B0B36DBS, BD6B36DBS: **Database Systems**

http://www.ksi.mff.cuni.cz/~svoboda/courses/192-B0B36DBS/

Lecture 7

Relational Algebra

Martin Svoboda

martin.svoboda@fel.cvut.cz

31. 3. 2020

Czech Technical University in Prague, Faculty of Electrical Engineering

Lecture Outline

Relational algebra

- **Operations**: syntax, semantics and examples
  - Selection, projection, attribute renaming
  - Cartesian product, natural join, theta join, …
  - Division
  - Outer join
- **Relational completeness**

Relational Model

Relational model

- Logical model where all data is represented in terms of **tuples** (rows) that are grouped into **relations** (tables)

Schema of a relation

- $S(a_1 : T_1, \ldots, a_n : T_n)$
  - S is a relation name
  - a_i are attribute names, T_i are optional domains (data types)

Relation = data

- Set of **tuples**
- Unordered, no duplicities, without missing values (null), atomic values only (first normal form)

Relational Model

Relation structure revisited

- *Formal definition for the purpose of this lecture…*

$$\boxed{\langle R, A_R \rangle}$$

- R = **set of tuples** = actual data
  - Tuple $t = \{(a_1, v_1), \ldots, (a_n, v_n)\}$, where:
    - $a_i \in A_R$ is an **attribute** name
    - $v_i \in T_i$ is a **value** this attribute is associated with
    - (a_i, v_i) is an attribute **binding**
  - I.e. each tuple acts as a function
    - $t : A_R \to \bigcup_{i=1,\ldots,n} T_i$
    - $t(a_i) = v_i \in T_i$
- A_R = **set of attributes** = schema of a relation
  - We continue to omit the domains T_i

Relation Structure: Example

Sample relation of actors

Actor(name, surname, year)

name	surname	year
Ivan	Trojan	1964
Jiří	Macháček	1966
Jitka	Schneiderová	1973

$$\Big\langle$$
$$\{\ \{(\mathrm{name}, \text{Ivan}), (\mathrm{surname}, \text{Trojan}), (\mathrm{year}, 1964)\},$$
$$\{(\mathrm{name}, \text{Jiří}), (\mathrm{surname}, \text{Macháček}), (\mathrm{year}, 1966)\},$$
$$\{(\mathrm{name}, \text{Jitka}), (\mathrm{surname}, \text{Schneiderová}), (\mathrm{year}, 1973)\}\ \},$$
$$\{\mathrm{name}, \mathrm{surname}, \mathrm{year}\}$$
$$\Big\rangle$$

Query Languages

Formal **query languages** based on the **relational model**

- **Relational algebra**
  - Algebraic expressions with relations and operations on them
  - E.g. names and surnames of all actors born in 1970 or earlier

    $\pi_{\text{name,surname}}(\sigma_{\text{year}\leq 1970}(\text{Actor}))$

    $\text{Actor}(\text{year} \leq 1970)[\text{name}, \text{surname}]$
- **Relational calculi**
  - Expressions based on the **first-order predicate logic**
  - Domain relational calculus
    - E.g. $\{\, (n, s) \mid \exists\, y : \text{Actor}(n, s, y) \wedge y \leq 1970 \,\}$
  - Tuple relational calculus
    - E.g. $\{\, t[\text{name}, \text{surname}] \mid \text{Actor}(t) \wedge t.\text{year} \leq 1970 \,\}$

Query Languages: Terminology

Query expression

- **Expression in a given language describing the intended query**
- Multiple equivalent expressions often exist

Query

- **Actual data** we are attempting to retrieve
  - I.e. result of the evaluation of a given query expression
- E.g. relation, table, ...

Query language

- **Set of all syntactically well-formed query expressions** with respect to a given grammar
- E.g. relational algebra, SQL, ...

