Database Model in Data Base Management System (DBMS)

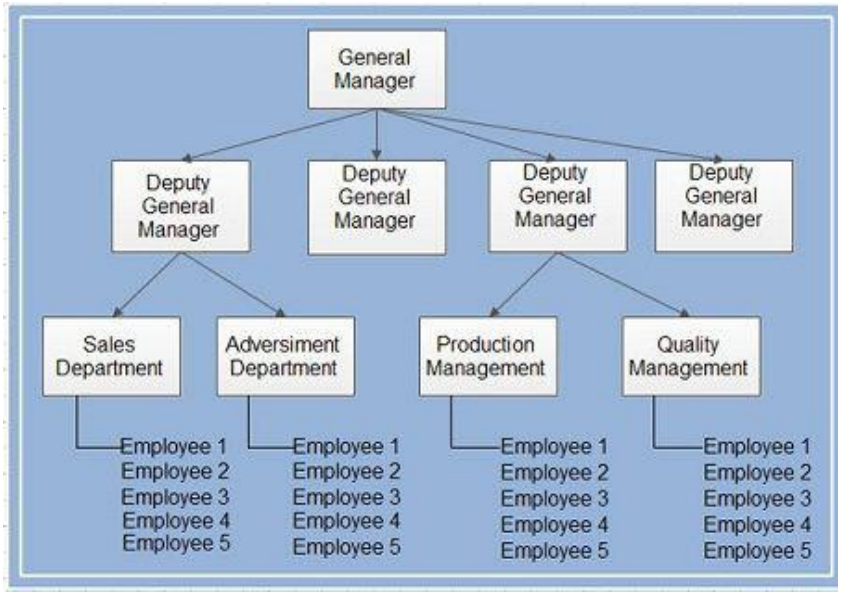
Table of Contests

Hierarchical Database Model

Hierarchical Database model is one of the oldest database models, dating from late 1950s. One of the first hierarchical databases Information Management System (IMS) was developed jointly by North American Rockwell Company and IBM. This model is like a structure of a tree with the records forming the nodes and fields forming the branches of the tree.

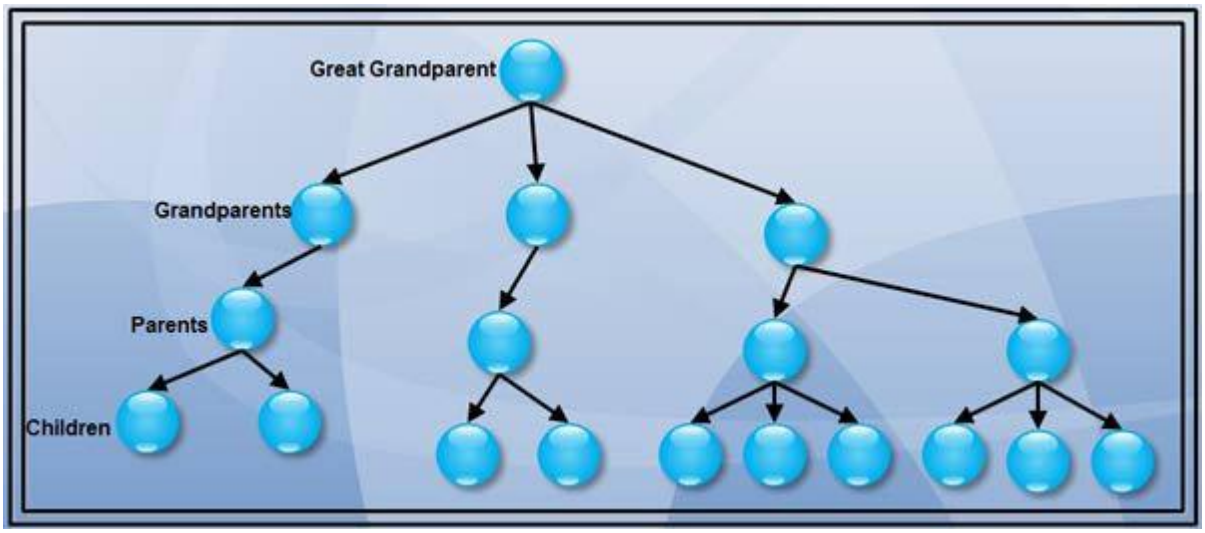
The hierarchical model organizes data elements as tabular rows, one for each instance of entity. Consider a company's organizational structure. At the top we have a General Manager (GM). Under him we have several Deputy General Managers (DGMs). Each DGM looks after a couple of departments and each department will have a manager and many employees.

