

Biyani's Think Tank

Concept based notes

Database Management System

(MCA)

Nandini Parwal

Information Technology

Biyani Institute of Science and Management

Jaipur



Published by :

Think Tanks
Biyani Group of Colleges

Concept & Copyright :

©**Biyani Shikshan Samiti**

Sector-3, Vidhyadhar Nagar,

Jaipur-302 023 (Rajasthan)

Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007

E-mail : acad@biyanicolleges.org

Website :www.gurukpo.com; www.biyanicolleges.org

ISBN :

Edition : 2011

Price :

While every effort is taken to avoid errors or omissions in this Publication, any mistake or omission that may have crept in is not intentional. It may be taken note of that neither the publisher nor the author will be responsible for any damage or loss of any kind arising to anyone in any manner on account of such errors and omissions.

Leaser Type Setted by :

Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the “Teach Yourself” style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

This book covers basic concepts related to the microbial understandings about diversity, structure, economic aspects, bacterial and viral reproduction etc.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director (Acad.)* Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

Syllabus

Overview of DBMS, Basic DBMS terminology, data base system v/s file system, data independence. Architecture of a DBMS

Introduction to data models: entity relationship model, hierarchical model: from network to hierarchical, relational model, comparison of network, hierarchical and relational models.

Data modeling using the Entity Relationship Model: ER model concepts, notation for ER diagram, mapping constraints, keys, Concepts of Super Key, candidate key, primary key, Generalization, aggregation, reduction of an ER diagrams to tables, extended ER model, relationships of higher degree.

Relational model: storage organizations for relations, relational algebra, relational calculus.

Normalization: Functional dependencies, normal forms, first, second, third normal forms, BCNF, inclusion dependencies, loss less join decompositions, normalization using FD, MVD, and JDs, alternative approaches to database design.

Introduction to SQL: Characteristics of SQL, Advantages of SQL, SQL data types and literals, Types of SQL commands, SQL operators and their procedure, Tables, views and indexes, Queries and sub queries, Aggregate functions, insert, update and delete operations, Joins, Unions, Intersection, Minus in SQL.

Contents		
S. No.	Name of Topic	Page No.
1.	Overview of DBMS 1.1 Basic DBMS Terminology 1.2 Data base system v/s file system 1.3 Advantages of DBMS 1.4 Data Independence	7-15
2.	Architecture of DBMS 2.1 Theory of Abstraction 2.2 Level of Architectures 2.3 Types of Users 2.4 Centralized and Distributed Databases	16-20
3.	Introduction to Data Models 3.1 Hierarchical Model 3.2 Relational Model 3.3 Network Model 3.4 Comparison of three models	21-29
4.	Entity Relationship Model 4.1 ER model concepts 4.2 Notation for ER diagram 4.3 Mapping Constraints 4.4 Concepts of Keys	30-52
5.	Normalization 5.1 Normal Forms 5.2 Functional dependencies 5.3 BCNF	53-58
6.	Relational Model and DBA 6.1 Relational Algebra 6.2 Relational Calculus 6.3 DBA	59-63
7.	Data and Query Processing	64-71

	7.1 Basic Retrieval Capacity 7.2 Query Language 7.3 Query Processing 7.4 Client/Server Design	
S. No.	Name of Topic	Page No.
8.	Structured Query Languages 8.1 Basic SQL & Keys 8.2 Nested Queries 8.3 QBEL & Quel 8.4 Aggregate Operators	72-96
9.	Advanced Features of SQL 9.1 Embedded SQL 9.2 Dynamic SQL 9.3 Cursors	97-102

--	--	--

Chapter 1

Overview of DBMS

Q.1 What do you mean by Data and Information?

Ans.: Data are plain facts. The word "data" is plural for "datum." When data are processed, organized, structured or presented in a given context so as to make them useful, they are called Information. It is not enough to have data (such as statistics on the economy). Data themselves are fairly useless, but when these data are interpreted and processed to determine its true meaning, they becomes useful and can be called Information.

Q.2 What do you mean by Database?

