RDBMS:OLAP & OLTP

-Aman Khandwe

February 25, 2022

Abstract

A relational database is a digital database that uses the relational data model. A relational database management system is a system for maintaining relational databases (RDBMS). Many relational database systems allow you to query and maintain the database using SQL (Structured Query Language).It's a programme that lets us create, delete, and update relational databases. A relational database is a database system that stores and retrieves data in the form of rows and columns in a tabular format. The main DBMS, such as SQL, My-SQL, and ORACLE, are all based on relational DBMS concepts.The fact that the values of each table are connected to each other is the cornerstone of relational DBMS. It is capable of handling enormous amounts of data and easily simulating queries.Here in this paper we will look at the realtional model and will look at the terminologies related to RDBMs and how to store objects in relational database, followed by advantages, disadvantages, and some applications of the same, and finally we will try to have a deeper look at the two major applications of RDBMS, OLAP (Online analytical processing) and OLTP (Online transaction processing) (Online transaction processing).

Introduction

RDBMS is a concept that is used to give the user with the ability to store and retrieve data that is connected with any other relation, also known as the Relational. A relation in a relational database is based on a relational scheme and is made up of a number of properties. A relational database is composed of several relations and a relational database architecture. Users have the ability to access and alter the data stored in the database in a convenient and functional manner. Furthermore, the DBMS exerts centralised control over the database, prohibits unauthorised users from accessing the data, and secures data privacy.The fact that the values of one table are connected to others is the foundation of relational DBMS. It is capable of handling bigger amounts of data and quickly simulating queries. Relational Database Management Systems ensure data integrity by imitating the following characteristics:

1. Entity Integrity: No two records in the database table may be identical.
2. Referential Integrity: Only those rows of those tables that are not utilised by other tables can be erased. Otherwise, data discrepancy may occur.
3. User-defined Integrity: Rules based on confidentiality and access that are established by the users.
4. Domain integrity: The database table columns are contained inside certain specified limitations based on default values, data type, or ranges.

In RDBMS, data must be saved in a DB file in tabular form, that is, in the form of rows and columns. Each table row is referred to as a record/tuple. The cardinality of the table refers to the collection of such items. Each column in the table is referred to as an attribute/field. The arity of the table is a collection of similar columns. There can be no duplicate records in the DB table. By employing a candidate key, data duplication is eliminated. A Candidate Key is a collection of properties that must be present in order for each record to be uniquely identified. Tables are linked to one another via foreign keys.Database tables also support NULL values, which means that if the values of any of the table's elements are not filled in or are missing, the value becomes a NULL value, which is not equal to zero. RDBMS often include data dictionaries and metadata collections that aid in data management. These provide for the programmatic support of well-defined data structures and relationships. Data storage management is a typical RDBMS functionality, and it has been characterised by data objects ranging from binary large object strings to stored procedures. This type of data item extends the scope of conventional relational database operations and may be handled in a number of ways by different RDBMSes.For acessing data in RDBMs SQL is used. Data manipulation language and data definition language statements are its primary language components. RDBMS employs complicated algorithms that allow several concurrent users to access the database while preserving data integrity. Another overlay function provided by the RDBMS for the fundamental database when used in business settings is security management, which enforces policy-based access.

Relational Model

The relational model is based on the mathematical idea of a relation, which is represented practically as a table. Relations are utilized in the relational model to store information about the items that will be represented in the database. A relation is represented as a two-dimensional table, with rows corresponding to individual records and columns corresponding to attributes. Attributes can occur in any sequence, and the connection remains the same and so conveys the same meaning. Domains are a very important aspect of the relational model. A domain is used to specify each attribute in a relation. Domains might be unique for each characteristic, or they can be shared by two or more attributes. The domain idea is significant because it allows the user to specify the meaning and source of values that attributes can carry in a centralized location. As a result, the system has more information accessible to it when performing a relational action, and operations that are semantically inaccurate can be avoided. The rows or tuples in the table are the elements of a relation. Tuples can be arranged in any sequence, and the connection will remain the same and hence communicate the same meaning. The structure of a relation, together with a definition of the domains and any other constraints on potential values, is referred to as its intension, which is normally fixed unless the meaning of the relation is updated to incorporate more characteristics. The tuples are referred to as a relation's extension (or state), which varies over time. A unary relation, also known as a one-tuple, is a relation with only one attribute and has degree one. Binary refers to a relationship with two qualities, ternary refers to a relationship with three attributes, and n-ary refers to a relationship with more than three attributes. A relation's degree is a characteristic of the relation's intension. The cardinality of the relation, refers to the number of tuples in the relation, which changes when tuples are added or removed. The cardinality of a connection is a feature of its extension that is defined by the specific instance of the relation at any given time.

A relation has the following properties:

- The relation has a unique name that differs from the names of all other relations in the relational schema.
- There is only one atomic (single) value in each cell of the relation.
- Each attribute is given a unique name.
- An attribute's values are all from the same domain.
- There are no duplicate tuples; each tuple is unique.
- The order of the attributes is irrelevant.
- The order of tuples has no theoretical importance.

Integrity Constraints

A data model contains two additional components: a manipulative part that defines the sorts of operations that may be performed on the data, and a set of integrity constraints that verify the data's accuracy. The relational integrity restrictions will be discussed in this section. Since each attribute has a domain, there are constraints (known as domain constraints) that limit the set of values that may be assigned to the attributes of relations. There are also two key integrity rules, which are limits or limitations that apply to all database instances. Entity integrity and referential integrity are the two main rules of the relational model. Multiplicity and general constraints are two further forms of integrity constraints.It's important to grasp the idea of nulls before we can establish entity and referential integrity. Null denotes the absence of a value for an attribute that is presently unknown or irrelevant to this tuple.A null is not the same as a numeric value of zero or a text string with spaces; all are values, but a null signifies the lack of a value. So, nulls must be handled differently from other values. Now let us talk about the relational integrity rules.