Sample Query

First names of all actors born in *1960* or later

$\pi_{\text{name}}(\sigma_{\text{year} \geq 1960}(\text{Actor}))$

$\text{Actor}(\text{year} \geq 1960)[\text{name}]$

name	surname	year
Ivan	Trojan	1964
Jiří	Macháček	1966
Jitka	Schneiderová	1973
Zdeněk	Svěrák	1936
Jitka	Čvančarová	1978

name
Ivan
Jiří
Jitka

Relational Algebra

Inductive construction of RA expressions

- Basic expressions
  - **Relation name**: R
  - **Constant relation**
- General expressions are formed using smaller subexpressions
  - **Projection**: $\pi_{a_1,\ldots,a_n}(E)$
  - **Selection**: $\sigma_{\varphi}(E)$
  - **Attribute renaming**: $\rho_{b_1/a_1,\ldots,b_n/a_n}(E)$
  - **Union**: $E_R \cup E_S$
  - **Difference**: $E_R \setminus E_S$
  - **Cartesian product**: $E_R \times E_S$
  - …

Projection

Projection: preserves only attributes we are interested in

$\pi_{a_1,\ldots,a_n}(E)$ or $E[a_1, \ldots, a_n]$

- $\langle R, A_R \rangle = [\![E]\!]$ is a relation R with attributes A_R
- $a_1, \ldots, a_n$ is a set of **attributes to be preserved**, each $a_i \in A_R$, all the other attributes are to be removed

$$[\![\ \pi_{a_1,\ldots,a_n}(E)\ ]\!] = \langle\ \{t[a_1, \ldots, a_n] \mid t \in R\}\ ,\ \{a_1, \ldots, a_n\}\ \rangle$$

- $t[a_1, \ldots, a_n] = \{(a, v) \mid (a, v) \in t, a \in \{a_1, \ldots, a_n\}\}$ is a restriction of a tuple t to attributes $a_1, \ldots, a_n$
- **Duplicate tuples** in the result are (of course) **suppressed**!

Projection: Example

First names of all actors

$\pi_{\text{name}}(\text{Actor})$ $\text{Actor}[\text{name}]$

name	surname	year
Ivan	Trojan	1964
Jiří	Macháček	1966
Jitka	Schneiderová	1973
Zdeněk	Svěrák	1936
Jitka	Čvančarová	1978

name
Ivan
Jiří
Zdeněk
Jitka

Selection

Selection: preserves only tuples we are interested in

$\sigma_\varphi(E)$ or $E(\varphi)$

- $\langle R, A_R \rangle = [\![E]\!]$
- φ is a condition (**Boolean expression**) to be satisfied
  - **Connectives**: $\wedge$ (and), $\vee$ (or), $\neg$ (negation)
  - Two forms of **atomic formulae**: $a \, \Theta \, b$ or $a \, \Theta \, v$
  - $a, b \in A_R$ are **attributes**, v is a value **constant**
  - $\Theta \in \{<, \leq, =, \neq, \geq, >\}$ is a **comparison** operator

$$[\![\, \sigma_\varphi(E) \,]\!] = \langle \, \{t \,|\, t \in R, t \vDash \varphi\} \, , \, A_R \, \rangle$$

Selection: Example

Actors born in *1960* or later having a first name other than *Jitka*

$$\sigma_{\text{year} \geq 1960 \wedge \text{name} \neq \text{Jitka}}(\text{Actor})$$

$$\text{Actor}(\text{year} \geq 1960 \wedge \text{name} \neq \text{Jitka})$$

name	surname	year
Ivan	Trojan	1964
Jiří	Macháček	1966
Jitka	Schneiderová	1973
Zdeněk	Svěrák	1936
Jitka	Čvančarová	1978

name	surname	year
Ivan	Trojan	1964
Jiří	Macháček	1966

Attribute Renaming

Rename: changes names of certain attributes

$\rho_{b_1/a_1,\ldots,b_n/a_n}(E)$ or $E\langle a_1 \rightarrow b_1, \ldots, a_n \rightarrow b_n \rangle$

