# Unit 6 □ Data Models

*Structure*



## 6.0 Objectives

The objectives of the Unit are to :

- Understand early data management systems
- Understand database approach for management of data system
- Examine the concepts of data models
- Discuss the basics of normalization

## 6.1 Introduction

Data is an important resource for any organization and applications. A database can loosely be defined as a collection of data, which exists for providing information. The data together with the system, which manage it, termed a database system. It is important to differentiate a database from a database system. There are many types of database management system-IRS (information retrieval system) and bibliographic database system etc.

## 6.2 Evolution of Data Management Systems

### 6.2.1 Early Computer based Information System

The objective of any computer-based information handling system is the acquisition and processing of data so that it can easily be placed into meaningful context. The early information management programmes were developed independently without adequate consideration for integrated acquisition, processing and reporting. In ealrly phases of library automation, many institutes developed separate information processing systems for Acquisition, Cataloguing, Serials Management, and Current Awareness Services etc. Each separate application had its own master file, input data, and application programmes.

This lack of integration of different application programmes developed for management of large volume of information gave rise to various problems such as :

- Data required for one application may have already been supplied partly or fully for some other applications, such as input of author name in Acquistion system and Cataloguing System.
- Wastage of storage space by storing same data in multiple files for different application programmes
- Wastage of processing time
- Occurrence of inconsistencies and other errors in data files. While updating may take place in one file (Author name in Catalogue Master File), the same data stored elsewhere may fail to be updated (in Acquisition File/Author-Authority File), thereby causing data inconsistency.
- Independent development of information management programmes for different applications creates difficulties in integrating information.
- The application programmes were developed and data files created along functional boundaries. The unique key data items were not logically related and could not be used for cross-reference purposes during processing and retrieval of information. This seriously limited the information reporting capabilities and generation of meaningful output by the systems.

### 6.2.2 Database Management System

Database Management System plays an important role to avoid problems associated with early information management system and to readily achieve integration of data. Currently, database is the most important component of any computer based information system. A database may be defined as collections of inter related relevant data stored together to serve multiple applications. A database management system

(DBMS) is collection of software/programmes for processing the database. The data is stored in the database so that it is independent of the programmes using it.

### 6.2.2.2 Advantages

Some of the advantages that accrue from having an integrated/centralized database are :

- Redundancy can be reduced
- Inconsistency can be avoided/reduced
- Data can be shared
- Standard can be enforced
- Security can be enhanced
- Integrity can be maintained
- Conflicting requirements can be balanced

### 6.2.2.3 Bibliographic Database System

Bibliographic databases programmes combine many of the characteristics of structured database programmes with some of those associated with textual database programmes. In addition to sorting and selecting features, bibliographic database programmes offer features that are designed specifically to deal with the predominance of textual materials and rules for organizing citations/library catalogue that are associated with the academic and research environments.

- Data Entry forms.
- Field types for specific types of bibliographic information
- List fields or "Authority Lists"
- Note fields
- Word processor compatibility
- "Keys" in documents cite works in the database
- Emphasis on text in basic database functions
- Catalogue records formatting

Bibliographic database programmes have the ability to rearrange the information entered in the forms as citations. Citation formats the citation as you complete the form in a "Preview box" in the publishing style or format you have selected, and also allows you the option of generating formatted citations for all the works cited in a document (using "keys").

- Data Entry forms : Unlike standard database programmes, bibliographic database programmes provide predefined data entry forms. These "forms" are

specifically designed to hold information for different bibliographic types of source works available in the library. Rather than entering information in formatted form for a book, for instance, one will enter the basic components for a book, along with keywords and a brief summary, in a form :

| | |
|---|---|
| Author | Majumder, Arun Kumar<br>Bhattacharyya, Pritimoy |
| Title | Database management system |
| Edition | 1st |
| Place | New Delhi |
| Publisher | Tata McGraw-Hill |
| Year of Publication | 1996 |
| Pages | 475 |
| ISBN | 0-07-462239-0 |
| Keywords | DBMS, Database Management |

Bibliographic database offers a choice of forms that correspond to nearly every type of resource material available in the library. Unlike standard database programmes, bibliographic databases are able to use multiple forms in a single view of the database. And unlike the online database collections, records in a bibligraphic database can be edited to include keywords and notes or abstracts that are important.

- Field types for specific types of bibliographic information : In addition to predefined forms, bibliographic database programmes have field "types" that are peculiar to certain kinds of bibliographic data, such as names, titles, keywords and notes or abstracts.
  - Names, for instance : author, editor, translators, and other names of contributors, are entered with first and last names inverted, and the names of individuals separated with semicolons : Majumder, Arun Kumar. This allows bibliographic database programmes to convert names entered into the database into any of the variety of special formats required for names. The ability to reformat names, titles, pages, and journal names for differing publishing styles is an essential characteristic of bibliographic database programmes.

