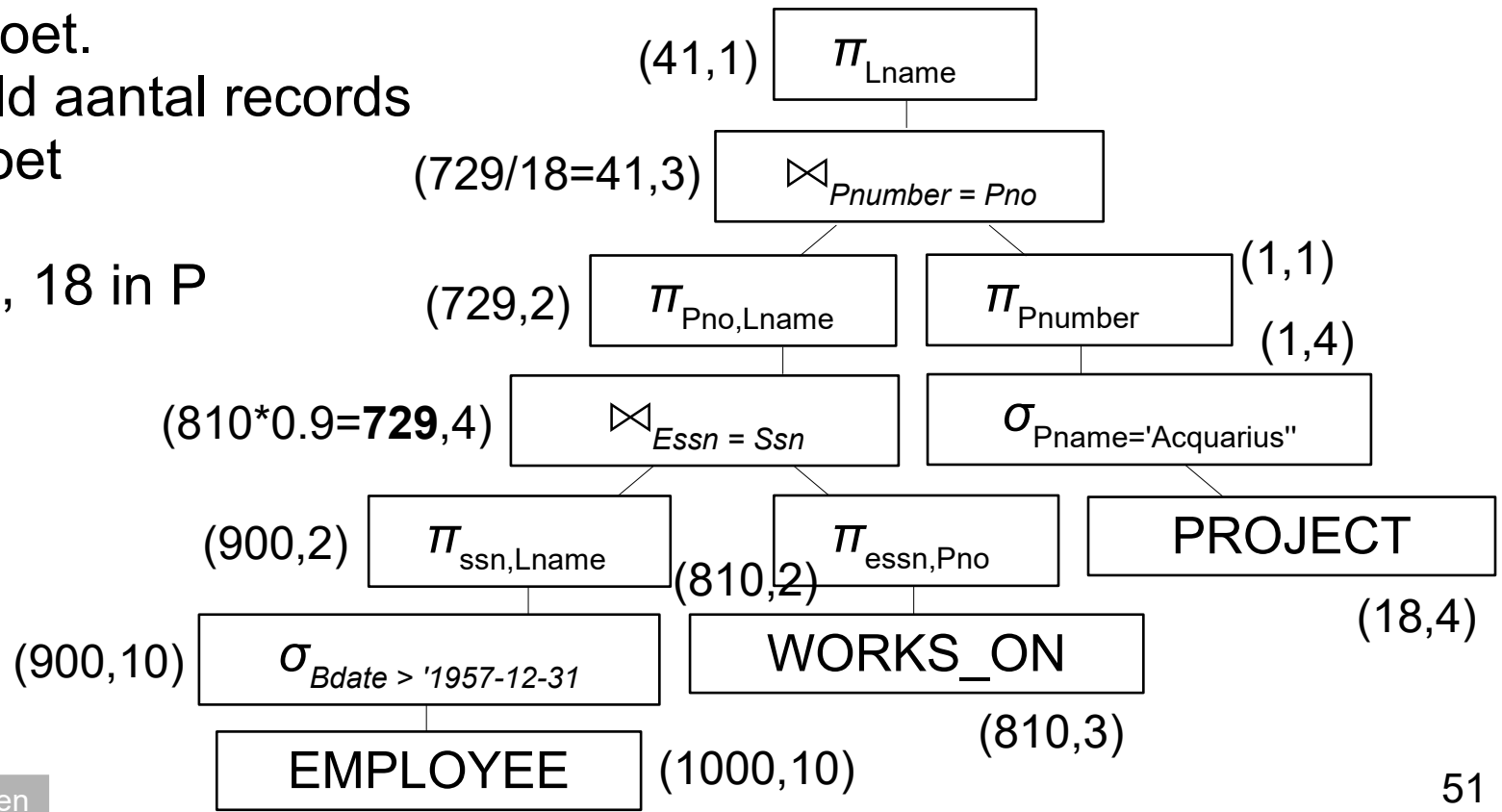
# Heuristische optimalisatie

Waarom "selectievere" ("kleine") relaties links?
Selectiviteit is het percentage records dat aan een bepaald = constraint voldoet.
Selectiecardinaliteit: gemiddeld aantal records die aan een = constraint voldoet
Aannamen:
* 1000 tupels in E, 810 in WO, 18 in P
* Pname uniek
* 90% van werknemers na 1957 geboren
* Uniforme verdeling