- $\langle R, A_R \rangle = [\![E]\!]$
- $a_1, \ldots, a_n$ are **current attributes**, each $a_i \in A_R$,
  $b_1, \ldots, b_n$ are **new attributes** (distinct)

$$[\![\, \rho_{b_1/a_1,\ldots,b_n/a_n}(E) \,]\!] = \langle\ \{t[b_1/a_1, \ldots, b_n/a_n] \mid t \in R\}\ ,\ (A_R \setminus \{a_1, \ldots, a_n\}) \cup \{b_1, \ldots, b_n\}\ \rangle$$

- $t[b_1/a_1, \ldots, b_n/a_n] =$
  $\{(a, v) \mid (a, v) \in t, a \notin \{a_1, \ldots, a_n\}\} \cup$
  $\{(b_i, v) \mid (a_i, v) \in t, i \in \{1, \ldots, n\}\}$

Attribute Renaming: Example

Actors with renamed attributes of first and last names

$$\rho_{\text{fname/name, lname/surname}}(\text{Actor})$$

$$\text{Actor}\langle \text{name} \rightarrow \text{fname}, \text{surname} \rightarrow \text{lname} \rangle$$

name	surname	year

fname	lname	year
Ivan	Trojan	1964
Jiří	Macháček	1966
Jitka	Schneiderová	1973
Zdeněk	Svěrák	1936
Jitka	Čvančarová	1978

Set Operations

Union, intersection, difference: standard set operations

$$\boxed{E_R \cup E_S} \qquad [\![\boxed{E_R \cup E_S}]\!] = \langle \boxed{R \cup S}, \boxed{A} \rangle$$

$$\boxed{E_R \cap E_S} \qquad [\![\boxed{E_R \cap E_S}]\!] = \langle \boxed{R \cap S}, \boxed{A} \rangle$$

$$\boxed{E_R \setminus E_S} \qquad [\![\boxed{E_R \setminus E_S}]\!] = \langle \boxed{R \setminus S}, \boxed{A} \rangle$$

- $\langle R, A \rangle = [\![E_R]\!]$ and $\langle S, A \rangle = [\![E_S]\!]$
- Both the relations must be **compatible**
  - I.e. they must have the same attributes

Set Operations: Difference: Example

Movies that do not have a good rating

AllMovies \ GoodMovies

title	rating
Vratné lahve	76
Samotáři	84
Medvídek	53
Štěstí	72

\

title	rating
Samotáři	84
Kolja	86

=

title	rating
Vratné lahve	76
Medvídek	53
Štěstí	72

Cartesian Product

Cartesian product (**cross join**): yields all combinations of tuples from two relations, i.e. unconditionally joins two relations

$$\boxed{E_R \times E_S}$$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$
- Both the relations must have **disjoint attributes**
  - ***Dot convention*** *based on names of relations is often used in practice, but this approach is not always applicable*
    - $R.a$ and $S.a$ for ambiguous attributes a

$$[\![\boxed{E_R \times E_S}]\!] = \langle \boxed{\{t_1 \cup t_2 \mid t_1 \in R, t_2 \in S\}} , \boxed{A_R \cup A_S} \rangle$$

- Resulting relations are **flat**
  - *I.e. our Cartesian product differs to the one in the set theory*
- Cardinality of the result: $|R|.|S|$

Cartesian Product: Example

All possible combinations of movies and actors

$\text{Movie} \times \text{Actor}$

title	rating
Vratné lahve	76
Samotáři	84
Medvídek	53

$\times$

actor
Ivan Trojan
Jiří Macháček

$=$

title	rating	actor
Vratné lahve	76	Ivan Trojan
Vratné lahve	76	Jiří Macháček
Samotáři	84	Ivan Trojan
Samotáři	84	Jiří Macháček
Medvídek	53	Ivan Trojan
Medvídek	53	Jiří Macháček

