BCA Part II

Paper-XI: DBMS Using MS-ACCESS

Topic: DATA MODELS

Prepared by: Dr. Kiran Pandey
(School of Computer Science)

Email-id: kiranpandey.nou@gmail.com

Types of DATA MODELS

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system. The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.

Basically there are four database models. These are:

a. Hierarchical model

b. Network model

c. Relational model

(a) Hierarchical model:
This model presents data to users in a hierarchy of data elements that can be represented in a sort of inverted tree. In a sales order processing system, a customer may have many invoices raised to him and each invoice may have different data elements. Thus, the root level of data is customer, the second level is invoice and the last level is line items such as invoice number, date, product, quantity, etc.

This structure is quite natural when seen from the event point of view. However, the lower levels are owned by higher level data elements, and elements at the same level have no linkage at all. As a result, the query such as what products are purchased by which customer, in the above example, shall be difficult to carry out in the hierarchical structure.

The query as to which customer purchased which product would be convenient. Thus, where there are many-to-many relationships between two entities, this model would not be appropriate. Figure below shows the hierarchical model of data for a sales order processing application.

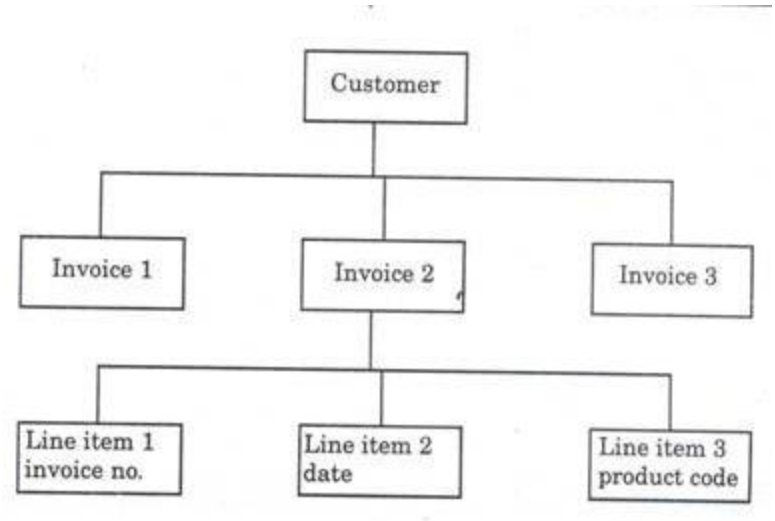

Figure 1: Hierarchical model for sales order processing

(b) Network model:

In the network model of database, there are no levels and a record can have any number of owners and also can have ownership of several records. Thus, the problem raised above in the sales order processing will not arise in the network model. As there is no definite path defined for retrieval of data, the number of links is very large and thus network databases are complex, slow and difficult to implement. In view of the difficulty in implementation, network model is used only when all other options are closed.

The typical example of a network database may be the employee and the department he/she has worked or can work with in future. Figure below shows the network model of data for an employee information system.

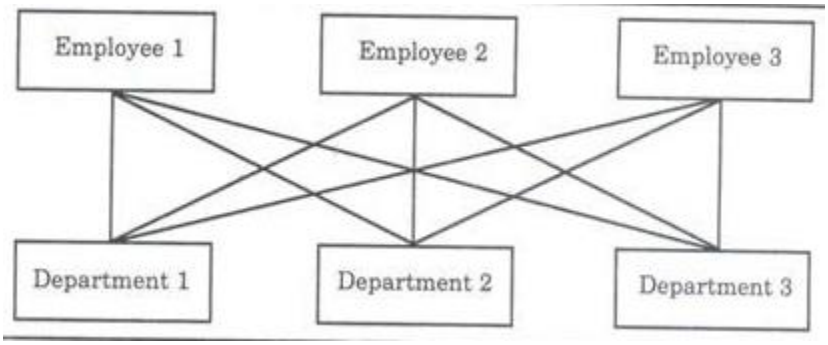

Figure 2: Network model of employee information system

(c) Relational model:

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model was developed to overcome the problems of complexity and inflexibility of the earlier two models in handling databases with many-to-many relationships between entities. These models are not only simple but also powerful. In the relational database, each file is perceived as a flat file (a two dimensional table) consisting of many lines (records), each record having key and non-key data item(s). The key item(s) is the data element(s) that identifies the record. This model is based on first-order predicate logic and defines a table as an **n-ary relation**.