- List fields or "Authority Lists" : In several fields (Keywords, Author, Publisher, Geographic Place Name), index will automatically alphabetize the terms that have been entered in the database, and display the list for reference. One can use the list to enter names and terms consistently, or to search for matching items in the database.
- Note fields : In most bibliographic database programmes, every record has a "Note" or abstract field that expands as you type notes, and lets you use the keystrokes you are used to using in your word processor for entering paragraphs.
- Word processor compatibility : Most bibliographic database programmes assume that the information entered into the database will be processed using word processor, are thus designed with a tight integration to the popular word processing systems.
- "Keys" in documents cite works in the database.
- Emphasis on text in basic database functions : Database programmes allow you to manipulate, search and sort information in a variety of ways.
- Display formatting : Bibliographic database programmes have the ability to rearrange the information entered in the forms at the time of displaying.

- Field Length : It is extermely difficult to predit/fix the filed length in bibliographic database. A bibliographic database enables variable field length
- Repeatable Field : Multiple occurrence of data element in a field is very natural in case of bibliographic database. A document may have more than one author; subject index may assign more than one subject headings to a given document. It is also not an efficient solution that one will fix the maximum occurrence of a given field.
- Data retrieval : Searching, sorting, and subset selection enhancements emphasize the fact that bibliographic information is, primarily, textual data. So, for instance, the sorting feature allows you to alphabetize references with a "bibliographic sort"-ignoring leading articles in titles, and the selection feature allows you to retrieve a subset of records containing a particular name or keyword.
- Data output : Ability to output data from the fields and records in the database in different arrangements. The principle difference between bibliographic database programmes and traditional database programmes is in the ability to output or rearrange the information in bibliographic records.

## 6.3 Data Modeling

The goal of the data model is to make sure that all data objects required by the application functions are competely and accurately represented. Because the data model uses easily understood notations and natural language, it can be reviewed and verified as correct by the end-users. The data model is also detailed enough to be used by database developers as a 'blueprint' for building the physical database. The information contained in the data model will be used to define relational tables, primary and foreign keys, stored procedures, and triggers. A poorly designed database will require more time in the long run.

There are two categories models :

- **Implementation Model :** It is concerned with how the data are represented in the database. Provides concepts that can be understood by
  - End users
  - Computer specialist
  - Hides some details of data storage
  - Can be implemented on a computer system in a direct way
  - Used in current commercial DBMS
    - Relational
    - Network
    - Hierarchical
  - Represents data using record structure (record-based)
- **Conceptual Model :** It is concerned with what is represented in the database. It provides concepts closer to the way many users perceive data. Uses concepts such as entities, attributes, and relationships
  - Entity : a real world object or concepts (e.g. Project)
  - Attribute : properties that describes objects (e.g., Project_Name)
  - Relationships : an interaction or links among entities (e.g., works-on.)
    - Entity-Attribute-Value (EAV) data model
    - Object-oriented Data Model

### 6.3.1 Hierarchical Model

The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have

repeating information, generally in the child data segments. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses Parent Child Relationships. These are a 1 : N mapping between record types. For example, an organization might store information about an employee, such as name, employee number, department, salary. The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierachy, were the employee data represents the parent segment and the children data represents the child segment. If an employee has three children then there would be three child segments associated with one employee segment. In a hierarchical database the parent-child relationship is one to many. This restricts a child segment to having only one parent segment. Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.

**Advantages**

- It promotes data security
- It promotes data independence
- It promotes data integrity (parent/child relationship)
- Useful for large databases
- Useful when users require a lot of transactions which are fixed over time
- Suitable for large storage media

**Disadvantages**

- Requires knowledge of the physical level of data storage.
- Cannot handle the case where a part may belong to two or more components
- New relations or nodes result in complex system management tasks.
- Programmers must be familiar with the appropriate navigation
- Modification to data structure lead to significant modificantions to application programmes
- Does not provide the favoured ad-hoc query capability easily.
- No specific or precise standard

### 6.3.2 Network Model

The popularity of the network data model coincided with the popularity of the hierarchical data model. 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 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 multi-parent 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 pair wise 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.

