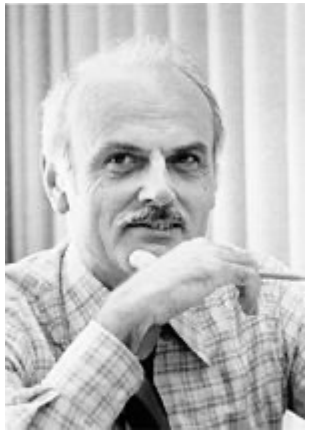
Introduction to Databases
Lecture 3:
The relational algebra

Floris Geerts

Relational Algebra

- Relational Algebra (RA) is a query language for relational databases
  - *Procedural* in nature
  - *Closed*: takes set of relations as input, produces a relation as output; fully compositional

- First step in query optimization:
  - Declarative SQL to procedural RA
  - Rewrite RA using algebraic identities
  - Make physical query execution plan


Edgar F. Codd

Universiteit Antwerpen

Outline

- Relational algebra (RA) operators
  - Set operations: union, intersection, difference
  - Basic Operations: Selection, Projection, Cartesian product
  - Renaming
  - Natural join, θ-join, and division

- Algebraic identities

Set Operations

- Let R and S be two relations with the same schema
- $R \cap S$, $R \cup S$, $R - S$ denote the usual set operations
  - Schema of the RA expression = schema of R = schema of S

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

S(A,B,C)

A	B	C
1	2	2
2	3	4
5	6	7

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

A	B	C
1	2	4
5	6	7

Universiteit Antwerpen

Selection

- Let R be a relation over schema (A_1, …, A_n) and θ an expression over A_1, …, A_n
- $\sigma_\theta (R) := \{ t \in R \mid \theta \text{ holds} \}$
  - Relation over schema (A_1, …, A_n) with all tuples that satisfy θ.

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

$\sigma_{B=C \vee A=2}$ (R)

A	B	C
1	2	2
2	3	4

Universiteit Antwerpen

Projection

- Let R be a relation and $B_1, \dots B_k$ attributes of R
- $\pi_{B1, \dots Bk}(R) := \{ (t(B_1), \dots, t(B_k)) \mid t \in R \}$
  - Relation over schema $(B_1, \dots B_k)$ that contains for every tuple t in R, the tuple $(t(B_1), \dots, t(B_k))$

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

$\pi_{A,B}(R)$

A	B
1	2
2	3

Examples

- Infamous beer drinkers example
  - Likes(Drinker, Beer)
  - Serves(Pub, Beer)
  - Visits(Drinker, Pub)

Give all drinkers who like "Guiness"

$$\pi_{\text{Drinker}} (\sigma_{\text{Beer}=\textit{"Guiness"}} (\text{Likes}))$$

Give all drinkers who visit "Irish pub" or like "Guiness"

$$(\pi_{\text{Drinker}} (\sigma_{\text{Beer}=\textit{"Guiness"}} (\text{Likes}))) \cup (\pi_{\text{Drinker}} (\sigma_{\text{Pub}=\textit{"Irish pub"}} (\text{Visits})))$$

Likes

Drinker	Beer
Jan	Guiness
Jan	Hoegaarden
Piet	Guiness
Kees	Palm

Serves

Pub	Beer
Irish pub	Guiness
Irish pub	Hoegaarden
Irish pub	Palm
Bar Baar	Heineken

Visits

Drinker	Pub
Jan	Irish pub
Piet	Bar Baar
Jan	Bar Baar

Universiteit Antwerpen

Examples

Give all beers served by "Irish pub"

$$\pi_{\text{Beer}}\ (\sigma_{\text{Pub}=\textit{"Irish pub"}}\ (\text{Serves}))$$

Give all drinkers who visit "Irish pub" and at least one other pub

$$(\pi_{\text{Drinker}}\ (\sigma_{\text{Pub}=\textit{"Irish pub"}}\ (\text{Visits})))$$
$$\cap\ (\pi_{\text{Drinker}}\ (\sigma_{\text{Pub}<>\textit{"Irish pub"}}\ (\text{Visits})))$$

Likes

Drinker	Beer
Jan	Guiness
Jan	Hoegaarden
Piet	Guiness
Kees	Palm

Serves

Pub	Beer
Irish pub	Guiness
Irish pub	Hoegaarden
Irish pub	Palm
Bar Baar	Heineken

Visits

Drinker	Pub
Jan	Irish pub
Piet	Bar Baar
Jan	Bar Baar

Examples

- Likes(Drinker, Beer)
- Serves(Pub, Beer)
- Visits(Drinker, Pub)