Natural Join

Natural join: joins two relations based on the pairwise equality of values of all the attributes they mutually share

$$\boxed{E_R \bowtie E_S} \text{ or } \boxed{E_R * E_S}$$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$

$$[\![\boxed{E_R \bowtie E_S}]\!] = \langle$$

$$\boxed{\{t_1 \cup t_2 \mid t_1 \in R, t_2 \in S, \forall\, a \in A_R \cap A_S : t_1(a) = t_2(a)\}},$$

$$\boxed{A_R \cup A_S}$$

$$\rangle$$

- When there are no shared attributes (i.e. $A_R \cap A_S = \emptyset$), $\bowtie$ corresponds to $\times$

Natural Join: Example

Movie characters with full actor names

$\text{Cast} \bowtie \text{Actor}$ $\text{Cast} * \text{Actor}$

title	actor
Vratné lahve	2
Vratné lahve	4
Samotáři	1
Medvídek	1
Medvídek	2

$\bowtie$

actor	name
1	Ivan Trojan
2	Jiří Macháček
3	Jitka Schneiderová

$=$

title	actor	name
Vratné lahve	2	Jiří Macháček
Samotáři	1	Ivan Trojan
Medvídek	1	Ivan Trojan
Medvídek	2	Jiří Macháček

Natural Join: Inference

$$\boxed{\boxed{E_R} \bowtie \boxed{E_S}} \equiv$$

$$\boxed{\pi_{r_1,\dots,r_m,a_1,\dots,a_n,s_1,\dots,s_o}\Big(\boxed{\sigma_{x_1=a_1\wedge\dots\wedge x_n=a_n}\Big(\boxed{\boxed{\rho_{x_1/a_1,\dots,x_n/a_n}\big(\boxed{E_R}\big)} \times \boxed{E_S}} \Big)} \Big)}$$

- $\langle R, A_R\rangle = [\![E_R]\!]$ and $\langle S, A_S\rangle = [\![E_S]\!]$
  - $a_1, \dots, a_n$ are all the attributes shared by R and S
  - $x_1, \dots, x_n$ are unused attributes, i.e. each $x_i \notin A_R$, $x_i \notin A_S$
  - $r_1, \dots, r_m$ are all the attributes from $A_R \setminus \{a_1, \dots, a_n\}$
  - $s_1, \dots, s_o$ are all the attributes from $A_S \setminus \{a_1, \dots, a_n\}$

Theta Join

Theta join (**Θ-join**): joins two relations based on a certain condition

$E_R \bowtie_\varphi E_S$ or $E_R[\varphi]E_S$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$
- **Disjoint attributes**, i.e. $A_R \cap A_S = \emptyset$
- φ is a **condition to be satisfied**
  - Works the same way as conditions in selections

$$[\![E_R \bowtie_\varphi E_S]\!] = \langle$$

$$\{ t_1 \cup t_2 \mid t_1 \in R, t_2 \in S, (t_1 \cup t_2) \vDash \varphi \},$$

$$A_R \cup A_S$$

$$\rangle$$

Theta Join

Inference

- $\boxed{\boxed{E_R} \bowtie_\varphi \boxed{E_S}} \equiv \boxed{\sigma_\varphi(\boxed{E_R} \times \boxed{E_S})}$

Theta Join: Example

Suitable combinations of movies and actors based on years

$\text{Movie} \bowtie_{\text{filmed} \geq \text{born}} \text{Actor}$

$\text{Movie}[\text{filmed} \geq \text{born}]\text{Actor}$

title	filmed
Vratné lahve	2006
Ecce homo Homolka	1970

$\bowtie_{\varphi}$

actor	born
Trojan	1964
Macháček	1966
Schneiderová	1973

$=$

title	filmed	actor	born
Vratné lahve	2006	Trojan	1964
Vratné lahve	2006	Macháček	1966
Vratné lahve	2006	Schneiderová	1973
Ecce homo Homolka	1970	Trojan	1964
Ecce homo Homolka	1970	Macháček	1966