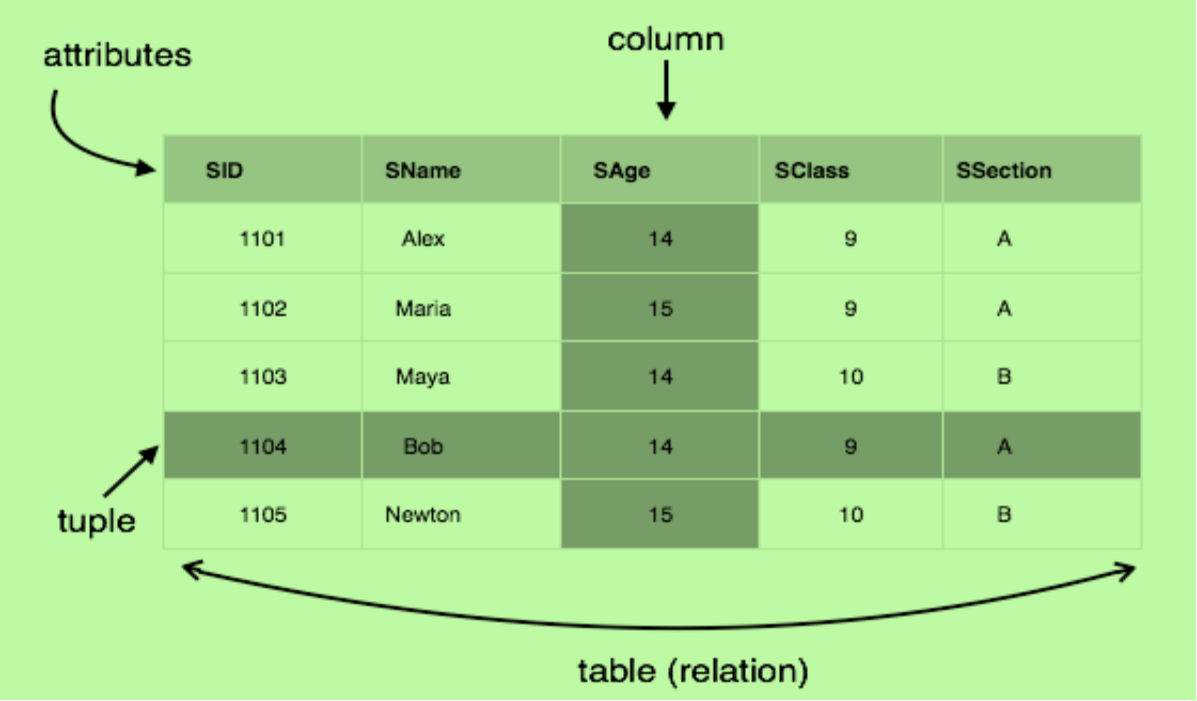

Figure 3: A relational model

The main highlights of this model are −

- Data is stored in tables called **relations**.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

The relational model draws greatly on the work of E.F. Codd who identifies features of a good relational database as following:

i) All information is logically represented as tables and the access of data are possible by the names of fields. Thus, the order, position or file linkage is not a matter of concern for users.

ii) The data dictionary has information regarding the database structure including the data type; size, etc., definitions, relationships and access permissions. The authorized users can learn about the database environment and change the environment using the data description language (DDL).

iii) A data manipulation language (DML) is available to users including programmers for creation, insertion, modification, retrieval, organizing and deletions of any part of the database. These manipulations are possible at the record level as well as for the whole file, giving the flexibility in defining access permissions for various categories of users.

iii) Any modification in the structure of database in terms of splitting the table horizontally or vertically should not have any impact on the logic of the program using the database. This data independence is the core advantage of the relational model of database.

iv) The distributed independence of data is another feature of a good relational database. The user programs do not require any change when data are first distributed or redistributed. The actual physical location of data does not matter to the user so long as that field appears in the data dictionary as local. The data redundancy can be avoided in this model. For this purpose, a process of data normalization is undertaken while designing the structure of a database.