Give all drinkers who do not like "Guiness"

Is the following query correct?

$$\pi_{\text{Drinker}} (\sigma_{\text{Beer} <> \text{"Guiness"}} (\text{Likes}))$$

Universiteit Antwerpen

Examples

- Likes(Drinker, Beer)
- Serves(Pub, Beer)
- Visits(Drinker, Pub)

Give all drinkers who do not like "Guiness"

The following query is **NOT correct**

$$\pi_{\text{Drinker}} (\sigma_{\text{Beer} <> \text{“}\textit{Guiness}\text{”}} (\text{Likes}))$$

Likes

Drinker	Beer
John	Guiness
John	Chimay
George	Guiness
Mary	Hoegaarden
Mary	Chimay

$\sigma_{\text{Beer} <> \text{“}\textit{Guiness}\text{”}}$ Likes

Drinker	Beer
John	Chimay
Mary	Hoegaarden
Mary	Chimay

$\pi_{\text{Drinker}}\ \sigma_{\text{Beer} <> \text{“}\textit{Guiness}\text{”}}$ Likes

Drinker
John
Mary

Universiteit Antwerpen

Examples

- Likes(Drinker, Beer)
- Serves(Pub, Beer)
- Visits(Drinker, Pub)

Give all drinkers who do not like "Guiness"

$$\underbrace{\boxed{(\ \pi_{\text{Drinker}}(\text{Likes}) \cup \pi_{\text{Drinker}}(\text{Visits}))}}_{\text{All drinkers}} - \underbrace{\boxed{(\pi_{\text{Drinker}}(\sigma_{\text{Beer} = \textit{"Guiness"}}(\text{Likes})))}}_{\text{Drinkers who like Guiness}}$$

Universiteit Antwerpen

Cartesian Product

- Let R and S be two relations with disjoint schemas $(A_1,\ldots,A_n)$ and $(B_1,\ldots B_m)$ respectively
- $R \times S := \{ (t(A_1), \ldots, t(A_n), s(B_1), \ldots, s(B_m)) \mid t \in R, s \in S \}$
    - Relation over Schema(R) $\cup$ Schema(S) containing any combination of a tuple from R and a tuple from S

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

T(D,E)

D	E
a	b
c	d

$R \times T$

A	B	C	D	E
1	2	2	a	b
1	2	2	c	d
1	2	4	a	b
1	2	4	c	d
2	3	4	a	b
2	3	4	c	d

Renaming

- Let R be a relation, A an attribute of R, and B an attribute not in the schema of R.
- $\rho_{B/A}$ (R) denotes the relation with the same tuples as R, but with attribute A renamed to B.

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

$\rho_{D/C}$ (R)

A	B	D
1	2	2
1	2	4
2	3	4

Universiteit Antwerpen

Examples

- Likes(Drinker, Beer)
- Serves(Pub, Beer)
- Visits(Drinker, Pub)

Give all drinkers who visit a bar that serves "Guiness"

$$\pi_{\text{Drinker}} (\sigma_{\text{GPub=Pub}} (\text{Visits} \times (\rho_{\text{GPub/Pub}} (\sigma_{\text{Beer} = \textit{"Guiness"}} (\text{Serves})))))$$

Give all drinkers who visit a bar that serves a beer they like

$$\pi_{\text{Drinker}} (\sigma_{\text{VP=SP} \wedge \text{SB=Beer} \wedge \text{Drinker = VD}} (\text{Likes} \times (\rho_{\text{SP/Pub, SB/Beer}} (\text{Serves})) \times \rho_{\text{VD/Drinker, VP/Pub}} (\text{Visits})))$$

=

$$\pi_{\text{Drinker}} (\sigma_{\text{SB=Beer} \wedge \text{Drinker = VD}} ((\text{Likes}) \times (\sigma_{\text{VP=SP}} ((\rho_{\text{SP/Pub, SB/Beer}} (\text{Serves})) \times (\rho_{\text{VD/Drinker, VP/Pub}} (\text{Visits})))))$$

Universiteit Antwerpen

Examples

- Likes(Drinker, Beer)
- Serves(Pub, Beer)
- Visits(Drinker, Pub)