Ans.: Definitions of **Database** :

- An organized body of related information.
- In computing, a database can be defined as a structured collection of records or data that is stored in a computer so that a program can consult it to answer queries. The records retrieved in answer to queries become information that can be used to make decisions.

- An organized collection of records presented in a standardized format searched by computers. Web Pals, ID Weeks Library's Online Catalog, is a database. The periodical indexes available through the library are also databases.
- A collection of data organized for rapid search and retrieval by a computer.
- A collection of related data stored in one or more computerized files in a manner that can be accessed by users or computer programs via a database management system.
- An organized collection of information, data, or citations stored in electronic format that can be searched for specific information or records by techniques specific to each database.
- A logical collection of interrelated information, managed and stored as a unit, usually on some form of mass-storage system such as magnetic tape or disk.
- A database is a structured format for organizing and maintaining information that can be easily retrieved. A simple example of a database is a table or a spreadsheet.
- A database in an organized collection of computer records. The most common type of database consists of records describing articles in periodicals otherwise known as a periodical index.
- A database collects information into an electronic file, for example a list of customer addresses and associated orders. Each item is usually called a 'record' and the items can be sorted and accessed in many different ways.
- A set of related files that is created and managed by a Database Management System (DBMS).
- A computerized collection of information.
- Integrated data files organized and stored electronically in a uniform file structure that allows data elements to be manipulated, correlated, or extracted to satisfy diverse analytical and reporting needs.

- A collection of information stored in one central location. Many times, this is the source from which information is pulled to display products or information dynamically on a website.
- Relational data structure used to store, query, and retrieve information.
- An organized set of data or collection of files that can be used for a specified purpose. A collection of interrelated data stored so that it may be accessed with user friendly dialogs.
- A large amount of information stored in a computer system.

Q.3 What are the basic objectives of the Database?

Ans.: A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy. Quick, inexpensive, and flexible for the user. In data base design, several **specific objectives** are considered.

- (i) Controlled Redundancy
- (ii) Ease of Learning and Use
- (iii) Data Independence
- (iv) Most Information in Low Cost
- (v) Accuracy and Integrity
- (vi) Recovery from failure
- (vii) Privacy and Security
- (viii) Performance

Q.4 Name the elements of Database.

Ans.: Elements of Database as follows:

- (i) Data Collection
- (ii) Direct Access Storage Device
- (iii) Data Dictionary

- (iv) Database Administrator
- (v) Database Management System

Q.5 Define the Database Management System.

Ans.: (i) A **Database Management System** (DBMS), or simply a **Database System** (DBS), consists of :

- A collection of interrelated and persistent data (usually referred to as the **Database** (DB)).
 - A set of application programs used to access, update and manage that data (which form the data Management System (MS)).
- (ii) The goal of a DBMS is to provide an environment that is both **convenient** and **efficient** to use in :
- Retrieving information from the database.
 - Storing information into the database.
- (iii) Databases are usually designed to manage **large** bodies of information. This involves :
- Definition of structures for information storage (data modeling).
 - Provision of mechanisms for the manipulation of information (file and systems structure, query processing).
 - Providing for the safety of information in the database (crash recovery and security).
 - Concurrency control if the system is shared by users.

Q.6 Why do we need Database Management System? Describe in the term of development.

Ans.: There are four basic components of Database Management System :

- (i) **Data** : Raw facts which we want to feed in the computer.
- (ii) **Hardware** : On which the data to be processed.

- (iii) **Software** : The interface between the hardware and user, by which the data will change into the information.
- (iv) **User** : There are so many types of users some of them are application programmer, endcase users and DBA.

Q.7 What is the basic purpose of Database Management System?

Ans.: Purpose of Database Systems :

- (i) To see why database management systems are necessary, let's look at a typical "File-Processing System" supported by a conventional operating system.

The application is a savings bank :

- Savings account and customer records are kept in permanent system files.
- Application programs are written to manipulate files to perform the following **tasks** :
 - Debit or credit an account.
 - Add a new account.
 - Find an account balance.
 - Generate monthly statements.

- (ii) Development of the System proceeds as follows :

- New application programs must be written as the need arises.
- New permanent files are created as required.
- but over a long period of time files may be in different formats, and
- Application programs may be in different languages.

