COMPUTER ORGANIZATION AND ARCHITECTURE

Lecture 4

The Relational Model

Dr Victoria Mukami

INTRODUCTION

This lecture focuses on the relational database model. We focus on how relationships work, and the emphasis is largely on the logical view of data rather than the conceptual design. We will review various relational database terminology different types of keys and integrity rules. This will then give way to relational algebra and the various relationships within the relational database.

Learning objectives

By the end of this topic, you should be able to:

1. Describe the relational model terminologies
2. Understand the purpose of keys in a database
3. Describe the role of relational algebra in a database
4. Describe the different relational algebra operators.

OVERVIEW

Previously we reviewed the relational data model which was introduced in the 70s and was a breakthrough. The relational data model specifically was introduced to be able to provide data independence, deal with consistency and redundancy and enable the use of data manipulation languages [2]. During this period, the structured query language (SQL) and various database management systems were developed. As we have seen throughout the lectures, the relational model is the foundation with which this unit lies and as such, we need to understand relational database inner workings more. We start with some relational database terminology before we move to relational algebra.

BASIC TERMINOLOGY

The relational model is largely based on the mathematical concept of a relation [2]. Remember from the previous lectures we saw that relation is the same as a table. The relational model picks mathematical concepts such as set theory and logic to explain its working. We first define a couple of terms.

Relation

As indicated previously a relation is similar to a table and contains rows and columns. The table is considered as the user view as that is how the user views the database. Each table row is considered as a single entity while a column has specific attributes

based on the column name [2]. A table row is also known as a tuple. A column values all share the same data type.

Attribute

This is a characteristic attributed to an entity. This is like the column in a table. We will cover attributes extensively in lecture 5.

Domain

This is the set of allowable values for the attributes [2]. For instance, when entering the age of people considered a youth, the range may be between 18-35 meaning the domain is (18,35). Those would be the only permissible values for that specific column

Degree

The degree of a relation or a table is the number of attributes that the table contains [2]. For instance, when working with a STUDENT table that has 5 columns (STUD_REG_NO, STUD_FNAME, STUD_LNAME, STUD_GENDER, STUD_COUNTRY) then the five attributes on each entity is considered as degree five.

Cardinality

Cardinality on the other hand is the number of tuples or records it contains [2]. Remember a tuple is one row of a table. This, therefore, means that cardinality will change as records are added or deleted.

KEY

In the last lecture, we reviewed the key or identifier of a relation. A key is used to uniquely identify a tuple within a relation. A key can be one or a combination of more than one attribute. For instance, a REGISTRATION table may have a STUD_REG_NO and CRS_CODE as the primary key that has been extracted from the Student and Course tables respectively. There are different types of keys, and these are discussed below:

Super Key

One attribute or several combined attributes uniquely identify each tuple [1]. For instance, in the STUDENT table, the STUD_REG_NO attribute can easily determine other attributes within the row.

Candidate Key

This is a type of super key and is considered the minimal super key [1]. The candidate key can be used as a super key. For instance, if the STUDENT table also held the students' national ID as STUD_NAT_ID, then this could be considered a candidate key.

Primary Key

A candidate key is selected to uniquely identify all other attributes within a row [1]. A primary key must be unique and must contain a value. A good example is STUD_REG_NO of the STUDENTS table. For one to be a student, they are assigned a student registration number and this number is used to uniquely identify the student which is the same for our STUDENT table.

Foreign Key

An attribute or combination of whose values match the primary key in another table [1]. For instance, A university student could be hosted by a department. Therefore, the primary key for the DEPARTMENT table being DEPT_ID would then be introduced as a foreign key to the STUDENT table. Foreign keys are used to enforce referential integrity which ensures that each reference to an entity by another entity is valid [1].

Secondary Key

An attribute that is used for data retrieval [1]. This is an attribute that can be used to pull up records. For instance, the telephone number could be used to pull up a student's records.

RELATIONAL ALGEBRA

Relational algebra is a theoretical way of manipulating tables using relational operators [1]. SQL is used to accomplish relational algebra operations [1]. There are five fundamental operations used in relational algebra. This includes Select, Project, Product, Union, and Difference. There are three more operations i.e., Join, Intersect, and Division that can be accomplished using the five fundamental operations. The Select and Project are considered as Unary operations as they only operate on one table (relation) [2]. We now review each.

Select

This is also known as the Restrict. It will return values from only one table. It will show the result of all rows or rows that match a specific criterion. This operator returns only the horizontal values and does not limit the attributes returned [1]. The select operator is represented by the sigma symbol (σ).

SELECT only STUD_BALANCE >5000

The statement above will show all the rows in the table where a student's balance is greater than 5000.

$\sigma_{\text{stud\_balance>5000}}(\text{fees})$

The statement is the mathematical representation that will return all rows from the fees table whose balance is greater than 5000. Note the stud_balance>5000 is in subscript.

