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)
- 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 a relation instance
- Six basic operators in relational algebra:

select	σ	selects a subset of tuples from a relation
project	π	deletes unwanted columns from a relation
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

- 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 $\sigma_P(r) := \{t \mid t \in r$ and $P(t)\}$ where r is a relation (name), P is a formula in propositional calculus, dealing with conditions of the form

  <attribute> = <attribute> or <constant>

  Instead of "=" any other comparison predicate is allowed ($\neq, <, >$ etc).
  Conditions can be connected 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

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 valid (applicable),
  1. r, s must have the same arity (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

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 since 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 renaming operation must be used.
- Example: relations r, s

A	B
α	1
β	2

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	$-$

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 renamed to $A_1, A_2, \ldots, A_k$.

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 valid (applicable),
  1. r, 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

Composition of Operations

- It is possible to build relational algebra expressions using multiple operat 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	$-$

Set-Intersection

- Notation: $r \cap s$
  Defined as $r \cap s := \{t \mid t \in r \textbf{ and } t \in s\}$
- For $r \cap s$ to be valid (applicable),
  1. r, s must have the same arity
  2. Attribute domains must be compatible
- Derivation: $r \cap s = r - (r - s)$
- Example: given the relations r and s:

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

$r \cap s$

A	B
α	2

Natural Join

- Notation: $r \bowtie s$
- Let r, s be relations on schemas R and S, respectively. The result is a relation on schema $R \cup S$. The result tuples are obtained by considering each pair of tuples $t_r \in r$ and $t_s \in s$.
- If t_r and t_s have the same value for each of the attributes in $R \cap S$ ("same name attributes"), a tuple t is added to the result such that
  - t has the same value as t_r on r
  - t has the same value as t_s on s
- Example: Given the relations $R(A, B, C, D)$ and $S(B, D, E)$
  - Join can be applied because $R \cap S \neq \emptyset$
  - the result schema is (A, B, C, D, E)
  - and the result of $r \bowtie s$ is defined as

$$\pi_{r.A,r.B,r.C,r.D,s.E}(\sigma_{r.B=s.B \wedge r.D=s.D}(r \times s))$$

- Example: given the relations r and s:

r

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

s

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	τ

$r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Condition Join

- Notation: $r \bowtie_C s$
  C is a condition on attributes in $R \cup S$, result schema is the same as that of Cartesian Product. If $R \cap S \neq \emptyset$ and condition C refers to these attributes, some of these attributes must be renamed.
  Sometimes also called *Theta Join* ($r \bowtie_\theta s$).
- Derivation: $r \bowtie_C s = \sigma_C(r \times s)$
- Note that – unlike a selection condition P in σ_P – C is a condition on attributes from both r and s
- Example: given two relations r, s:

r

A	B	C
1	2	3
4	5	6
7	8	9

s

D	E
3	1
6	2

$r \bowtie_{\text{B<D}} s$

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

If C involves only the comparison operator "$=$", the condition join is also called *Equi-Join*.

- Example 2:

r

A	B	C
4	5	6
7	8	9

s

C	D
6	8
10	12

$r \bowtie_{\text{C=SC}} (\rho_{\text{S(SC,D)}}(s))$

A	B	C	SC	D
4	5	6	6	8

Division

- Notation: $r \div s$
- Precondition: attributes in S must be a subset of attributes in R, i.e., $S \subseteq R$. Let r, s be relations on schemas R and S, respectively, where
  - $R(A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S(B_1, \ldots, B_n)$

  The result of $r \div s$ is a relation on schema $R - S = (A_1, \ldots, A_m)$
- Suited for queries that include the phrase "for all".

  The result of the division operator consists of the set of tuples from r defined over the attributes $R - S$ that match the combination of **every** tuple in s.

  $r \div s := \{t \mid t \in \pi_{R-S}(r) \wedge \forall u \in s : tu \in r\}$
- Example: given the relations r, s:

r

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

s

D	E
a	1
b	1

$r \div s$

A	B	C
α	a	γ
γ	a	γ