- (iii) So we can see there are problems with the Straight File-Processing Approach :

- **Data Redundancy and Inconsistency :**
 - Same information may be duplicated in several places.
 - All copies may not be updated properly.
- **Difficulty in Accessing Data :**
 - May have to write a new application program to satisfy an unusual request.
 - E.g. find all customers with the same postal code.
 - Could generate this data manually, but a long job.
- **Data Isolation :**
 - Data in different files.
 - Data in different formats.
 - Difficult to write new application programs.
- **Multiple Users :**
 - Want concurrency for faster response time.
 - Need protection for concurrent updates.
 - E.g. two customers withdrawing funds from the same account at the same time - account has \$500 in it, and they withdraw \$100 and \$50. The result could be \$350, \$400 or \$450 if no protection.
- **Security Problems :**
 - Every user of the system should be able to access only the data they are permitted to see.
 - E.g. payroll people only handle employee records, and cannot see customer accounts; tellers only access account data and cannot see payroll data.
 - Difficult to enforce this with application programs.
- **Integrity Problems :**
 - Data may be required to satisfy constraints.
 - E.g. no account balance below \$25.00.

- Again, difficult to enforce or to change constraints with the file-processing approach.

These problems and others led to the development of **Database Management Systems**.

Q.8 What are the advantages and disadvantages of DBMS over Conventional File System?

Ans.: Advantages :

- An organized and comprehensiveness of recording the result of the firms activities.
- A receiver of data to be used in meeting the information requirement of the MIS users.
- Reduced data redundancy.
- Reduced updating errors and increased consistency.
- Greater data integrity and independence from applications programs.
- Improved data access to users through use of host and query languages.
- Improved data security.
- Reduced data entry, storage, and retrieval costs.
- Facilitated development of new applications program.
- Standard can be enforced: Standardized stored data format is particularly desirable as an old data to interchange or migration (change) between the system.
- Conflicting requirement can be handled.

Disadvantages :

- It increases opportunity for person or groups outside the organization to gain access to information about the firms operation.
- It increases opportunity for fully training person within the organization to misuse the data resources intentionally.
- The data approach is a costly due to higher H/W and S/W requirements.
- Database systems are complex (due to data independence), difficult, and time-consuming to design.

- It is not maintain for all organizations .It is only efficient for particularly large organizations.
- Damage to database affects virtually all applications programs.
- Extensive conversion costs in moving form a file-based system to a database system.
- Initial training required for all programmers and users.

Q.9 Why would you choose a database system instead of simply storing data in operating system files? When would it make sense not to use a database system?

Ans: A database is an integrated collection of data, usually so large that it has to be stored on secondary storage devices such as disks or tapes. This data can be maintained as a collection of operating system files, or stored in a DBMS (database management system). The advantages of using a DBMS are:

Data independence and efficient access. Database application programs are independent of the details of data representation and storage. The conceptual and external schemas provide independence from physical storage decisions and logical design decisions respectively. In addition, a DBMS provides efficient storage and retrieval mechanisms, including support for very large files, index structures and query optimization.

Reduced application development time. Since the DBMS provides several important functions required by applications, such as concurrency control and crash recovery, high level query facilities, etc., only application-specific code needs to be written. Even this is facilitated by suites of application development tools available from vendors for many database management systems.

Data integrity and security. The view mechanism and the authorization facilities of a DBMS provide a powerful access control mechanism. Further, updates to the data that violate the semantics of the data can be detected and rejected by the DBMS if users specify the appropriate integrity constraints.

Data administration. By providing a common umbrella for a large collection of data that is shared by several users, a DBMS facilitates maintenance and data

administration tasks. A good DBA can effectively shield end-users from the chores of fine-tuning the data representation, periodic back-ups etc.

Concurrent access and crash recovery. A DBMS supports the notion of a transaction, which is conceptually a single user's sequential program. Users can write transactions as if their programs were running in isolation against the database.