Give all beers no one likes

$$(\pi_{Beer}\,(Serves)) - (\pi_{Beer}\,(Likes))$$

Give all pubs that serve at least one beer no none likes

$$\pi_{Pub}(((\pi_{Pub}\,(Serves)) \times (\pi_{Beer}\,(Serves) - \pi_{Beer}\,(Likes))) \cap Serves)$$

Give all pubs that only serve beers no one likes

$$\pi_{Pub}\,(Serves) - \pi_{Pub}\,\sigma_{Beer=SBeer}\,(\rho_{Beer/SBeer}\,Serves \times \pi_{Beer}\,Likes)$$

Universiteit Antwerpen

Natural Join

- Combining two relations often involves equality on attributes
- Let R and S be relations and let $C_1, \dots, C_k$ be the attributes they have in common
- The natural join of R and S, denoted $R \bowtie S$ is:

$$\pi_{\text{schema}(R) \cup \text{schema}(S)}\, \sigma_{C1=SC1 \wedge \dots \wedge Ck=SCk}\, (R \times \rho_{SC1/C1, \dots, SCk/Ck}\, S)$$

  - Relation with schema schema(R) ∪ schema(S) containing a tuple for any pair of tuples of R and S that agree on their common attributes.

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

U(C,E)

C	E
1	b
4	d

$R \bowtie U$

A	B	C	E
1	2	4	d
2	3	4	d

Universiteit Antwerpen

θ-Join

- Let R and S be relations that have no attributes in common, and let θ be an expression over schema(R) ∪ schema(S)
- The join of R and S with condition θ is:
  $R \bowtie_\theta S := \sigma_\theta(R \times S)$
  - Relation with schema schema(R) ∪ schema(S) containing a tuple for any pair of tuples of R and S that satisfy θ.

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

V(D,E)

D	E
1	b
4	d

$R \bowtie_{C=D} V$

A	B	C	D	E
1	2	4	4	d
2	3	4	4	d

Universiteit Antwerpen

Examples Revisited

- Likes(Drinker, Beer)
- Serves(Pub, Beer)
- Visits(Drinker, Pub)

Give all drinkers who visit a pub that serves "Guiness"

$\pi_{\text{Drinker}}\ \sigma_{\text{GPub=Pub}}\ (\text{Visits} \times (\rho_{\text{GPub/Pub}}\ \sigma_{\text{Beer}=\textit{"Guiness"}}\ \text{Serves}))$

$\pi_{\text{Drinker}}\ (\text{Visits} \bowtie \sigma_{\text{Beer}=\textit{"Guiness"}}\ \text{Serves})$

Give all drinkers who visit a pub that serves a beer they like

$\pi_{\text{Drinker}}\ \sigma_{\text{VP=SP} \wedge \text{SB=Beer} \wedge \text{Drinker = VD}}$
$(\text{Likes} \times \rho_{\text{SP/Pub, SB/Beer}}\ \text{Serves} \times \rho_{\text{VD/Drinker, VP/Pub}}\ \text{Visits})$

$\pi_{\text{Drinker}}\ \text{Likes} \bowtie \text{Serves} \bowtie \text{Visits}$

Division

- Relations R and S, schema(S) $\subseteq$ schema(R)
- The division R $\div$ S is the largest relation T such that S $\times$ T $\subseteq$ R
  - R $\div$ S has attributes schema(R) - schema(S)
  - $(t_1,\dots,t_k)$ is in R $\div$ S if and only if for every $(s_1, \dots, s_m) \in S$, the tuple $(t_1, \dots, t_k, s_1, \dots, s_m) \in R$

Example

R(A,B,C)

A	B	C
1	2	2
1	2	4
2	3	4

W(C)

C
2
4

R $\div$ W

A	B
1	2

Universiteit Antwerpen

Examples

- Likes(Drinker, Beer)
- Serves(Pub, Beer)
- Visits(Drinker, Pub)

Give all drinkers who visit all pubs (that are visited by at least one drinker)

$$\text{Visits} \div (\pi_{\text{Pub}}\ \text{Visits})$$

Give all pubs that serve all beers that George likes

$$\text{Serves} \div (\pi_{\text{Beer}}\ \sigma_{\text{Drinker=“George”}}\ \text{Likes})$$

Universiteit Antwerpen

Expressing Division

- Division can be expressed using the other operators:
  - Let schema(R) - schema(S) = $\{C_1, \ldots, C_k\}$

$$(\pi_{C1,\ldots,Ck}\ R) - \pi_{C1,\ldots,Ck}([(\pi_{C1,\ldots,Ck} R) \times S] - R)$$