Semijoin

Left / right (natural) **semijoin**: yields tuples from the left / right relation that can be naturally joined with the other relation

$$E_R \ltimes E_S \text{ or } E_R <\!* \; E_S \;/\; E_R \rtimes E_S \text{ or } E_R *\!> E_S$$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$

Left semijoin:

$$[\![\, E_R \ltimes E_S \,]\!] = \langle$$

$$\{ t_1 \mid t_1 \in R, \exists\, t_2 \in S : \forall\, a \in A_R \cap A_S : t_1(a) = t_2(a) \},$$

$$A_R$$

$$\rangle$$

Right semijoin: analogously

Semijoin

Inference

- $\boxed{\boxed{E_R} \ltimes \boxed{E_S}} \equiv \boxed{\pi_{r_1,\ldots,r_n}(\boxed{E_R} \bowtie \boxed{E_S})}$
  - where $r_1, \ldots, r_n$ are all attributes from the left relation
- Analogously for the right semijoin

Semijoin: Example

Movie characters who have actor details available

Cast $\ltimes$ Actor   Cast $<*$ Actor

title	actor
Vratné lahve	2
Vratné lahve	4
Samotáři	1
Medvídek	1
Medvídek	2

$\ltimes$

actor	name
1	Ivan Trojan
2	Jiří Macháček
3	Jitka Schneiderová

$=$

title	actor
Vratné lahve	2
Samotáři	1
Medvídek	1
Medvídek	2

Antijoin

Left / right antijoin: yields tuples from the left / right relation that cannot be naturally joined with the other relation

$E_R \triangleright E_S$ / $E_R \triangleleft E_S$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$

Left antijoin:

$$[\![E_R \triangleright E_S]\!] = \langle$$

$$\{ t_1 \mid t_1 \in R, \neg \exists\, t_2 \in S : \forall\, a \in A_R \cap A_S : t_1(a) = t_2(a) \} \,,$$

$$A_R$$

$$\rangle$$

Right antijoin: analogously

Antijoin

Inference

- $\boxed{\boxed{E_R} \triangleright \boxed{E_S}} \equiv \boxed{\boxed{E_R} \setminus (\boxed{E_R} \ltimes \boxed{E_S})}$
- Analogously for the right antijoin

Antijoin: Example

Movie characters that do not have actor details available

Cast ▷ Actor

title	actor
Vratné lahve	2
Vratné lahve	4
Samotáři	1
Medvídek	1
Medvídek	2

▷

actor	name
1	Ivan Trojan
2	Jiří Macháček
3	Jitka Schneiderová

=

title	actor
Vratné lahve	4

Theta Semijoin

Left / right theta semijoin (Θ-semijoin): yields tuples from a given relation that can be joined using a certain condition

$\boxed{E_R \ltimes_\varphi E_S}$ or $\boxed{E_R \ \langle\varphi]\ E_S}$ / $\boxed{E_R \rtimes_\varphi E_S}$ or $\boxed{E_R \ [\varphi\rangle\ E_S}$

- $\langle R, A_R\rangle = [\![E_R]\!]$ and $\langle S, A_S\rangle = [\![E_S]\!]$
- Disjoint attributes, condition φ to be satisfied

Left Θ-semijoin:

$$[\![\ \boxed{E_R \ltimes_\varphi E_S}\ ]\!] = \langle$$

$$\boxed{\{t_1 \mid t_1 \in R, \exists\, t_2 \in S : (t_1 \cup t_2) \vDash \varphi\}}\ ,\ \boxed{A_R}$$

$$\rangle$$

Right Θ-semijoin: analogously

Theta Semijoin

Inference

- $\boxed{\boxed{E_R} \ltimes_\varphi \boxed{E_S}} \equiv \boxed{\pi_{r_1,\ldots,r_n}(\boxed{E_R} \bowtie_\varphi \boxed{E_S})}$
  - where $r_1, \ldots, r_n$ are all attributes from the left relation
- Analogously for the right Θ-semijoin