The DBMS executes the actions of transactions in an interleaved fashion to obtain good performance, but schedules them in such a way as to ensure that conflicting operations are not permitted to proceed concurrently. Further, the DBMS maintains a continuous log of the changes to the data, and if there is a system crash, it can restore the database to a transaction-consistent state. That is, the actions of incomplete transactions are undone, so that the database state reflects only the actions of completed transactions. Thus, if each complete transaction, executing alone, maintains the consistency criteria, then the database state after recovery from a crash is consistent.

If these advantages are not important for the application at hand, using a collection of files may be a better solution because of the increased cost and overhead of purchasing and maintaining a DBMS.

Q.10 What do you mean by flat file database?

Ans: It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

Q.11 What is "transparent DBMS"?

Ans: It is one, which keeps its Physical Structure hidden from user.

Q.12 What is durability in DBMS?

Ans: Once the DBMS informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

Q.13 What is Data Independence?

Ans: Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

1. **Physical Data Independence:** Modification in physical level should not affect the logical level.
2. **Logical Data Independence:** Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

Q.14 Explain the difference between logical and physical data independence.

Ans: Logical data independence means that users are shielded from changes in the logical structure of the data, while physical data independence insulates users from changes in the physical storage of the data. We saw an example of logical data independence in the answer to Exercise 1.2. Consider the Students relation from that example (and now assume that it is not replaced by the two smaller relations). We could choose to store Students tuples in a heap file, with a clustered index on the sname field. Alternatively, we could choose to store it with an index on the gpa field, or to create indexes on both fields, or to store it as a file sorted by gpa. These storage alternatives are not visible to users, except in terms of improved performance, since they simply see a relation as a set of tuples. This is what is meant by physical data independence.

Q.15 Does the relational model, as seen by an SQL query writer, provide physical and logical data independence? Explain.

Ans: The user of SQL has no idea how the data is physically represented in the machine. He or she relies entirely on the relation abstraction for querying. Physical data independence is therefore assured. Since a user can define views, logical data independence can also be achieved by using view definitions to hide changes in the conceptual schema.

Chapter 2

Architecture of DBMS

Q.1 Describe the three levels of data abstraction?

Ans: There are three levels of abstraction:

1. **Physical level:** The lowest level of abstraction describes how data are stored.
2. **Logical level:** The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
3. **View level:** The highest level of abstraction describes only part of entire database.

Q.2 Define the Overall System Structure of Database Management System.

Ans.: Overall System Structure :

- a) Database systems are partitioned into modules for different functions. Some functions (e.g. file systems) may be provided by the operating system.
- b) **Components include :**

- **File Manager** manages allocation of disk space and data structures used to represent information on disk.
- **Database Manager** : The interface between low-level data and application programs and queries.
- **Query Processor** translates statements in a query language into low-level instructions the database manager understands. (May also attempt to find an equivalent but more efficient form.)
- **DML Precompiler** converts DML statements embedded in an application program to normal procedure calls in a host language. The precompiler interacts with the query processor.
- **DDL Compiler** converts DDL statements to a set of tables containing metadata stored in a data dictionary.

In addition, several data structures are required for physical system implementation :

- **Data Files** : store the database itself.
- **Data Dictionary** : stores information about the structure of the database. It is used **heavily**. Great emphasis should be placed on developing a good design and efficient implementation of the dictionary.
- **Indices** : provide fast access to data items holding particular values.