Entity Integrity The main keys of base relations are subject to the first integrity rule. No attribute of a main key can be null in a base relation. A primary key, by definition, is a single identifier that is used to uniquely identify tuples. This indicates that no subset of the main key is adequate to identify tuples uniquely. Allowing a null for any component of a primary key implies that not all attributes are required to differentiate between tuples, which is contrary to the primary key's definition.

Referential Integrity Foreign keys are subject to the second integrity rule. If a relation has a foreign key, the value of the foreign key must either match a candidate key value of some tuple in the home relation or be completely null.Referential integrity constraints are used to preserve consistency between tuples in two relations and are expressed between two relations or tables.

General Constraints These are the additional rules set by database users or database administrators that define or limit a particular area of the business.Additionally, users can set additional constraints that the data must meet. For example, if the number of people who can work at a branch office is limited to 20, the user must be able to define this general limitation and expect the DBMS to enforce it. If the number of staff presently allocated to a branch is 20, it should not be able to add a new member of staff to the Staff relation in this situation. However, the amount of support for generic restrictions varies considerably from one system to the next.

Storing Objects in a Relational Database

Using an RDBMS as the underlying storage engine is one way to achieve persistence with an object-oriented programming language like C++ or Java. This necessitates mapping class instances (i.e., objects) to one or more tuples distributed across one or more relations.Consider the inheritance structure depicted in Figure 1, which includes a superclass called Staff and three subclasses called Manager, SalesPersonnel, and Secretary.

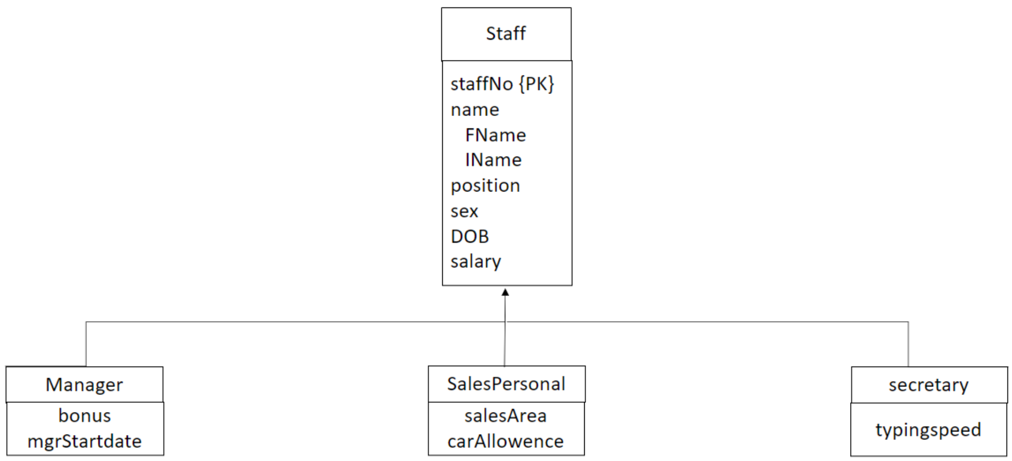

Figure 1: Inheritance hierarchy for Staff.

We have two main activities to undertake when dealing with this sort of class hierarchy:

1. Design the relations to represent the class hierarchy.
2. Create a plan for how objects will be accessible, which includes:
   - developing code that decomposes objects into tuples and stores them in relations;
   - writing code that reads tuples from relations and reconstructs the objects.

Now let us look at in depth about the above two mentioned tasks-

1.Mapping Classes to Relations

There are several strategies for mapping classes to relations, each of which results in a loss of semantic information. The code used to make objects persistent and read them back from the database is determined by the strategy used. We consider the following three options:

1. Map each class or subclass to a relation.
2. Map each subclass to a relation.
3. Map the hierarchy to a single relation.

1.1 Map each class or subclass to a relation

One method is to map each class or subclass with a relation.This would result in the following four relationships for the hierarchy shown in Figure 1:

1. Staff (staffNo, fName, lName, position, sex, DOB, salary).
2. Manager (staffNo, bonus, mgrStartDate).
3. SalesPersonnel (staffNo, salesArea, carAllowance).
4. Secretary (staffNo, typingSpeed).

Although we presume that the RDBMS supports the underlying data type of each attribute, this may not be the case, in that case we will need to create additional code to manage the translation from one data type to another. Unfortunately, we have lost semantic information with this relational schema: it is no longer evident which relation represents the superclass and which relation represents the subclasses. As a result, we'd have to embed this knowledge into each programme, which, as we've discussed before, can lead to code duplication and inconsistency.

1.2 Map each subclass to a relation

A second option is to map each subclass with a relation. This would result in the following three relations for the hierarchy shown in Figure 1:

1. Manager (staffNo, fName, lName, position, sex, DOB, salary, bonus, mgrStartDate)

2. SalesPersonnel (staffNo, fName, lName, position, sex, DOB, salary, salesArea, carAllowance)

3. Secretary (staffNo, fName, lName, position, sex, DOB, salary, typingSpeed)

Again, semantic information has been lost in this mapping: it is no longer evident whether these relations are subclasses of a single generic class. To get a list of all employees in this example, we'd have to choose the tuples from each relation and then combine the results.