When represented in hierarchical model, there will be separate rows for representing the GM, each DGM, each department, each manager and each employee. The row position implies a relationship to other rows. A given employee belongs to the department that is closest above it in the list and the department belongs to the manager that is immediately above it in the list and so on as shown.



In the hierarchical data model, records are linked with other superior

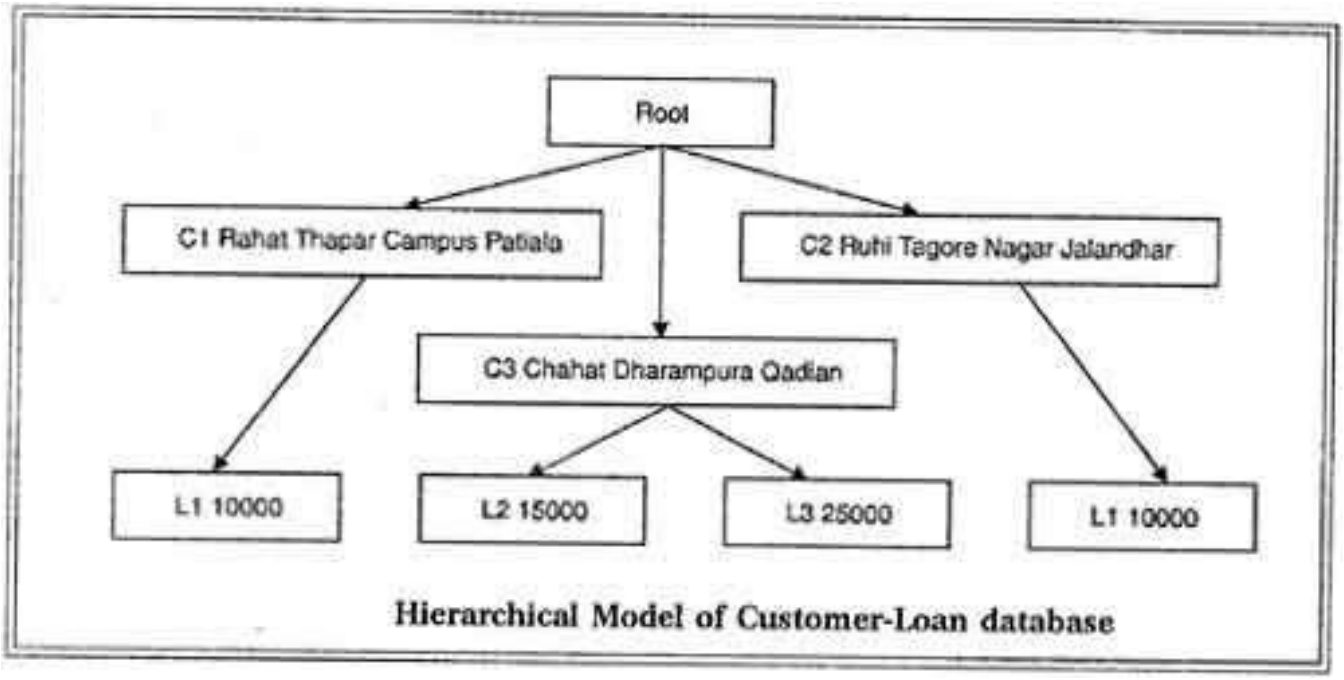
records on which they are dependent and also on the records, which are dependent on them. A tree structure may establish one-to-many relationship. Figure illustrates the structure of a family. Great grandparent is the root of the structure. Parents can have many children exhibiting one to many relationships. The great grandparent record is known as the root of the tree. The grandparents and children are the nodes or dependents of the root. In general, a root may have any number of dependents. Each of these dependent may have any number of lower level dependents, and so on, with no restriction of levels.