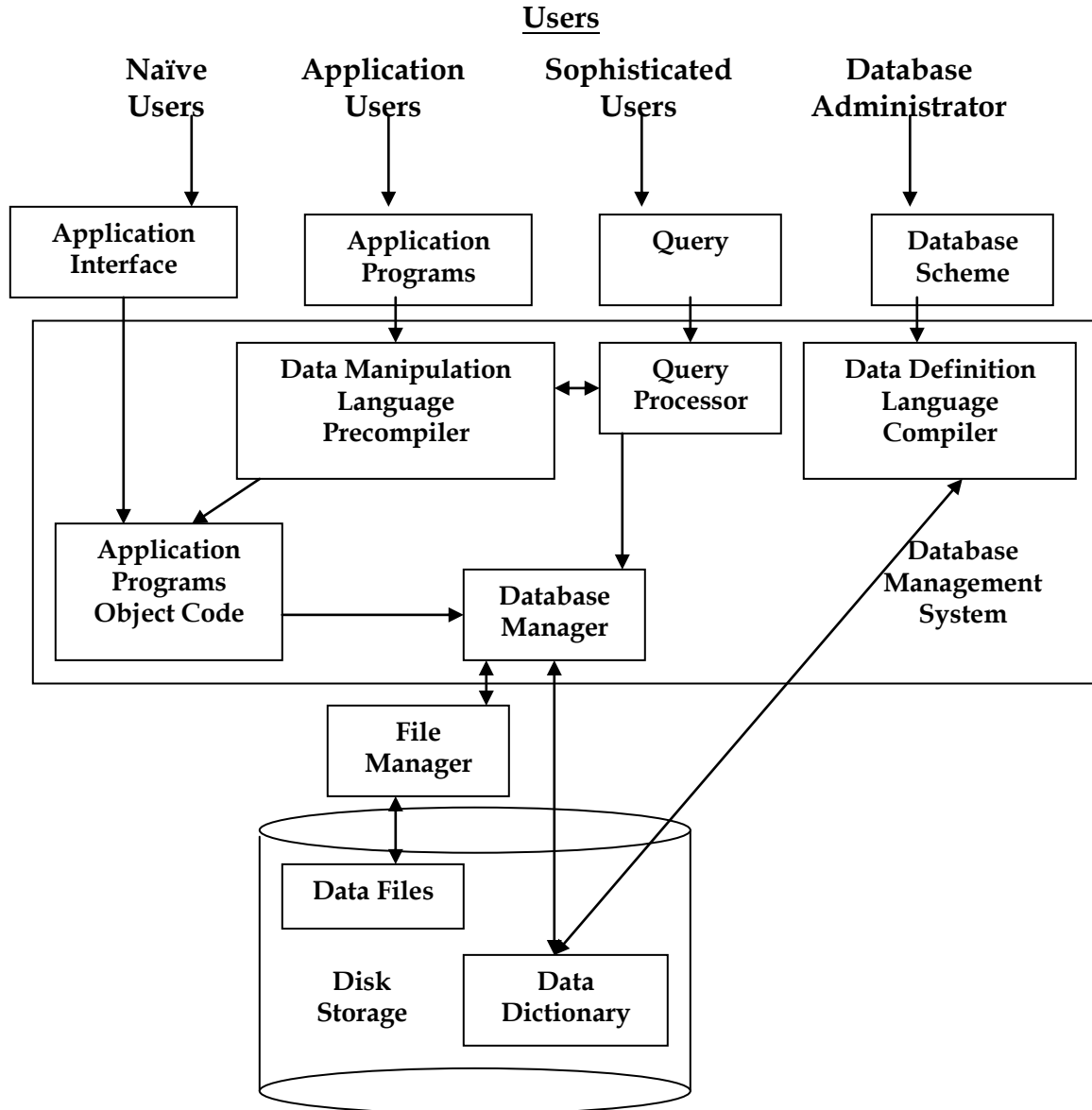


Figure : Database System Structure

Chapter 3

Introduction to Data

Q.1 Explain Hierarchical, Network and Relational, Database Models

Ans: Database models continue to evolve as the information management needs of organizations become more complex. From flat files to **relational databases**, the growing demands on data integrity, reliability and performance of database management systems (DBMS), has shaped the design of databases and their underlying models. In this document, three database models are discussed comparing and contrasting their major features. The three models in order of discussion are Hierarchical, Network and Relational database models.

Hierarchical Database Model

The hierarchical model is the oldest of the three models discussed here. This model is an improvement of the flat-file database system since it employs a simple data relationship scheme. The relationships in the hierarchical model are child/parent relationships. The name "hierarchical" is derived from one major restriction on the child/parent relationships, that is, although a parent entity can have several child entities, a child entity can only have one and only one parent. For this reason all the relationships form a hierarchy that traces back to one root. In fact, this model is often visualized as an upside down tree, where the entity at the top is seen as a root and as such all other entities sprout from the root.