**Advantages**

- Improves on hierarchical model
- An application can access an owner record and all the member records in the set.
- The movement from one owner to another is eased.
- Promotes data integrity because of the required owner-member relationship

**Disadvantages**

- Difficult to design and use properly
- The user and the programmer must be familiar with the data structure.
- Does not promote structural independence
- Navigational data access problems

### 6.3.3 Relational Model

RDBMS (relational database management system) is a database 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 in tables. A table is a collection of records and each record in a table contains the same fields. Properties of Relational Tables are :

- Values are Atomic
- Each Row is Unique
- Column Values are of the same kind

- The sequence of columns is insignificant
- The sequence of rows is insignificant
- Each column has a unique name

Certain fields may be designated as keys, which mean that searches for specific values of that field will use indexing to speed them up. Where fields in two different tables take values from the same set, a join operation can be performed to select related records in the two tables by matching values in those fields. Often, but not always, the fields will have the same name in both tables. For example, an "orders" table might contain (customer-ID, product-code) pairs and a "products" table might contain (product-code, price) pairs so to calculate a given customer's bill you would sum the prices of all products ordered by that customer by joining on the product-code fields of the two tables. This can be extended to joining multiple table on multiple fields. Because these relationships are only specified at retrieval time, relational databases are classed as dynamic database management system. The relational database model is based on the Relational Algebra.

**Advantages**

- Structural independence : i.e. can concentrate on the logical view.
- Data independence
- SQL capability

**Disadvantages**

- More hardware and operating system overhead : i.e. RDBMS may be slower.
- Ease of use can be a liability : i.e. possible misuse.

Dr. E.F. Codd provided 12 rules that define the basic characteristics of a relational database but implementation of these rules varies from vendor to vendor. In practice, many database products are considered 'relational' even if they do not strictly adhere to all 12 rules. A summary of Dr. Codd's 12 rules is presented below :

- Codd's Rule #**1. Data is presented in tables :** A set of related tables forms a database and all data is represented as tables. A ***table*** is a logical grouping of related data in tabular form (rows and columns)
  - Each ***row*** describes an item (person, place or thing) and each row contains information about a single item in the table
  - Each ***column*** describes a single characteristic about an item
  - Each ***value*** (datum) is defined by the intersection of a row and column
  - Data is ***atomic*****;** there is no more than one value associated with the intersection of a row and column

- There is ***no hierarchical ranking of tables***
- The relationships among tables are logical; there are ***no physical relationships among tables***

- Codd's Rule **#2. Data is logically accessible :** A relational database does not reference data by physical location; there is no such thing as the 'fifth row in the customers table' ***Each piece of data must be logically accessible by referencing1) a table; 2) a primary or unique key value; and 3) a column***
- Codd's Rule **#3. Nulls are treated uniformaly as unknown :** ***Null*** must always be interpreted as an ***unknown value.*** Null means no value has been entered; the value is not known. 'Unknown' is ***not*** the same thing as an empty string ("") or zero
- Codd's Rule **#4. Database is self-describing :** In addition to user data, a relational database contains data about itself. There are two types of tables in a RDBMS : ***user tables*** that contain the 'working' data and ***system tables*** contain data about the database structure.
- Codd's Rule **#5. A single language is used to communicate with the database management system :** There must be a single language that handles all communication with the database management system.
- Codd's Rule **#6. Provides alternatives for viewing data :** A relational database must not be limited to source tables when presenting data to the user. *Views* are *Virtual tables* or abstractions of the source tables. A view is an alternative way of looking at data from one or more tables. A view definition does not duplicate data; a view is not a copy of the data in the source tables. Onece created, a view can be manipulated in the same way as a source table.
- Codd's Rule **#7. Supports set-based or relational operations :** Rows are treated as ***sets*** for data manipulation operations (SELECT, INSERT, UPATE, DELETE).

  A relational database must support ***basic relational algebra operations*** (selection, projection; & join) and ***set operations*** (union, intersection, division, and difference).

  Set operations and relational algebra are used to operate on 'relations' (tables) to produce other relations.

  A database that supports only row-at-a-time (navigational) operations does not meet this requirement and is not considered 'relational'.

- Codd's Rule **#8. Physical data independence :** Applications that access data in a relational database must be unaffected by changes in the way the data is physically stored (i.e., the physical structure).

  An application that accesses data in a relational database contains only a basic definition of the data (data type and length); it does not need to know how the data is physically stored or accessed
- Codd's Rule **#9. Logical data independence :** Logical independence means the ***relationships among tables*** can change without impairing the function of applications and adhoc queries. The database schema or structure of tables and relationships (logical) can change without having to re-create the database or the applications that use it
- Codd's Rule **#10. Data integrity is a function of the DBMS :** In order to be considered relational, data integrity must be an internal function of the *DBMS*; not the *application programme*
- Codd's Rule **#11. Supports distributed operations :** Data in a relational database can be stored centrally or distributed. Users can join data from tables on different servers (distributed queries) and from other relational databases (heterogeneous queries). Data integrity must be maintained regardless of the number of copies of data and where it resides
- Codd's Rule **#12. Data integrity cannot be subverted :** The DBMS must prevent data from being modified by machine language intervention

### 6.3.4 Entity-Attribute-Value (EAV) data model

The best way to understand the rationale of EAV design is to understand row modeling (of which EAV is a generalized form). Consider a supermarket database that must manage thousands of products and brands, many of which have a transitory existence. Here, it is intuitively obvious that product names should not be hard-coded as names of columns in tables. Instead, one stores product descriptions in a Products table : purchases/sales of individual items are recorded in other tables as separate rows with a product ID referencing this table. Conceptually an EAV design involves a single table with three colums, an entity an attribute (such as species, which is actually a pointer into the metadata table) and a value for the attribute (e.g., rat). In EAV design, one row stores a single fact. In a conventional table that has one column per attribute, by contrast, one row stores a set of facts. EAV design is appropriate when the number of parameters that potentially apply to an entity is vastly more than those that actually apply to an individual entity.