The different elements (e.g. records) present in the hierarchical tree structure have Parent-Child relationship. A Parent element can have many children elements but a Child element cannot have many parent elements. That is, hierarchical model cannot represent many to many relationships among records.

Another example, of hierarchical· model is shown. It shows a database of Customer-Loan, here a customer

can take multiple loans and there is also a provision of joint loan where more than one person can take a joint loan. As shown, CI customer takes a single loan Ll of amount 10000 jointly with customer C2. Customer C3 takes two loans L2 of amount 15000 and L3 of amount 25000.



Hierarchical Model of Customer-Loan database

Sample Database

In order to understand the hierarchical data model better, let us take the example of the sample database consisting of supplier, parts and shipments. The record structure and some sample records for supplier, parts and shipments elements are as given in following tables.

The Supplier records

Sno	Name	Status	City
S1	Suneet	20	Qadian
S2	Ankit	10	Amritsar
S3	Amit	10	Amritsar

The Part records

Pno	Name	Color	Weight	City
PI	Nut	Red	12	Qadian
P2	Bolt	Green	17	Amritsar
P3	Screw	Bule	17	Jalandhar
P4	Screw	Red	14	Qadian

The Shipment records

Sno	Pno	Qty
S1	P1	250
S1	P2	300
S1	P3	500
S2	P1	250
S2	P2	500
S3	P2	300

We assume that each row in Supplier table is identified by a unique SNo (Supplier Number) that uniquely identifies the entire row of the table. Likewise each part has a unique Pno (Part Number). Also we assume that no more than one shipment exists for a given supplier/part combination in the shipments table.

Hierarchical View for the Suppliers-Parts Database

The tree structure has parts record superior to supplier record. That is parts form the parent and supplier forms the children. Each of the four trees figure, consists of one part record occurrence, together with a set of subordinate supplier record occurrences. There is one supplier record for each supplier of a particular part. Each supplier occurrence includes the corresponding shipment quantity.

For example, supplier S3 supplies 300 quantities of part P2. Note that the set of supplier occurrences for a given part occurrence may contain any number of members, including zero (for the case of part P4). Part PI is supplied by two suppliers, S1 and S2. Part P2 is supplied by three suppliers, S1, S2 and S3 and part P3 supplied by only supplier SI as shown in figure.

Operations on Hierarchical Model

There are four basic operations Insert, Update, Delete and Retrieve that can be performed on each model. Now, we consider in detail that how these basic operations are performed in hierarchical database model.

Insert Operation: It is not possible to insert the information of the

supplier e.g. S4 who does not supply any part. This is because a node cannot exist without a root. Since, a part P5 that is not supplied by any supplier can be inserted without any problem, because a parent can exist without any child. So, we can say that insert anomaly exists only for those children, which has no corresponding parents.

Update Operation: Suppose we wish to change the city of supplier S1 from Qadian to Jalandhar, then we will have to carry out two operations such as searching S1 for each part and then multiple updations for different occurrences of S1. But, if we wish to change the city of part P1 from Qadian to Jalandhar, then these problems will not occur because there

is only a single entry for part P I and the problem of inconsistency will not arise. So, we can say that update anomalies only exist for children not for parent because children may have multiple entries in the database.

Delete Operation: In hierarchical model, quantity information is incorporated into supplier record. Hence, the only way to delete a shipment (or supplied quantity) is to delete the corresponding supplier record. But such an action will lead to loss of information of the supplier, which is not desired. For example: Supplier S2 stops supplying 250 quantity of part PI, then the whole record of S2 has to be deleted under part PI which may lead to loss the information of supplier. Another

problem will arise if we wish to delete a part information and that part happens to be only part supplied by some supplier. In hierarchical model, deletion of parent causes the deletion of child records also and if the child occurrence is the only occurrence in the whole database, then the information of child records will also lost with the deletion of parent. For example: if we wish to delete the information of part P2 then we also lost the information of S3, S2 and S1 supplier. The information of S2 and Sl can be obtained from PI, but the information about supplier S3 is lost with the deletion of record for P2.