As shown in the diagram, one major problem with the hierarchical model is the increased risk of data inconsistency. In the case above, a separate "Customers" table must exist for every product line due to the fact that a child entity cannot have more than one parent. However, there is a great chance that there are many customers who purchased more than one type of product. Consequently, information about those customers must exist in more than one

table causing data redundancy. Another problem with the hierarchical model is the inflexibility of the model. For example, in the diagram above, the database is restricted to three product lines. Adding a new product line would require redesigning the database since all the relationships must be predefined. Finally, another problem with the hierarchical model is in the child/parent relationship restriction. Every child must have a parent. Therefore, in the example above, it is impossible to add new customers who have not bought any new products yet. To overcome some of the limitations of the hierarchical model, the network model was created.

Network Database Model

The network model is an improvement to its predecessor, the hierarchical model. In the network model, a child entity can have more than one parent. Therefore, the design can be visualized as several inverted trees interconnected by branches as opposed to the single inverted tree characteristic of the hierarchical model

Q.2 Difference between hierarchical data model or network data mode and relational data model.

Ans: **1.The hierarchical data 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. Data in a series of records, which have a set of field values attached to it. 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. This is done by using trees, like set theory used in the relational model, "borrowed" from maths.

2: The network data 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.

3: The relational model:

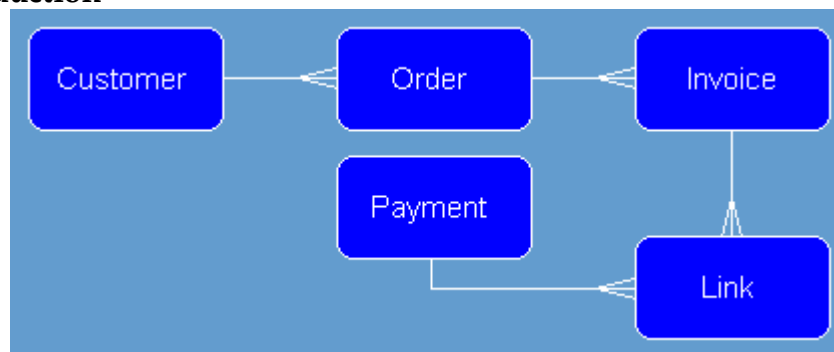
A database based on the relational model developed by E.F. Codd. A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organized in tables. A table is a collection of records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

Properties of relational database:

1. Values Are Atomic
2. Each Row is Unique
3. Column Values Are of the Same Kind
4. The Sequence of Columns is Insignificant
5. The Sequence of Rows is Insignificant
5. Each Column Has a Unique Name

Q.3 Explain ER Diagrams and Notations.

Ans:

- Introduction

Entity relationship diagramming is a technique that is widely used in the world of business and information technology to show how information is, or should be, stored and used within a business system.

The success of any organization relies on the efficient flow and processing of information.

In this example information flows around the various departments within the organization. This information can take many forms, for example it could be written, oral or electronic.

Here is an **example** of the sort of information flows that you might be analyzing:

The general manager regularly communicates with staff in the sales and marketing and accounts departments by using e-mail. Orders received by sales and marketing are forwarded to the production and accounts departments, for fulfillment and invoicing. The accounts department forward regular written reports to the general manager, they also raise invoices and send these to the customers.

Data modeling is a technique aimed at optimizing the way that information is stored and used within an organization. It begins with the identification of the main data groups, for example the invoice, and continues by defining the detailed content of each of these groups. This results in structured definitions for all of the information that is stored and used within a given system.

The technique provides a solid foundation for systems design and a universal standard for system documentation. Data modeling is an essential precursor to analysis & design, maintenance & documentation and improving the performance of an existing system.

Entity Relationship Diagram - Diagram Notation

Entity relationship diagramming uses a standard set of symbols to represent each of these defined data groups and then proceeds by establishing the relationships between them. The first of these symbols is the soft-box entity symbol.



An entity is something about which data will be stored within the system under consideration. In this example the data group invoice can be identified as a system entity.

The other main component on a data model is the relationship line. A Relationship is an association between two entities to which all of the occurrences of those entities must conform.