Division

Division: returns restrictions of tuples from the first relation such that all combinations of these restricted tuples with tuples from the second relation are present in the first relation

$$\boxed{E_R \div E_S}$$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$
- **Assumption on attributes**: $A_S \subset A_R$ (proper subset)

$$[\![\boxed{E_R \div E_S}]\!] = \langle \boxed{\{t \mid \forall\, t_2 \in S : (t \cup t_2) \in R\}} , \boxed{A_R \setminus A_S} \rangle$$

- Division allows for the simulation of the **universal quantifier**

Division: Example: 1

Movies in which all the actors played

Cast ÷ Actor

title	actor
Vratné lahve	Jiří Macháček
Vratné lahve	Zdeněk Svěrák
Samotáři	Ivan Trojan
Medvídek	Ivan Trojan
Medvídek	Jiří Macháček

÷

actor
Ivan Trojan
Jiří Macháček

=

title
Medvídek

Division: Example: 2

Movies in which all the actors played

Cast ÷ Actor

title	name	surname
Vratné lahve	Jiří	Macháček
Vratné lahve	Zdeněk	Svěrák
Samotáři	Ivan	Trojan
Medvídek	Ivan	Trojan
Medvídek	Jiří	Macháček

÷

name	surname
Ivan	Trojan
Jiří	Macháček

=

title
Medvídek

Division: Inference

$$\boxed{\boxed{E_R} \div \boxed{E_S}} \equiv$$

$$\boxed{\boxed{\pi_{r_1,\ldots,r_m}(\boxed{E_R})} \setminus \boxed{\pi_{r_1,\ldots,r_m}\Big(\boxed{\Big(\boxed{\boxed{\pi_{r_1,\ldots,r_m}(\boxed{E_R})} \times \boxed{E_S}}\Big) \setminus \boxed{E_R}}\Big)}}$$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$
- $A_R = \{r_1, \ldots, r_m\} \cup \{s_1, \ldots, s_n\}$ and $A_S = \{s_1, \ldots, s_n\}$
  - I.e. $s_1, \ldots, s_n$ are all the attributes shared by R and S, $r_1, \ldots, r_m$ are all the remaining attributes in R

Join Operations

Inner joins (and antijoins)

- **Cartesian product**: $E_R \times E_S$
- **Natural join**: $E_R \bowtie E_S$
- **Theta join**: $E_R \bowtie_\varphi E_S$
- Left / right **semijoin**: $E_R \ltimes E_S$, $E_R \rtimes E_S$
- Left / right **antijoin**: $E_R \rhd E_S$, $E_R \lhd E_S$
- Left / right **theta semijoin**: $E_R \ltimes_\varphi E_S$, $E_R \rtimes_\varphi E_S$
- Left / right **theta antijoin**: $E_R \rhd_\varphi E_S$, $E_R \lhd_\varphi E_S$

Join Operations

Outer joins

- Left / right / full **outer join**:
  $E_R ⟕ E_S$, $E_R ⟖ E_S$, $E_R ⟗ E_S$
- Left / right / full **outer theta join**:
  $E_R ⟕_\varphi E_S$, $E_R ⟖_\varphi E_S$, $E_R ⟗_\varphi E_S$

Extended relational model with **null** values is required

Outer Join

Left / right / full outer join: natural join of two relations extended by tuples of the first / second / both relations that cannot be joined

