CSC343
Guide

Pierre-William Lessard

Contents

Chapter 1

Relations

1.1   The Relational Model

> **Definition 1.1: Relation**
>
> A set of tuples that have a certain number of attributes, or columns.
> Each tuple in the relation has the same number of attributes, and each attribute has a unique name.
>
> A relation is defined by schemas which indicate the structure of a relation.
> Instances of relations include tuples which all follow a relation's schema.

Relations are traditionally thought of as tables. Where the rows are denoted as **tuples** and the columns as "attributes". The "arity" of a relation is its number of attributes, and its "cardinality" is the number of tuples within a relation instance.

In Relational Algebra, a relation is a set of tuples, meaning:

- There are no duplicate tuples

- The order of tuples does not matter

> **Definition 1.1.1: Key**
>
> A key for a relation, R, is defined as a set of attributes $a_1, a_2, \ldots, a_n$ such that:
>
> - $\forall$ tuples $t_1, t_2 \in R.\ \exists i \in [1, n] \cap \mathbb{N}.\ t_1[a_i] \neq t_2[a_2]$
>   This means that no two tuples may have all the attributes denoted in the key, agree.
>
> - There is no subset of $a_1, a_2, \ldots, a_n$ with this same property

This means that keys uniquely identify tuples in a relation's instance, as there must not exist another tuple with the same values for the attributes indicated by the key.

Keys are denoted in the schema by underlining the attributes of the key. A **superkey** is any set of attributes which meet the first property defined above, yet not necessarily the second.

1.2   Relational Algebra

Relational algebra is a type of algebra where the operands are instances of relations. It is a formal system for manipulating and querying data stored in relational databases. It is based on a set of fundamental operations that allow us to combine data from multiple tables in a declarative way. Some of the most common operations

in relational algebra include selection, projection, union, intersection, and difference. These operations allow us to specify the data we want to retrieve from a database, as well as the relationships between different data sets. Relational algebra is an important theoretical foundation for the design and implementation of database systems.

Projection

Projection refers to the operation that extracts certain columns from a relation (table). For example, if you have a relation with columns "name", "age", "gender", and "country", and you only want to see the "name" and "age" columns, you would use the projection operation to create a new relation that only contains those columns. This operation is useful for simplifying and organizing the information in a relation, and for focusing on the specific data that you are interested in.

Definition 1.2.1: Projection

Projection is denoted with the π symbol. Let R be an arbitray relation with some subset of attributes, $a_1, a_2, \ldots, a_n$. Projecting $a_1, a_2, \ldots, a_n$ onto R is notated as:

$$\pi_{a_1, a_2, \ldots, a_n} R$$

The result is a new relation with all the same tuples as R, but only including attributes $a_1, a_2, \ldots, a_n$

Selection

Selection is an operation that selects a subset of tuples from a relation. It is typically denoted by the symbol σ, and it is typically used in combination with other operations such as projection and soon to be learned join. Selection is a fundamental operation in relational algebra and is used to specify the desired subset of a relation in order to answer a query. For example, consider a relation R with attributes A, B, and C, and suppose we want to select all tuples from R where the value of attribute A is greater than 5. We can use the select operator to return a relation where this is true.

Definition 1.2.2: Selection

Selection is denoted with the σ symbol. Let R be an arbitray relation with some subset of attributes, $a_1, a_2, \ldots, a_n$. Let $L(a_1, a_2, \ldots, a_n)$ be a logical statement incorporating a subset attributes of R. Selecting tuples which render $L(a_1, a_2, \ldots, a_n)$ true in R is notated as:

$$\sigma_{L(a_1, a_2, \ldots, a_n)} R$$

The result is a new relation with all the same tuples as R, but only including tuples where $L(a_1, a_2, \ldots, a_n)$ rendered true.

Overall, the selection operation is a powerful tool for filtering and selecting subsets of data from a relation, based on specified conditions. It is often used in combination with other operations in relational algebra, such as projection and join, to manipulate and query relational data.

Cartesian Product

The Cartesian product is a binary operation that combines every row of one table with every row of another table. The resulting table, known as the Cartesian product, has a number of rows equal to the product of the number of rows in the two input tables. This operation is also known as a simple join.

> **Definition 1.2.3: Cartesian Product (Join)**
>
> Cartesian Product is denoted with the $\times$ symbol. Let R_1 and R_2 be arbitray relations with some set of attributes A_1 and A_2 respectively. Joining tuples R_1 and R_2 is notated as:
>
> $$R_1 \times R_2$$
>
> The result is a new relation which we will denote $R_\times$. The following defines the new relation:
>
> $$\forall \text{ tuples } t_1 \in R_1. \forall \text{ tuples } t_2 \in R_2. \exists \text{ a tuple } t_\times \in R_\times. t_1|t_2 = t_\times$$
>
> Where $t_1|t_2$ is the tuple such that the attributes of either tuple is concatenated.

This means that each tuple in the first relation is paired with every tuple in the next relation. Therefore, the new table includes all possible combinations of rows from the two input tables. Meaning, given two relations R_1 and R_2 with cardinalities n_1 and n_2, the cardinality of $R_1 \times R_2$ is $n_1 \cdot n_2$. The arities of the resulting relation is the sum of the two relations' original arities.