Record Retrieval: Record retrieval methods for hierarchical model are

complex and asymmetric which can be clarified with the following queries:

Query1: ***Find the supplier number for suppliers who supply part P2.***

Solution: In order to get this information, first we search the information of parent P2 from database, since parent occurs only once in the whole database, so we obtain only a single record for P2. Then, a loop is constructed to search all suppliers under this part and supplier numbers are printed for all suppliers.

Algorithm

get [next] part where PNO=P2;

do until no more shipments under this part;

get next supplier under this part;

print SNO;

end;

Query2: ***Find part numbers for parts supplied by supplier S2.***

Solution: In order to get required part number we have to search S2 under each part. If supplier S2, is found under a part then the corresponding part number is printed, otherwise we go to next part until all the parts are searched for supplier S2.

Algorithm

do until no more parts;

get next part;

get [next] supplier under this part where SNO=S2;

if found then print PNO;

end;

In above algorithms "next" is interpreted relative the current position (normally the row most recently accessed; for the initial case we assume it to be just prior to the first row of the table). We have placed square brackets around "next" in those statements where we expect at the most one occurrence to satisfy the specified conditions.

Since, both the queries involved different logic and are complex, so we can conclude that retrieval operation

of this model is complex and asymmetric.

Conclusion: As explained earlier, we can conclude that hierarchical model suffers from the Insertion anomalies, Update anomalies and Deletion anomalies, also the retrieval operation is complex and asymmetric, and thus hierarchical model is not suitable for all the cases.

Record

A collection of field or data items values that provide information on an entity. Each field has a certain data type such as integer, real or string. Records of the same type are group into record type.

Parent Child Relationship Type

It is 1:N relation between two record type. The record type 1 side is parent record type and one on the N side is

called child record type of the PCR type.

Advantages

1. Simplicity

Data naturally have hierarchical relationship in most of the practical situations. Therefore, it is easier to view data arranged in manner. This makes this type of database more suitable for the purpose.

2. Security

These database system can enforce varying degree of security feature unlike flat-file system.

3. Database Integrity

Because of its inherent parent-child structure, database integrity is highly promoted in these systems.

4.Efficiency:

The hierarchical database model is a very efficient, one when the database contains a large number of I: N relationships (one-to-many relationships) and when the users require large number of transactions, using data whose relationships are fixed.

Disadvantages

1. Complexity of Implementation:

The actual implementation of a hierarchical database depends on the physical storage of data. This makes the implementation complicated.

2. Difficulty in Management:

The movement of a data segment from one location to another cause all the accessing programs to be

modified making database management a complex affair.

3. Complexity of Programming:

Programming a hierarchical database is relatively complex because the programmers must know the physical path of the data items.

4. Poor Portability:

The database is not easily portable mainly because there is little or no standard existing for these types of database.

5. Database Management Problems:

If you make any changes in the database structure of a hierarchical database, then you need to make the necessary changes in all the application programs that access the database. Thus, maintaining the

database and the applications can become very difficult.

6. Lack of structural independence:

Structural independence exists when the changes to the database structure does not affect the DBMS's ability to access data. Hierarchical database systems use physical storage paths to navigate to the different data segments. So, the application programs should have a good knowledge of the relevant access paths to access the data. So, if the physical structure is changed the applications will also have to be modified. Thus, in a hierarchical database the benefits of data independence are limited by structural dependence.

7. Programs Complexity:

Due to the structural dependence and the navigational structure, the application programs and the end users must know precisely how the data is distributed physically in the database in order to access data. This requires knowledge of complex pointer systems, which is often beyond the grasp of ordinary users (users who have little or no programming knowledge).

8. Operational Anomalies:

As discussed earlier, hierarchical model suffers from the Insert anomalies, Update anomalies and Deletion anomalies, also the retrieval operation is complex and asymmetric, thus hierarchical model is not suitable for all the cases.

