Chapter 2: Introduction to the Relational Model

Contents

1 Structure of Relational Databases

1) Consists of **tables**. Each table is assigned a unique name.
2) Table represents a *relationship* among a set of values.
3) Each row of a table is called a **tuple**.
4) Each column of a table is called an **attribute**.
5) A specific instance of a relation is called a **relation instance**.
6) The set of values each attribute in a relation can take is called the **domain** of that relation.
7) A domain is **atomic** if the elemebts of the domain are considered to be indivisible units.
8) Each domain contains a special element called the **null value**, which signifies that the value is unknown/does not exist.
9) Represented as $r(A_1, A_2, \ldots, A_n)$, where
   a) r is the name of the relation.
   b) A_i are the names of the attributes.

2 Database Schema

1) **Database schema:** The logical design of the database.
2) **Database instance:** Snapshot of data in the database at any point of time.
3) The notion of *relation* corresponds to that of a variable in programming, but that of a *relation schema* corresponds to the notion of type definition.

3 Keys

1) **Superkey:** A set of one or more attributes that uniquely identify a tuple in a relation. Mathematically, if $K \subseteq R$ is a superkey of relation r, where R is the set of attributes of r, then $t1 = t2 \iff t1.K = t2.K$.
2) **Candidate Key:** A minimal superkey (i.e., no subset of this key is also a superkey).
3) **Primary Key:** A candidate kay that is chosen to identify tuples within a relation. Also called *primary key constraints*.
4) Attributes of a primary key are *underlined* while representing a relation.
5) A primary key should be chosen such that its attributes rarely change.
6) **Foreign-key constraint:** It is a constraint from attributes A of relation r_1 to the primary key B or r_2, stating that on any database instance, the value of A for each tuple in r_1 must also be the value of B of some tuple in r_2.
   a) A is called the **foreign key** from r_1 referencing r_2.
   b) r_1 is called the **referencing relation** of the foreign- key constraint.
   c) r_2 is called the **referenced relation** of the foreign- key constraint.
7) **Referential-integrity constraint:** Values appearing in the specified attributes of the referencing relation must also appear in the referenced relation. It is a more general case of the foreign-key constraint.

4 Schema Diagrams

It is a means of representing a database schema, with the constraints.

1) Each relation is represented by a box, with its name at the top and attributes below.
2) Primary-key attributes are underlined in the relation.
3) Foreign-key constraints are represented as arrows from the referenced relation to the referencing relation.
4) A two-headed arrow represents a referential-integrity constraint that is not a foreign-key constraint.

5 Relational Query Language

Query language: A language in which a user requests information from the database. Higher-level than standard programming languages. Classified as:

1) **Imperative:** User instrcuts the system to perform a sequence of operations to compute the required result. Have notion of state variables.
2) **Functional:** Computation is expressed as the evaluation of functions. They do not update the program state. For example, *relational algebra*.
3) **Declarative:** User describes the required information needed without specifying how to compute it. The database system must figure out how to find the required information. For example, *tuple relational calculus* and *domain relational calculus*.

6 Relational Algebra

1) Consists of a set of operations that take one (for *unary* operations) or two (for *binary* operations) relations and outputs a relation.
2) Here, we consider duplicate tuples to be eliminated, though they are allowed in databases in practice.

6.1 The Select Operation

1) Select tuples that satisfy a given predicate.
2) Syntax: $\sigma_P(r)$, where
   a) σ denotes the select operation
   b) P is a logical predicate or a combination of predicates.
   c) r is the name of the relation.

6.2 The Project Operation

1) Returns the argument realtion with certain attricutes left out.
2) **Any duplicate rows in the resulting relation is eliminated.**
3) Syntax: $\Pi_{A_1,A_2,A_3,\ldots}(r)$, where
   a) Π denotes the project operation
   b) A_i are the attrubutes to be included in the result relation
   c) r is the argument relation.
   d) *Note:* A more generalized version allows the use of expressions as well as attributes in the list of attributes to project.

6.3 The Cartesian Product Operation

1) Used to combine information from two relations.
2) Cartesian product on $r_1(R_1)$ and $r_2(R_2)$ produces a relation $r(R)$ where R is the concatenation of R_1 and R_2 and $(t_1, t_2) \in r \iff t_1 \in r_1, t_2 \in r_2$. Here, (t_1, t_2) denotes the concatenation of tuples in r_1 and r_2 respectively, in that order.
3) Syntax: $r_1 \times r_2$, where
   a) $\times$ denotes the cartesian product operation.
   b) r_1 and r_2 are the relations on which the cartesian product needs to be formed.
4) If the cardinalities are n_1 and n_2, then the cardinality of the cartesian product is $n_1 n_2$.
5) To allow for repeating attribute names, the attribute name is prepended by the relation name using the dot operator, such as $r.A$.

6.4 The Join Operation

1) Allows to choose a subset of the cartesian product of two relations based on a predicate.
2) Used to find relevant information from two or more relations.
3) Combines cartesian product and selection in one single operation.
4) Syntax: $r \bowtie_\theta s = \sigma_\theta(r \times s)$, where
   a) $\bowtie$ denotes the join operation
   b) θ is the predicate on which to join the two relations.
   c) r and s are the relations on which the join operation is to be carried out.

6.5 Set Operations

6.5.1 Union:

1) It is the relation where each tuple belongs to either of the two relations.
2) Works only with **compatible** relations, i.e., relations where
   a) The **arity** or the number of attributes is same.
   b) The types associated with the i^{th} attribute is same for each i.
3) Syntax: $r \cup s$, where
   a) $\cup$ denotes the union operation.
   b) r and s are the compatible input relations.

6.5.2 Intersection:

1) Finds tuples that are present in both input relations.
2) Works only with complatible relations.
3) Syntax: $r \cap s$, where
   a) $\cap$ denotes the intersection operation.
   b) r and s are the compatible input relations.

6.5.3 Set-difference:

1) Find tuples that are in one relation and not in the other.
2) Works only with compatible relations.
3) Syntax: $r - s$, where
   a) $-$ denotes the set-difference operation.
   b) r and s are the two compatible input relations.

6.6 The Assignment Operation

1) Allows assigning a relation-algebra expression to temporary relation variables for future use.
2) Syntax: $v \leftarrow r$, where
   a) $\leftarrow$ denotes the assignment operation.
   b) r is the input relation.
   c) v is the variable r is assigned to.
3) Allows a query to be written as a sequential program for convenience.

6.7 The Rename Operation

1) Used to give resulting relations names that can be used for reference.
2) Syntax: $\rho_x(E)$, where
   a) ρ denotes the rename operation.
   b) E is the input relational-algebra expression.
   c) x is the new name of r.
3) Another syntax: $\rho_{x(A_1,A_2,\ldots,A_n)}(E)$, where
   a) ρ denotes the rename operation.
   b) E is the input relation-algebra expression.
   c) x is the new name of the relation.
   d) $A_i, 1 \leq i \leq n$ are the new names for the attributes of E
   e) n is the arity of E.
4) Not strictly required since one can use positional notation to refer to attributes. It is only a matter of convenience.

6.8 Equivalent Queries

Two queries that give the same result on any database are said to be **equivalent**. There can be many ways to write a query. Query optimizers find the most efficient way to compute a result by using a more efficient equivalent query rather than the one specified.