UNIT-II

RELATIONAL DATA MODEL AND LANGUAGE

Relational Data Model

'*The* '**Relational Database Model** is the most common model in industry today.

A relational database is based on the relational model developed by E.F. Codd.

A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organized into tables. A table is a collection of records and each record in a table contains the same fields.

Properties of the Relational database model

Properties of Relational Tables:

1. Data is presented as a collection of relations.

2. Each relation is depicted as a table.

3. Columns are attributes that belong to the entity modelled by the table (ex. In a student table, you could have name, address, student ID, major, etc.).

4. Each row ("tuple") represents a single entity (ex. In a student table, John Smith, 14 Oak St, 9002342, Accounting, would represent one student entity).

5. Every table has a set of attributes that taken together as a "key" (technically, a "superkey") uniquely identifies each entity (Ex. In the student table, “student ID” would uniquely identify each student – no two students would have the same student ID).

Concepts

Tables – In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

Tuple – A single row of a table, which contains a single record for that relation is called a tuple.

Relation instance – A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

Relation schema – A relation schema describes the relation name (table name), attributes, and their names.

Relation key – Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

Attribute domain – Every attribute has some pre-defined value scope, known as attribute domain.

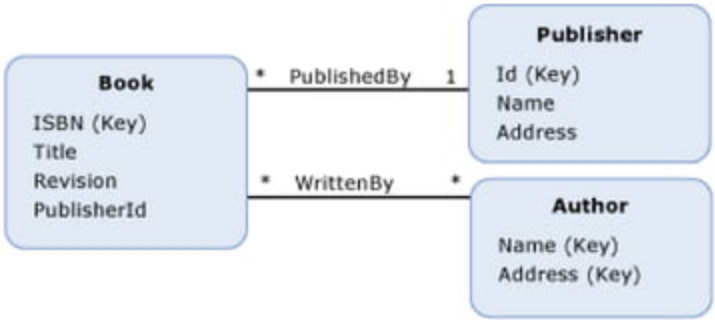
Relational Integrity Constraints

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called Relational Integrity Constraints. There are three main integrity constraints –

1. Key constraints
2. Domain constraints
3. Referential integrity constraints

Example

The diagram below shows a conceptual model with two associations: WrittenBy and PublishedBy. The Book entity type has a property, PublisherId, that references the entity key of the Publisher entity type when you define a referential integrity constrainst on the PublishedBy association.



Key Constraints

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

Key constraints force that −

in a relation with a key attribute, no two tuples can have identical values for key attributes.

a key attribute can not have NULL values.

Key constraints are also referred to as Entity Constraints.

Domain Constraints

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Referential Integrity Constraints

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

...

Referential integrity is a property of data which, when satisfied, requires every value of one attribute (column) of a relation (table) to exist as a value of another attribute (column) in a different (or the same) relation (table).

A **foreign key** (FK) is a column or combination of columns that is used to establish and enforce a link between the data in two tables.

Integrity

Data **integrity** refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data.

Relational Database

Relational database systems are expected to be equipped with a query language that can assist its users to query the database instances.

There are two kinds of query languages –

1. Relational algebra

2. Relational calculus.

Relational Algebra

The main application of relational algebra is providing a theoretical foundation for relational databases, particularly query languages for such databases, chief among which is SQL.

Relational algebra is a theoritcal procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows –

Select	Project	Union
Set Difference	Cartesian product	Rename

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation – $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like – =, ≠, ≥, < , >, ≤.

For example –

σsubject = "database"(Books)

Output – Selects tuples from books where subject is 'database'.

σsubject = "database" and price = "450"(Books)

Output – Selects tuples from books where subject is 'database' and 'price' is 450.

σsubject = "database" and price = "450" or year > "2010"(Books)

Output – Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation – ∏A1, A2, An (r)

Where A1, A2 , An are attribute names of relation r.

Duplicate rows are automatically eliminated, as relation is a set.

For example –

∏subject, author (Books)

- **Output:** Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as –

Notation – r U s

Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold –

- r, and s must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

For Example: ∏ author (Books) ∪ ∏ author (Articles)

- Output – Projects the names of the authors who have either written a book or an article or both.

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation – r − s

Finds all the tuples that are present in r but not in s.

For Example: ∏ author (Books) − ∏ author (Articles)

- Output – Provides the name of authors who have written books but not articles.

Cartesian Product (X)

Combines information of two different relations into one.

Notation – r X s

Where r and s are relations and their output will be defined as –

σauthor = 'nishant'(Books X Articles)

- Output – Yields a relation, which shows all the books and articles written by nishant.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho ρ.

Notation – ρ x (E)

Where the result of expression E is saved with name of x.

Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms

- Tuple Relational Calculus (TRC)
- Domain Relational Calculus (DRC)

Tuple Relational Calculus

Filtering variable ranges over tuples.

Notation – {T | Condition}

Returns all tuples T that satisfies a condition.

For example –

{ T.name | Author(T) AND T.article = 'database' }

Output – Returns tuples with 'name' from Author who has written article on 'database'.

Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation –

{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where a1, a2 are attributes and P stands for formulae built by inner attributes.

For example – **{< article, page, subject > | ∈ Sahoo ∧ subject = 'database'}**

Output – Yields Article, Page, and Subject from the relation Sahoo, where subject is database.

THANKS

Connect: @nishantmunjal