Definition 1 (**Query Language**). *A query language is a language in which a user requests information from the database.*

These languages are usually on a level higher than that of a standard programming language. Query languages can be categorized as – (i) procedural, (ii) nonprocedural.

1. **Procedural Language:** In a procedural language, the user instructs the system **to perform a sequence of operations** on the database to compute the desired result. Example: Relational algebra.
2. **Non-Procedural Language:** In a nonprocedural language, the user describes the **desired information without giving a specific procedure** for obtaining that information. Example: Tuple relational calculus and Domain relational calculus.

Definition 2 (**Relational Algebra**). *It consists of a set of operations that take one or two relations as input and produce a new relation as their result*

- **Fundamental Operations:**
  - Unary Operation: Select, Project, Rename
  - Binary Operation: Union, Set-Difference, Cartesian-Product
- **Other Operations:** Set-Intersection, Natural-Join, Assignment

Definition 3 (**Degree/Arity of Relation**). *The degree (or arity) of the relation is the number of attributes in it.*

Sequence of relational algebra operations forms a relational algebra expression.

1 Fundamental Operations

1.1 Unary Operations

1. **Select:** This operation selects a subset of the tuples from a relation that satisfies a selection condition (or predicate). Mathematically, $\sigma_{\langle \texttt{condition} \rangle}(r)$.

   - The degree of the relation resulting from a **Select** operation is the same as the degree of r.
   - Number of tuples in the resulting relation $\leq$ Number of tuples in original relation r.

   $$|\sigma_{\langle \texttt{condition} \rangle}(r)| \leq |r|$$

   - **Select** operation is commutative.

   $$\sigma_{\langle \texttt{condition-1} \rangle}\left(\sigma_{\langle \texttt{condition-2} \rangle}(r)\right) = \sigma_{\langle \texttt{condition-2} \rangle}\left(\sigma_{\langle \texttt{condition-1} \rangle}(r)\right)$$

   - A sequence of **Select** operations can be combined into a single **Select** operation with a conjunctive **(AND)** condition.

   $$\sigma_{\langle \texttt{condition-1} \rangle}\left(\sigma_{\langle \texttt{condition-2} \rangle}\left(\ldots\left(\sigma_{\langle \texttt{condition-n} \rangle}(r)\right)\ldots\right)\right) = \sigma_{\langle \texttt{condition-1} \rangle \wedge \langle \texttt{condition-2} \rangle \wedge \ldots \wedge \langle \texttt{condition-n} \rangle}(r)$$

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

(a) Relation r

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

(b) $\sigma_{A=B}(r)$

A	***B***	***C***	***D***
α	α	1	7
β	β	23	10

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

Figure 1: Select operation.

2. **Project:** This operation selects certain columns from the table and discards the other columns. Mathematically, $\Pi_{\langle \texttt{attribute-list} \rangle}(r)$.

- The result of the **Project** operation has only the attributes specified in ⟨`attribute-list`⟩ in the same order as they appear in the list.
- The degree of the relation resulting from a **Project** operation is equal to the number of attributes in ⟨`attribute-list`⟩.
- Duplicate tuples are removed from the resulting relation.
  - This scenario can occur when the ⟨`attribute-list`⟩ includes only nonkey attributes of r.
- Number of tuples in the resulting relation $\leq$ Number of tuples in original relation r.

$$|\Pi_{\langle \texttt{attribute-list} \rangle}(r)| \leq |r|$$

- If the ⟨`attribute-list`⟩ list is a superkey of r, the resulting relation has the same number of tuples as r.
- If ⟨`attribute-list-2`⟩ contains the attributes in ⟨`attribute-list-1`⟩, then

$$\Pi_{\langle \texttt{attribute-list-1} \rangle}\left(\Pi_{\langle \texttt{attribute-list-2} \rangle}(r)\right) = \Pi_{\langle \texttt{attribute-list-1} \rangle}(r)$$

- It is also noteworthy that commutativity does not hold on **Project**.

A	***B***	***C***
α	10	1
α	20	1
β	30	1
β	40	2

(a) Relation r

A
α
β

(b) $\Pi_A(r)$

A	***C***
α	1
β	1
β	2

(c) $\Pi_{A,C}(r)$

Figure 2: Project operation.

