3.2 Relational Algebra

Query Languages

- A query language (QL) is a language that allows users to manipulate and retrieve data from a database.
- The relational model supports simple, powerful QLs (having strong formal foundation based on logics, allow for much optimization)
- Query Language != Programming Language
  - QLs are not expected to be Turing-complete, not intended to be used for complex applications/computations
  - QLs support easy access to large data sets
- Categories of QLs: procedural versus declarative
- Two (mathematical) query languages form the basis for "real" languages (e.g., SQL) and for implementation
  - *Relational Algebra:* procedural, very useful for representing query execution plans, and query optimization techniques.
  - *Relational Calculus:* declarative, logic based language
- Understanding algebra (and calculus) is the key to understanding SQL, query processing and optimization.

Relational Algebra

- Procedural language
- Queries in relational algebra are applied to relation instances, result of a query is again a relation instance
- Six basic operators in relational algebra:

select	σ	selects a subset of tuples from reln
project	π	deletes unwanted columns from reln
Cartesian Product	$\times$	allows to combine two relations
Set-difference	$-$	tuples in reln. 1, but not in reln. 2
Union	$\cup$	tuples in reln 1 plus tuples in reln 2
Rename	ρ	renames attribute(s) and relation

- The operators take one or two relations as input and give a new relation as a result (relational algebra is "closed").

Select Operation

- Notation: $\sigma_P(r)$

  Defined as

  $$\sigma_P(r) := \{t \mid t \in r \text{ and } P(t)\}$$

  where
  - r is a relation (name),
  - P is a formula in propositional calculus, composed of conditions of the form

    $$\langle\text{attribute}\rangle = \langle\text{attribute}\rangle \text{ or } \langle\text{constant}\rangle$$

    Instead of "=" any other comparison predicate is allowed ($\neq, <, >$ etc).
    Conditions can be composed through $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

- Example: given the relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \wedge D>5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Project Operation

- Notation: $\pi_{A_1,A_2,\ldots,A_k}(r)$
  where $A_1, \ldots, A_k$ are attribute names and
  r is a relation (name).
- The result of the projection operation is defined as the relation that has k columns obtained by erasing all columns from r that are not listed.
- Duplicate rows are removed from result because relations are sets.
- Example: given the relations r

r

A	B	C
α	10	2
α	20	2
β	30	2
β	40	4

$\pi_{A,C}(r)$

A	C
α	2
β	2
β	4

Cartesian Product

- Notation: $r \times s$ where both r and s are relations

  Defined as $r \times s := \{tq \mid t \in r \textbf{ and } q \in s\}$

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint, i.e., $R \cap S = \emptyset$.

  If attributes of $r(R)$ and $s(S)$ are not disjoint, then the rename operation must be applied first.

- Example: relations r, s:

r

A	B
α	1
β	2

s

C	D	E
α	10	$+$
β	10	$+$
β	20	$-$
γ	10	$-$

$r \times s$

A	B	C	D	E
α	1	α	10	$+$
α	1	β	10	$+$
α	1	β	20	$-$
α	1	γ	10	$-$
β	2	α	10	$+$
β	2	β	10	$+$
β	2	β	20	$-$
β	2	γ	10	$-$

Union Operator

- Notation: $r \cup s$ where both r and s are relations

  Defined as $r \cup s := \{t \mid t \in r \textbf{ or } t \in s\}$

- For $r \cup s$ to be applicable,

  1. r, s must have the same number of attributes
  2. Attribute domains must be compatible (e.g., 3rd column of r has a data type matching the data type of the 3rd column of s)

- Example: given the relations r and s

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

$r \cup s$

A	B
α	1
α	2
β	1
β	3

Set Difference Operator

- Notation: $r - s$ where both r and s are relations

  Defined as $r - s := \{t \mid t \in r \textbf{ and } t \notin s\}$

- For $r - s$ to be applicable,
  1. r and s must have the same arity
  2. Attribute domains must be compatible

- Example: given the relations r and s

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

$r - s$

A	B
α	1
β	1

Rename Operation

- Allows to name and therefore to refer to the result of relational algebra expression.
- Allows to refer to a relation by more than one name (e.g., if the same relation is used twice in a relational algebra expression).
- Example:

  $$\rho_x(E)$$

  returns the relational algebra expression E under the name x

  If a relational algebra expression E (which is a relation) has the arity k, then

  $$\rho_{x(A_1,A_2,\ldots,A_k)}(E)$$

  returns the expression E under the name x, and with the attribute names $A_1, A_2, \ldots, A_k$.

Composition of Operations

- It is possible to build relational algebra expressions using multiple operators similar to the use of arithmetic operators (nesting of operators)

- Example: $\sigma_{A=C}(r \times s)$

$r \times s$

A	B	C	D	E
α	1	α	10	$+$
α	1	β	10	$+$
α	1	β	20	$-$
α	1	γ	10	$-$
β	2	α	10	$+$
β	2	β	10	$+$
β	2	β	20	$-$
β	2	γ	10	$-$

$\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	$+$
β	2	β	10	$+$
β	2	β	20	$-$