9. Implementation Limitation:

Many of the common relationships do not conform to the l:N format required by the hierarchical model. The many-to-many (N:N) relationships, which are more common in real life are very difficult to implement in a hierarchical model.

Network Model

The popularity of the network data model coincided with the popularity of the hierarchical data model. Some data were more naturally modeled with more than one parent per child. So, the network model permitted the modeling of many-to-many relationships in data.

In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model. The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type. A member record type can have that role in more than one set; hence the multiparent concept is supported.

An owner record type can also be a member or owner in another set. The

data model is a simple network, and link and intersection record types (called junction records by IDMS) may exist, as well as sets between them. Thus, the complete network of relationships is represented by several pairwise sets; in each set some (one) record type is owner (at the tail of the network arrow) and one or more record types are members (at the head of the relationship arrow).

Usually, a set defines a 1:M relationship, although 1:1 is permitted. The CODASYL network model is based on mathematical set theory.

Network model is a collection data in which records are physically linked through linked lists .*A DBMS is said to be a* Network DBMS if the relationships among *data in the database are of type many-to-*

many. The relationship among many-to-many appears in the form of a network.

Thus the structure of a network database is extremely complicated because of these many-to-many relationships in which one record can be used as a key of the entire database. A network database is structured in the form of a graph that is also a data structure.

Relational Model

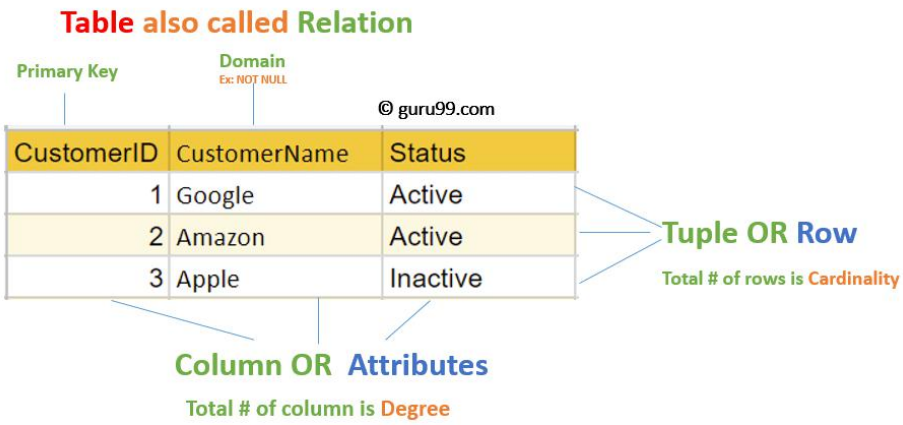
The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

Relational Model Concepts

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.

7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** - Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain



Relational Integrity constraints

Relational Integrity constraints is referred to conditions which must be present for a valid relation. These integrity constraints are derived from the rules in the mini-world that the database represents.

There are many types of integrity constraints. Constraints on the Relational database management system is mostly divided into three main categories are:

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

Domain Constraints

Domain constraints can be violated if an attribute value is not appearing in

the corresponding domain or it is not of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Example:

```
Create DOMAIN CustomerName

CHECK (value not NULL)
```

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

Key constraints

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

Example:

In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

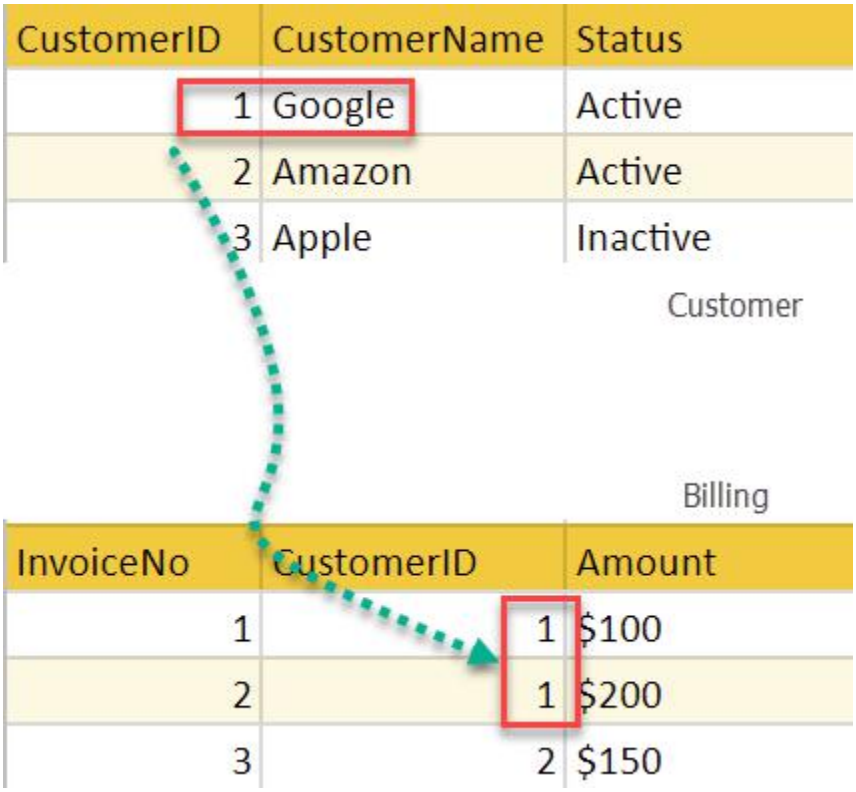
CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Referential integrity constraints

Referential integrity constraints is base on the concept of Foreign Keys. A foreign key is an important attribute

of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Example:



Operations in Relational Model

Four basic update operations performed on relational database model are

Insert, update, delete and select.

- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples.
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert Operation

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

In the above example, we have 2 relations, Customer and Billing.

Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount $300

Operations in Relational Model

Four basic update operations performed on relational database model are

Insert, update, delete and select.

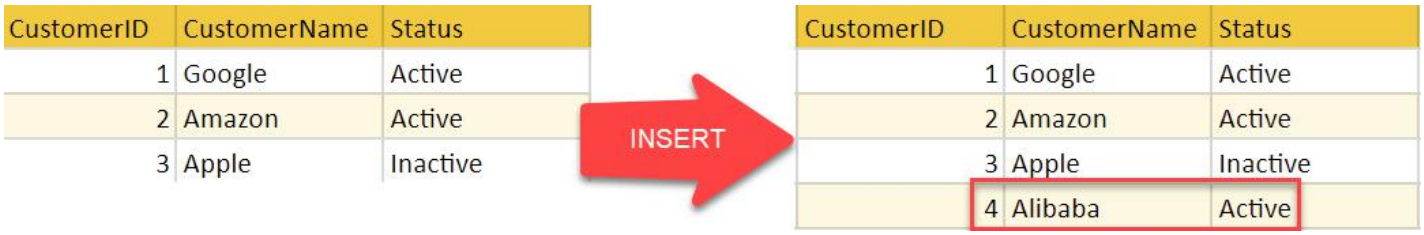
. Insert is used to insert data into the relation

- Delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples.
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert Operation

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.



Update Operation

You can see that in the below-given relation table CustomerName= 'Apple' is updated from Inactive to Active.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

Delete Operation

To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

In the above-given example, CustomerName= "Apple" is deleted from the table.

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign

keys from other tuples in the same database.

Select Operation

In the above-given example, CustomerName="Amazon" is selected

Best Practices for creating a Relational Model

- Data need to be represented as a collection of relations
- Each relation should be depicted clearly in the table
- Rows should contain data about instances of an entity
- Columns must contain data about attributes of the entity
- Cells of the table should hold a single value

- Each column should be given a unique name
- No two rows can be identical
- The values of an attribute should be from the same domain

Advantages

- **Simplicity**: A relational data model is simpler than the hierarchical and network model.
- **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use**: The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
- **Query capability**: It makes possible for a high-level query

language like SQL to avoid complex database navigation.

- **Data independence**: The structure of a database can be changed without having to change any application.
- **Scalable**: Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.

- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.