Example 1.2.1

Let's say we have two relations, Students and Courses, with the following schemas:

- Students(ID, Name, Age, Gender)

- Courses(Code, Name, Credits)

The cartesian product of these two relations would be a new relation, StudentCourses, with the schema:
StudentCourses(ID, Name, Age, Gender, Code, CourseName, Credits)

The relation would have a row for every possible combination of a student and a course, with the student's ID, name, age, and gender paired with the course's code, name, and credits.
Let us say we have an instance of relations Students and Courses, denoted by S and C:

ID	Name	Age	Gender
1	Alice	18	F
2	Bob	19	M
...	...	...	...

Code	CourseName	Credits
CS101	Introduction to Computer Science	4
MATH201	Calculus I	4
ENGL101	Composition	3
...	...	...

The result of $S \times C$ would be the following:

ID	Name	Age	Gender	Code	CourseName	Credits
1	Alice	18	F	CS101	Introduction to Computer Science	4
1	Alice	18	F	MATH201	Calculus I	4
1	Alice	18	F	ENGL101	Composition	3
2	Bob	19	M	CS101	Introduction to Computer Science	4
2	Bob	19	M	MATH201	Calculus I	4
2	Bob	19	M	ENGL101	Composition	3
...	...	...	...	...	...	...

This table shows every possible combination of a student and a course, with each row representing a student taking a specific course. Note that the original relation schemas are repeated in the new schema to indicate that each attribute belongs to a specific relation.

Natural Join

A natural join is a way of combining two relations into a single relation. The resulting relation includes only pairs of tuples who values of similar attributes agree. Natural joins are useful when combining two relations which agree on some attribute.

Definition 1.2.4: Natural Join

Natural Join is denoted with the $\bowtie$ symbol. Let R_1 and R_2 be arbitray relations with some set of attributes A_1 and A_2 respectively. Assume their exists some attribute a that is in both A_1 and A_2. Natural joining tuples R_1 and R_2 is notated as:

$$R_1 \bowtie R_2$$

The result is a new relation. Let A be the set of attributes that agree between A_1 and A_2, meaning $A = A_1 \cap A_2$. The following defines the new relation in terms of select (σ) and cartesian product ($\times$):

$$R_1 \bowtie R_2 = \sigma_{\forall a \in A. R_1[a] = R_2[a]}(R_1 \times R_2)$$

Meaning the natural join of two relations is simply their cartesian product, where the pairs selected agree on the shared attributes. In the final relation, the shared attributes are express as one, since we already know they agree.

This property of natural join makes it very useful when combining two relations who are defined as agreeing on some attribute (which we will explain in the next chapter on integrity constraints). The following example illustrates this on two relations, $R(A, B, C)$ and $S(C, D, E)$. Note that these relations agree on the attribute 'C'.

Example 1.2.2

R

A	B	C
1	2	3
4	5	6

$\bowtie$

S

C	D	E
3	7	8
6	9	10

=

R $\bowtie$ S

A	B	C	D	E
1	2	3	7	8
4	5	6	9	10

This example shows two tables R and S, which are joined using the natural join operation. The resulting table contains all combinations of rows from R and S where the values in the C column match.

Important properties of Natural Join are as follows:

- Natural join is associative, *i.e.* for all arbitrary relations R,S,T: $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$

- Natural join is commutative, *i.e.* $R \bowtie S = S \bowtie R$

- Natural join can produce unintended results when there are attributes of the same name which are not meant to be related.

- When no attributes match between the two relations, natural join acts the same as cartesian product

1.3 Integrity Constraints

Referential integrity constraints are important in constraining more custom rules onto a relation. They sometimes refer to some reference between relations. This can be simple as an instance's attribute's values needing to be a

subset of some other relations. This can become much more complicated through using relational algebra.

Referential Integrity Constraints

A referential integrity constraint is a rule that is used to ensure that relationships between tables in a database are maintained. It is a way of ensuring that data in one table references data in another table and that the data in the other table is valid and consistent. For example, if a table contains a foreign key that references a primary key in another table, a referential integrity constraint can be used to ensure that the value in the foreign key column is always a valid primary key value in the other table. This helps to maintain the integrity of the data and ensures that the relationships between the tables are consistent. More rigorously we can define it as follows:

Definition 1.3.1: Referential Integrity Constraint

Let R_1 and R_2 be two unique relations. Let X and Y be arbitrary sets of attributes of equal length in R_1 and R_2 respectively. The referential integrity constraint between these two relations, denoted $R_1[X] \subseteq R_2[Y]$ implies:

$$\forall \text{ values } v_1 \in R_1[X]. \exists \text{ a value } v_2 \in R_2[Y]. v_1 = v_2$$

Meaning the set of values in $R_1[X]$ are a subset of the values in $R_2[Y]$

Referential Integrity Constraints are important in defining the relationship between two relations. When Y is a key we call the referential integrity constraint a **Foreign Key Constraint**