### 6.3.5 Object-Oriented Model

Object DBMSs add database functionality to object programming languages. They bring much more than persistent storage of programming language objects. Object DBMSs extend the semantics of the C++, Smalltalk and Java object programming languages to provide full-featured database programming capability, while retaining native language compatibility. A major benefit of this approach is the unification of the application and database development into a seamless data model and language envorinment. As a result, applications require less code, use more natural data modeling, and code bases are easier to maintain. Object developers can write complete database applications with a modest amount of additional effort.

In contrast to a relational DBMS where a complex data structure must be flattened out to fit into tables or joined together from those tables to form the in-memory structure, object DBMSs have no performance overhead to store or retrieve a web or hierarchy of interrelated objects. This one-to-one mapping of object programming language objects to database objects has two benefits over other storage approaches : it provides higher performance management of objects, and it enables better management of the complex interrelationships between objects. This makes object DBMSs better suited to support applications such as financial portfolio, risk analysis systems, telecommunications service applications, World Wide Web document structures, design and manufacturing systems, and hospital patient record systems, which have complex relationships between data.

## 6.4 Database Normalization Basics

Normalization is often brushed aside as a luxury that only academics have time for. However, knowing the principles of normalization and applying them to your daily database design tasks really isn't all that complicated and it could drastically improve the performance of the DBMS. The concept of normalization and a brief overview of the most common normal forms have been presented.

Basically, normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process : eliminate redundant data (for example, storing the same data in more than one table) and ensure data dependencies (only storing related data in a table). Both of these reduce the amount of space a database consumes and ensure that data is logically stored.

The database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF)

through five (fifth normal form or 5NF). In practical applications, one will often see 1NF, 2NF, and 3NF along with the occasional 4NF. Fifth normal form is very rarely seen.

The normal forms, it's important to point out that they are guidelines and guidelines only. Occasionally, it becomes necessary to stray from them to meet practical business requirements. However, when variations take place, it's extremely important to evaluate any possible ramifications they could have on the system and account for possible inconsistencies.

1. First normal form (1NF) sets the very basic rules for an organized database :
   - Eliminate duplicate columns from the same table.
   - Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).
2. Second normal form (2NF) further addresses the concept of removing duplicate data :
   - Meet all the requirements of the first normal form.
   - Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
   - Create relationships between these new tables and their predecessors through the use of foreign keys.
3. Third normal form (3NF) goes one large step further :
   - Meet all the requirements of the second normal form.
   - Remove columns that are not dependent upon the primary key.
4. Finally, fourth normal form (4NF) has one additional requirement :
   - Meet all the requirements of the third normal form.
   - A relation is in 4NF if it has no multi-valued dependencies.
   - Remember, these normalization guidelines are cumulative. For a database to be in 2NF, it must first fulfill all the criteria of a 1NF database.

**References and Further Reading List**


1 2005 Data structure **(http://en. wikipedia. org/wiki/Data_structure).** sept 2005. Last visited :

2 2005 Chapple (Mike). Database normalization basics. (http:// databases.about.com/od/specificproducts/a/normalization.htm). Visited last : 22/10/2005

3 2001 Codd's 12 rules. **(http://www.itworld.com/nl/db_mgr/05072001/).** Visited last : 21/10/2005

4 2000 Database models. **(http://www.frick-cpa. com/ss7/Theory_Models. asp#File).** 2000. Visited last : 22/10/2005

5 1996 Majumder (Arun K) and Bhattacharyya (Pritimoy). Database management systems.. New Delhi : Tata McGraw-Hill, 1996.

6 1994 Elmasri (Ramez) and Navathe (Shamkant B). Fundamentals of database systems. California : Benjamin/Cummings, 1994.

7 1985 Data (CJ). An Introduction to database systems. 3rd ed. New Delhi : Narosa, 1985.

8 1983 Date (CJ). Database : a primer. Reading : Addison-Wesley, 1983.

## 6.5 Exercise

1. Describe evolution of data management systems.
2. Briefly describe different data models.
3. Discuss basics of database normalization.
4. Discuss 12 rules that define the basic characteristics of a relational database.