3. **Rename:** This operation rename either the relation name or the attribute names, or both. Let $r\,(A_1, A_2, \ldots, A_n)$ be a relation. The general **Rename** operation when applied to a relation is denoted by any of the following three forms –

   - Rename relation – $\rho_x(r)$.
   - Rename attributes – $\rho_{(B_1, B_2, \ldots, B_n)}(r)$.
   - Rename relation and attributes – $\rho_{x(B_1, B_2, \ldots, B_n)}(r)$.

   - This operation can also be applied to relational-algebra expression.
   - A relation r by itself is considered a (trivial) relational-algebra expression.
   - Unlike relations in the database, the results of relational-algebra expressions do not have a name that we can use to refer to them. It is useful in some cases to give them names and for this we have rename operator.

1.2 Binary Operations

Two relations $r\,(A_1, A_2, \ldots, A_n)$ and $s\,(B_1, B_2, \ldots, B_n)$ are said to be union compatible (or type compatible) if –

- r and s have the same degree.
- The domain of the i^{th} attribute of r and the i^{th} attribute of s must be the same.

$$\textsc{Domain}(A_i) = \textsc{Domain}(B_i) \quad \forall i = \{1, 2, \ldots, n\}$$

1. **Union:** The result of this operation is a relation that includes all tuples that are either in r or in s or in both r and s. Mathematically, $r \cup s$.

   - Duplicate tuples are eliminated.
   - Union operation is commutative.

$$r \cup s = s \cup r$$

A	B
α	1
α	2
β	1

(a) Relation r

A	B
α	2
β	3

(b) Relation s

A	B
α	1
α	2
β	1
β	3

(c) $r \cup s$

Figure 3: Union operation.

- Union operation is associative.

$$r \cup (s \cup t) = (r \cup s) \cup t$$

2. **Set-Difference:** The result of this operation is a relation that includes all tuples that are in r but not in s. Mathematically, $r - s$.

   - This operation is also known as Minus.
   - Set-Difference operation is not commutative.

$$r - s \neq s - r$$

A	B
α	1
α	2
β	1

(a) Relation r

A	B
α	2
β	3

(b) Relation s

A	B
α	1
β	1

(c) $r - s$

Figure 4: Set-Difference operation.

3. **Cartesian-Product:** The result of this operation has one tuple for each combination of tuples – one from r and one from s. Mathematically, $r \times s$.

   - This operation is also known as Cross-Product or Cross-Join.
   - The relations on which it is applied do not have to be union compatible.
   - If r has n_r tuples and s has n_s tuples, then $r \times s$ will have $n_r \times n_s$ tuples.

$$|r \times s| = |r| \times |s|$$

   - The result of $r(A_1, A_2, \ldots, A_n) \times s(B_1, B_2, \ldots, B_m)$ is a relation $t(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ where first all the attributes of r appears then the attributes of s appears.
   - Same attribute name may appear in the schemas of both the relations r and s, so there should be a mechanism to distinguish between these attributes. We do so here by attaching to an attribute the name of the relation from which the attribute originally came. For example,

$$r(A_1, A_2, \ldots, A_n) \times s(B_1, B_2, \ldots, B_m) = t(r.A_1, r.A_2, \ldots, r.A_n, s.B_1, s.B_2, \ldots, s.B_m)$$

   - For those attributes that appear in only one of the two schemas, the relation-name prefix can be omitted.

2 Other Operations

1. **Set-Intersection:** The result of this operation, denoted by $r \cap s$, is a relation that includes all tuples that are both in r and s. Duplicate tuples are eliminated.

   - Set-Intersection operation is commutative, *i.e.*, $r \cap s = s \cap r$.
   - Set-Intersection operation is associative, *i.e.*, $r \cap (s \cap t) = (r \cap s) \cap t$.

A	B
α	1
β	2

(a) Relation r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

(b) Relation s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

(c) $r \times s$

Figure 5: Cartesian-Product operation.

A	B
α	1
β	2

(a) Relation r

B	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

(b) Relation s

A	$r.B$	$s.B$	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

(c) $r \times s$

Figure 6: Cartesian-Product operation.