$E_R ⟕ E_S$ / $E_R ⟖ E_S$ / $E_R ⟗ E_S$

$E_R *_{\mathsf{L}} E_S$ / $E_R *_{\mathsf{R}} E_S$ / $E_R *_{\mathsf{F}} E_S$

- $\langle R, A_R \rangle = [\![E_R]\!]$ and $\langle S, A_S \rangle = [\![E_S]\!]$
- $A_R = \{r_1, \ldots, r_m\}$, $A_S = \{s_1, \ldots, s_n\}$

$$E_R ⟕ E_S \equiv (E_R \bowtie E_S) \cup ((E_R \rhd E_S) \times \{(\textsf{null}, \ldots, \textsf{null})\}_{s_1, \ldots, s_n})$$

$$E_R ⟖ E_S \equiv (E_R \bowtie E_S) \cup (\{(\textsf{null}, \ldots, \textsf{null})\}_{r_1, \ldots, r_m} \times (E_R \lhd E_S))$$

$$E_R ⟗ E_S \equiv (E_R ⟕ E_S) \cup (E_R ⟖ E_S)$$

Outer Join: Example

Movie characters with full actor names if possible

Cast ⟕ Actor

Cast $*_L$ Actor

title	actor
Vratné lahve	2
Vratné lahve	4
Samotáři	1
Medvídek	1

⟕

actor	name
1	Ivan Trojan
2	Jiří Macháček
3	Jitka Schneiderová

=

title	actor	name
Vratné lahve	2	Jiří Macháček
Vratné lahve	4	null
Samotáři	1	Ivan Trojan
Medvídek	1	Ivan Trojan

Observations

Relational algebra

- **Declarative** query language
  - Query expressions describe **what data to retrieve**, not (necessarily) how such data should be retrieved
- Both *inputs* and *outputs* of queries are **relations**
- Only values actually present in the database can be returned
  - I.e. **derived data cannot be returned** (such as various calculations, statistics, aggregations, …)

Observations

Query evaluation

- Construction of a **syntactic tree** (query expression parsing)
  - Based on an inductive structure of a given query expression
  - I.e. based on parentheses (often omitted), operation priorities, associativity conventions, …
- Nodes
  - **Leaf nodes** correspond to individual input relations
  - **Inner nodes** correspond to individual operations
- Evaluation
  - Node can be evaluated when all its child nodes are evaluated, i.e. when all operands of a given operation are available
  - **Root node** represents the result of the entire query

Observations

Equivalent expressions

- Query expressions that **define the same query** (regardless the input relations)
- Various causes
  - **Inference of extended operations** using the basic ones
  - **Commutativity, distributivity or associativity** of (some) operations
  - …
- Examples
  - Commutativity of selection: $(E(\varphi_1))(\varphi_2) \equiv (E(\varphi_2))(\varphi_1)$
  - Selection cascade: $(E(\varphi_1))(\varphi_2) \equiv E(\varphi_1 \wedge \varphi_2)$
  - …

Observations

Basic operations

- **Not all the introduced operations are actually necessary** in order to form expressions of all the possible queries
- The minimal set of required operations:
  - **Projection**: $\pi_{a_1,\ldots,a_n}(E)$
  - **Selection**: $\sigma_{\varphi}(E)$
  - **Attribute renaming**: $\rho_{b_1/a_1,\ldots,b_n/a_n}(E)$
  - **Union**: $E_R \cup E_S$
  - **Difference**: $E_R \setminus E_S$
  - **Cartesian product**: $E_R \times E_S$

Extended operations

- Intersection, division, all types of joins except the Cartesian product, …

Observations

Relational completeness

- Query language that is able to express all queries of RA is relational complete
  - SQL is relational complete

Conclusion

Relational algebra

- **Declarative query language** for the **relational model**
- Operations
  - Basic: **projection**, **selection**, **attribute renaming**, **union**, **difference**, **Cartesian product**
  - Extended: intersection, natural join, theta join, semijoin, antijoin, division, outer join
  - …
- **Relational completeness**
- Motivation
  - Evaluation of SQL queries