Properties of relational model

Values Are Atomic
This property implies that columns in a relational table are not repeating group or arrays. Such tables are referred to as being in the "first normal form" (1NF). The atomic value property of relational tables is important because it is one of the cornerstones of the relational model.
The key benefit of the one value property is that it simplifies data manipulation logic.

Column Values Are of the Same Kind
In relational terms this means that all values in a column come from the same domain. A domain is a set of values which a column may have. For example, a Monthly Salary column contains only specific monthly salaries. It never contains other information such as comments, status flags, or even weekly salary.

This property simplifies data access because developers and users can be certain of the type of data contained in a given column. It also simplifies data validation. Because all values are from the same domain, the domain can be defined and enforced with the Data Definition Language (DDL) of the database software.

Each Row is Unique
This property ensures that no two rows in a relational table are identical; there is at least one column, or set of columns, the values of which uniquely identify each row in the table. Such columns are called primary keys. This property guarantees that every row in a relational table is meaningful and that a specific row can be identified by specifying the primary key value.

The Sequence of Columns is Insignificant
This property states that the ordering of the columns in the relational table has no meaning. Columns can be retrieved in any order and in various sequences. The benefit of this property is that it enables many users to share the same table without concern of how the table is organized. It also permits the physical structure of the database to change without affecting the relational tables.

The Sequence of Rows is Insignificant
This property is analogous the one above but applies to rows instead of columns. The main benefit is that the rows of a relational table can be retrieved in different order and sequences. Adding information to a relational table is simplified and does not affect existing queries.

Each Column Has a Unique Name
Because the sequence of columns is insignificant, columns must be referenced by name and not by position. In general, a column name need not be unique within an entire database but only within the table to which it belongs.

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

Basic concepts related to Relational model

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 represent 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 predefined value scope, known as attribute domain.

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:

- Key constraints
- Domain constraints
- Referential integrity constraints

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 cannot 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.

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: relational algebra and relational calculus.

Relational Algebra

Relational algebra is a 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 different**
- **Cartesian product**
- **Rename**

We will discuss all these operations in the following sections.

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)

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:

r ∪ s = { t | t ∈ r or t ∈ s}

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.

∏ 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**.

∏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:

r X s = { q t | q ∈ r and t ∈ s}

∏author = 'kiran_pandeyt'(Books X Articles)

Output: Yields a relation, which shows all the books and articles written by kiran_pandey

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**.

Additional operations are:

- **Set intersection**
- **Assignment**
- **Natural join**

COMPARISON OF HIERACHICAL, NETWORK AND RELATIONAL MODEL

Characteristic	Hierarchical model	Network model	Relational model
Data structure	✓One to many or one to one relationships ✓Based on parent child relationship	✓Allowed the network model to support many to many relationships ✓A record can have many parents as well as many children.	✓One to One, One to many, Many to many relationships ✓Based on relational data structures
Data manipulation	✓Does not provide an independent stand alone query interface ✓Retrieve algorithms are complex and asymmetric	✓Uses CODASYL (Conference on Data Systems Languages) ✓Retrieve algorithms are complex and symmetric	✓Relational databases are what brings many sources into a common query (such as SQL) ✓Retrieve algorithms are simple and symmetric
Data integrity	✓Cannot insert the information of a child who does not have any parent. ✓Multiple occurrences of child records which lead to problems of inconsistency during the update operation ✓Deletion of parent results in deletion of child records	✓Does not suffer form any insertion anomaly. ✓Free from update anomalies. ✓Free from delete anomalies	✓Does not suffer from any insert anomaly. ✓Free form update anomalies ✓Free from delete anomalies

CODD'S RULES FOR RELATIONAL MODEL

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database. These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following: data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.