- Set-Intersection in terms of Set-Difference: $r \cap s = r - (r - s)$
- Set-Intersection in terms of Union and Set-Difference:

$$\begin{aligned} r \cap s &= ((r \cup s) - (r - s)) - (s - r) \\ &= ((r \cup s) - (s - r)) - (r - s) \\ &= (r \cup s) - ((r - s) \cup (s - r)) \end{aligned}$$

A	B
α	1
α	2
β	1

(a) Relation r

A	B
α	2
β	3

(b) Relation s

A	B
α	2

(c) $r \cap s$

Figure 7: Set-Intersection operation.

2. **Natural-Join:** The natural join allows us to combine certain selections and a Cartesian product into one operation. It is denoted by the join symbol $\bowtie$.

   - This operations performs three things.
     - Performs Cartesian product on its two relations.
     - Performs a selection forcing equality on those attributes that appear in both relation schemas.
     - Removes duplicate attributes.
   - The relations on which it is applied do not have to be union compatible.
   - Consider two relations $r(R)$ and $s(S)$.
     - $R = \{A_1, A_2, \ldots, A_n\}$: List of attributes for first relation
     - $S = \{B_1, B_2, \ldots, B_m\}$: List of attributes for second relation

The natural join of r and s, denoted by $r \bowtie s$, is a relation on schema $R \cup S$ formally defined as follows.

$$r \bowtie s = \Pi_{R \cup S} \left(\sigma_{r.C_1 = s.C_1 \wedge r.C_2 = s.C_2 \wedge \ldots \wedge r.C_k = s.C_k} \left(r \times s \right) \right)$$

where $R \cap S = \{C_1, C_2, \ldots, C_k\}$

- If $R \cap S = \emptyset$ then $r \bowtie s = r \times s$.
- Natural-Join is commutative.

$$r \bowtie s = s \bowtie r$$

- Natural-Join is associative.

$$(r \bowtie s) \bowtie t = r \bowtie (s \bowtie t)$$

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

(a) Relation r

B	***D***	***E***
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

(b) Relation 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

(c) $r \bowtie s$

Figure 8: Natural-Join operation.

3. **Assignment:** The assignment operation, denoted by $\leftarrow$, works like assignment in a programming language. $r \bowtie s$ can be written as:

$$\begin{aligned} \texttt{temp1} &\leftarrow r \times s \\ \texttt{temp2} &\leftarrow \sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_k = s.A_k} \left(\texttt{temp1} \right) \\ \texttt{result} &\leftarrow \Pi_{R \cup S} \left(\texttt{temp2} \right) \end{aligned}$$

- The evaluation of an assignment does not result in any relation being displayed to the user.
- For relational-algebra queries, assignment must always be made to a temporary relation variable.
- Assignments to permanent relations constitute a database modification.

Given two relations `r` and `s`. The number of tuples in `r` and `s` is N_r and N_s respectively. In relational algebra, what is the minimum / maximum number of tuples in

- $\texttt{r} \cup \texttt{s}$
- $\texttt{r} \cap \texttt{s}$
- $\texttt{r} - \texttt{s}$
- $\texttt{s} - \texttt{r}$
- $\texttt{r} \times \texttt{s}$
  - Assume there is at-least one common attribute in both the relations.
  - Assume there is no common attribute in both the relations.
- $\texttt{r} \bowtie \texttt{s}$
  - Assume there is at-least one common attribute in both the relations.
  - Assume there is no common attribute in both the relations.

Given two relations `r` and `s`. The arity of `r` and `s` is C_r and C_s respectively. In relational algebra, what is the arity of

- $\texttt{r} \cup \texttt{s}$
- $\texttt{r} \cap \texttt{s}$
- $\texttt{r} - \texttt{s}$
- $\texttt{s} - \texttt{r}$
- $\texttt{r} \times \texttt{s}$
  - Assume there is C_{rs} number of common attribute in both the relations.
  - Assume there is no common attribute in both the relations.
- $\texttt{r} \bowtie \texttt{s}$
  - Assume there is C_{rs} number of common attribute in both the relations.
  - Assume there is no common attribute in both the relations.