Project

The project operator will show all values of selected attributes. This is the opposite of the select operator. While select will return the row or horizontal subset, the project will return the vertical results. The project will return all rows but will limit them to a specific column. Using the FEES table that contains STUD_REG, SEMESTER, STUD_BALANCE the following would then happen. The project operator is denoted by the pi symbol (π).

PROJECT STUD_REG and STUD_BALANCE

This statement would return all values of the STUD_REG and STUD_BALANCE.

$\pi_{\text{stud\_reg, stud\_balance}}(\text{fees})$

The statement is the mathematical representation that will return the STUD_REG and STUD_BALANCE columns

Product

The product operator will return all possible pairs from any two tables [1]. The product operator is also known as the Cartesian Product. Assuming the COURSE table has 3

rows, and the FEES table has 5 rows then the final list by the product operator would have 3 × 5 = 15 rows. The product operator is denoted by the × symbol.

course × fees

The statement is the mathematical representation that will return the COURSE and FEES tables combined

Union

The union combines all the rows from two tables and excludes duplicate rows. For the union to work, the tables must have similar characteristics [1]. The number of columns and domains must be compatible as well with the data types. If the OLD_PRODUCT table and NEW_PRODUCT TABLE needed to be merged, then the union operator would be used so that if OLD_PRODUCT has 10 rows and NEW_PRODUCT had 20 rows then the new table would have 30 rows. Union is represented by the ∪ symbol.

old_product ∪ new_product

The resulting table would combine all the rows in both tables.

Assuming the two tables were not compatible, then the project operator can be used together with the union as shown below to combine specific columns.

$\pi_{\text{old\_product\_name}}$ (old_product) ∪ $\pi_{\text{new\_product\_name}}$ (new_product)

Difference

The difference operator will show all tables not found in the other table [1]. Similar to union the tables have to be compatible. The difference operator is represented by the − symbol.

OLD_PRODUCT − NEW_PRODUCT

This mathematical statement will only show unique rows by removing products in OLD_PRODUCT that are also found in NEW_PRODUCT and displaying the remaining rows in the OLD_PRODUCT.

OLD_PRODUCT − NEW_PRODUCT is not the same as

NEW_PRODUCT − OLD_PRODUCT.

If the tables are not compatible then the project can be used like in the union.

π old_product_name (old_product) − π new_product_name (new_product)

Intersect

The intersect will only show tables that appear on both tables. Both have to be union-compatible to display results [1]. If columns have different data types, then the intersect operator cannot be used.

OLD_PRODUCT ∩ NEW_PRODUCT

This mathematical statement will show only the unique rows between the two tables.

In the case that the tables are not union-compatible, then the project statement can be used.

π old_product_name (old_product) ∩ π new_product_name (new_product)

Join

Join allows for information to be intelligently combined into two or more tables [1]. There are five main types of joins.

Natural Join: Links tables by only choosing rows with common values in their common attribute. The first step includes the use of the product operator to combine both tables. Next, the select is used to show results where a specific column matches. Finally, the project operator is used to only show a single copy of the attribute. The natural join also known just as the join is denoted by the ⋈ symbol.

student ⋈ lecturer

This statement would combine the two and only return similar values.

Equijoin: will combine tables based on the equality condition. It does not remove duplicate columns. This uses the equal symbol or operator.

Theta join: any other join that does not use the equal sign is known as the theta join.

Left outer join: It will show all the results including any that do not have a matching value in the second table. The left join is represented by the ⟕ symbol.

Right outer join: This will show all the results including any that do not have a matching value in the first table. The right outer join is represented by the ⟖ symbol.

Divide

The divide operator is used to answer questions about one set of data being associated with all values of data in another set of data [1]. It uses one two-column table as the dividend and a one-column table as the divisor. The divide operator is represented by the ÷ symbol.

SUMMARY

We have fully reviewed the relational table during this lecture. We first reviewed the basic terminology of the relational model including relation, attribute, domain, degree, and cardinality. Next, we conducted a review of keys and specifically, we looked at primary keys, foreign keys, candidate keys, and secondary keys. Finally, we focused on relational algebra and reviewed the Select, Project, Product, Union, Difference, Join, Intersect, and Division operators.

DISCUSSION TOPIC

For the lands office that we were designing a database for, we would like you as the student to identify five tables that can serve the office. Identify what primary keys each of the tables would have and if any foreign keys would be used within the same tables. Keep in mind that a foreign key is used to ensure that there is consistency

REFERENCES

[1] Database systems: design, implementation, and management, Coronel, C., & Morris, S, Cengage Learning, 2019.

[2] Database Systems: A Practical Approach to Design, Implementation, and Management, Connolly, T., & Begg, C., Pearson, 2015.

[3] Fundamentals of database systems, Elmasri, R., & Navathe, S. B., Pearson Education Limited, 2016.