**1. Procedural vs. Non-Procedural Languages**

When querying a database, we can use different types of languages based on how much control we want over the retrieval process.

**Procedural Query Languages:**

* The user specifies *what* data is needed **and** *how* to retrieve it.
* It requires writing step-by-step instructions to fetch data.
* Example: **Relational Algebra**, where we explicitly define operations like selection, projection, and joins to manipulate relations.

**Non-Procedural (Declarative) Query Languages:**

* The user only specifies *what* data is needed, not *how* to retrieve it.
* The database management system (DBMS) determines the best way to fetch the data.
* Example: **SQL (Structured Query Language)** – Users write queries using SELECT, WHERE, etc., without worrying about execution details.

**2. "Pure" Query Languages**

These are formal, mathematical query languages that provide the foundation for database query processing.

**Relational Algebra (Procedural)**

* A formal language with a set of operations (such as selection, projection, join, and union) to manipulate relations (tables).
* Example:
  + **Selection (σ)**: σ(condition)(Relation) → Filters rows based on a condition.
  + **Projection (π)**: π(attributes)(Relation) → Selects specific columns.
  + **Join (⨝)**: Combines two relations based on a common attribute.

**Tuple Relational Calculus (Non-Procedural)**

* A declarative language where queries are expressed as logical formulas.
* Instead of specifying operations, it describes *what* conditions the result must satisfy.
* Uses tuple variables.
* Example query (finding students who have scored more than 90 marks):

{ t | t ∈ Students ∧ t.marks > 90 }

This means: "Return all tuples t from the Students relation where t.marks is greater than 90."

**Domain Relational Calculus (Non-Procedural)**

* Similar to Tuple Relational Calculus, but queries use domain variables (representing column values instead of rows).
* Example:

{<s\_name, marks> | ∃ s (s ∈ Students ∧ s.name = s\_name ∧ s.marks > 90)}

This retrieves student names and marks where marks are greater than 90.