Example

$$\text{Visits} \div (\pi_{Pub}\ \text{Visits})$$

is short for:

$$\pi_{Drinker}\ \text{Visits} - \pi_{Drinker}([(\pi_{Drinker}\ \text{Visits}) \times (\pi_{Pub}\ \text{Visits})] - \text{Visits})$$

Universiteit Antwerpen

Outline

- Relational algebra (RA) operators
  - Set operations: union, intersection, difference
  - Basic Operations: Selection, Projection, Cartesian product
  - Renaming
  - Natural join, θ-join, and division

- Algebraic identities

Universiteit Antwerpen

Expression tree

- A relation algebra query can straightforwardly be represented as an *expression tree*
- This tree represents the order in which the operations are executed and as such represent a high-level *query plan*
- By rearranging the operators in the tree we may create an equivalent yet more efficient execution plan
  - Called query rewriting

- Allowable rewritings: expressed by *algebraic identities*

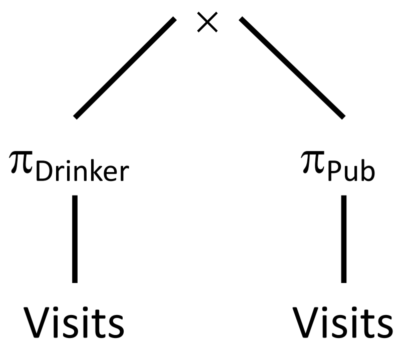
Universiteit Antwerpen

Example Expression Tree

$\pi_{\text{Drinker}}\,\text{Visits} - \pi_{\text{Drinker}}\left([(\pi_{\text{Drinker}}\,\text{Visits}) \times (\pi_{\text{Pub}}\,\text{Visits})] – \text{Visits}\right)$

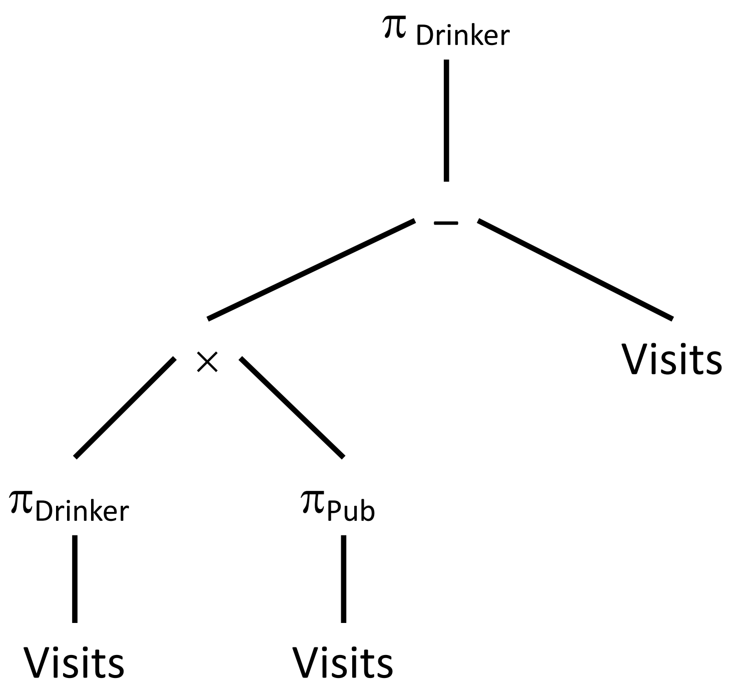
Example Expression Tree

$\pi_{\text{Drinker}}\,\text{Visits} - \pi_{\text{Drinker}}\big(\mathbf{[(\pi_{Drinker}\ Visits) \times (\pi_{Pub}\ Visits)]} - \text{Visits}\big)$