# Heuristische optimalisatie

Waarom "selectievere" ("kleine") relaties links?
Selectiviteit is het percentage records dat aan een bepaald = constraint voldoet.
Selectiecardinaliteit: gemiddeld aantal records die aan een = constraint voldoet
Aannamen:
* 1000 tupels in E, 810 in WO, 18 in P
* Pname uniek
* 90% van werknemers na 1957 geboren
* Uniforme verdeling



# Heuristische optimalisatie

Schatten grootte andere operaties:

* Voor $\pi_A R$:
  - $n_R$ met duplicaten, V(A,R) zonder

* Voor verzamelingoperatoren (met duplicaten)
  - $R \cap S$ => $\min(n_R, n_S)$
  - $R - S$ => $n_R$
  - $R \cup S$ => $n_R + n_S$

# Van Query-boom naar Uivoering

Na heuristische optimalisatie:

* Query-boom geeft volgorde van operaties
* Indicatie van deelbomen waarvoor éé algoritme bestaat

Maar nog niet:

* Welke implementaties van operaties gebruiken
* Welke indexen gebruiken
* "Materialized" vs. "Pipelined" evaluatie
  - met/zonder tijdelijke bestanden
  - later meer hierover

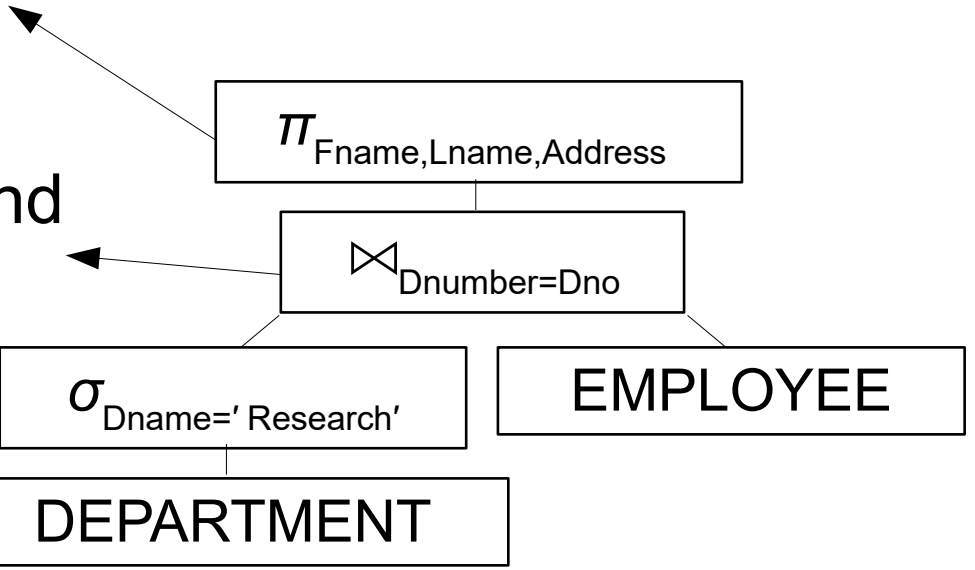
# Voorbeeld Uivoeringsplan

3. Doorloop resultaat van join voor projectie

2. Gebruik geneste lussen voor join
   - Doorloop volledige EMPLOYEE bestand

Of als index op Dno: single-loop join

1. Gebruik index voor selectie op DEPARTMENT (als die bestaat)




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