Example Expression Tree

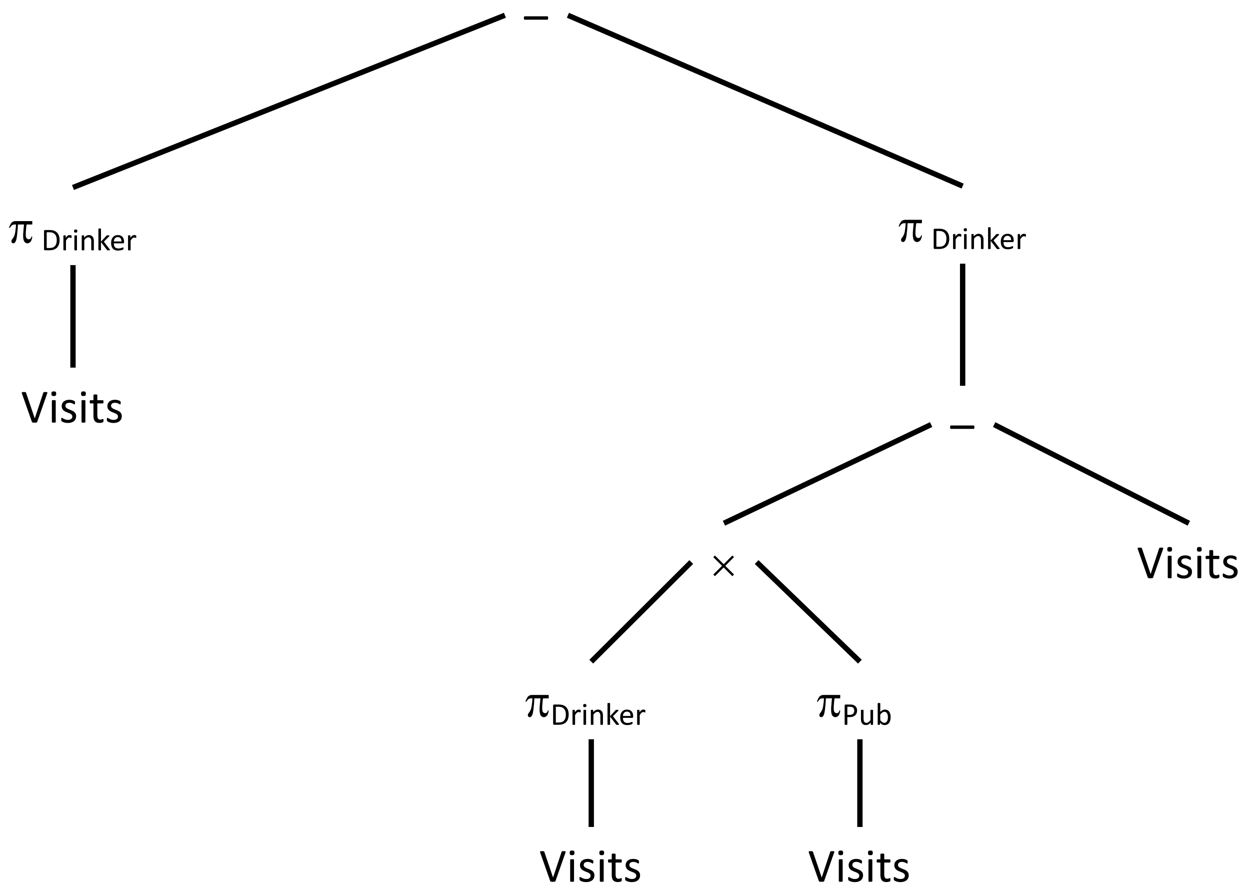
π_{Drinker} Visits - **π_{Drinker} ([(π_{Drinker} Visits) × (π_{Pub} Visits)] – Visits)**



Universiteit Antwerpen

Example Expression Tree

π_{Drinker} **Visits** - π_{Drinker} **([(π_{Drinker} Visits) × (π_{Pub} Visits)] – Visits)**



Algebraic Identities

- Let R(A,B,C), S(A,B,C), U(D,E) be relation schemas. Some example identies:

$$\pi_{A,D}(R \times U) = (\pi_{A} R) \times (\pi_{D} U)$$

$$\pi_{A,B}(R \cup S) = (\pi_{A,B} R) \cup (\pi_{A,B} S)$$

$$\sigma_{A=B}(R \bowtie_{C=D} U) = (\sigma_{A=B} R) \bowtie_{C=D} U$$

$$\sigma_{B=5 \wedge A<>E}(R \bowtie_{C=D} U) = \sigma_{A<>E}((\sigma_{B=5} R) \bowtie_{C=D} U)$$

- Such rules can be used to reorder the expression tree
  - For instance: push down selections/projections

Summary

- Relational algebra as a procedural query language for relational databases
  - Closed: input are relations and output is again a relation
  - Fully compositional

- Main operations:
  - Set operations $\cap$, $\cup$, -
  - Selection σ, projection π, Cartesian product $\times$
  - Renaming ρ
- Syntactic sugar:
  - Natural join $\bowtie$ and theta join $\bowtie_{\theta}$
  - Division $\div$

Universiteit Antwerpen