The relationship is represented by a line that joins the two entities, to which it refers. This line represents two reciprocal relationships: That of the first entity with respect to the second, and that of the second entity with respect to the first.

Entity relationship diagramming is all about identifying entities and their relationships and then drawing a diagram that accurately depicts the system. This applies equally to the design of a new system or the analysis of an existing one.

The end result of entity relationship diagramming should be a clear picture of how information is stored and related within a proposed, or existing, system.

Entity Relationship Diagram - Entities

Here, we illustrate the concept of an entity, which can be applied to almost anything that is significant to the system being studied. Some examples of information systems and their entities are listed below:

Banking system: Customer, Account, Loan.

Airline system: Aircraft, Passenger, Flight, Airport.

An entity is represented by a box containing the name of that entity. A precise definition of 'entity' is not really possible, as they even vary in nature.

For example, in the airline system, whilst an aircraft is a physical object (entities often are) a flight is an event and an airport is a location. However entities are nearly always those things about which data will be stored within the system under investigation.

Note that entities are always named in the singular; for example: customer, account and loan, and not customers, accounts and loans.

This course uses symbols that are standard in the IT industry. This uses the soft- box symbol shown to represent an entity. If a site uses a different symbol set, this is not a problem, as entity relationship diagramming techniques are the same regardless of the symbols being used.

Entity Relationship Diagram - Entity Types & Occurrence

Similar entity occurrences are grouped together and collectively termed an entity type. It is entity types that are identified and drawn on the data model. An entity occurrence identifies a specific resource, event, location, notion or (more typically) physical object.

In this course the term 'entity' is, by default, referring to entity type. The term entity occurrence will be specifically used where that is relevant. Each entity has a data group associated with it. The elements of the data group are referred to as the 'attributes' of the entity. The distinction between what is an attribute of an entity and what is an entity in its own right is often unclear. This is illustrated shortly.

Entity Relationship Diagram - Entity Naming

Entity names are normally single words and the name chosen should be one familiar to the users. The entity name can include a qualifier in order to clarify their meaning. However, if different names are currently used to describe a given entity in different areas of the organization then a new one should be chosen that is original, unique and meaningful to all of the users.

For example, the terms 'signed contract', 'sale' and 'agreement' might be recreated as the entity 'completed'.

Conversely an organization may be using a 'catch all' term to describe what the analyst identifies as being a number of separate entities. For example the term 'invoice' may be being used to describe 3 invoice types - each of which is, in fact, processed in a different manner.

In this case prefixing the entity names with qualifiers, is likely to be the best solution. **Notification**

The process of identifying entities is one of the most important steps in developing a data model.

It is common practice for an experienced analyst to adopt an intuitive approach to entity identification, in order to produce a shortlist of potential entities. The viability of each of these potential entities can then be considered using a set of entity identification guidelines. This should result in some of the potential entities being confirmed as entities, whilst others will be rejected.

Q.4 Explain the following terms briefly:

Attribute, Domain, Entity, Relationship, Entity set, Relationship set, one-to-many relationship, many-to-many relationship, participation constraint, Overlap constraint, Covering constraint, Weak Entity Set, Aggregation, and Role Indicator.

Ans: Term explanations:

Attribute - a property or description of an entity. A toy department employee entity could have attributes describing the employee's name, salary, and years of service.

Domain - a set of possible values for an attribute.

Entity - an object in the real world that is distinguishable from other objects such as the green dragon toy.

Relationship - an association among two or more entities.

Entity set - a collection of similar entities such as all of the toys in the toy department.

Relationship set - a collection of similar relationships

One-to-many relationship - a key constraint that indicates that one entity can be associated with many of another entity. An example of a one-to-many relationship is when an employee can work for only one department, and a department can have many employees.

Many-to-many relationship - a key constraint that indicates that many of one entity can be associated with many of another entity. An example of a many-to-many relationship is employees and their hobbies: a person can have many different hobbies, and many people can have the same hobby.

Participation constraint - a participation constraint determines whether relationships must involve certain entities. An example is if every department entity has a manager entity. Participation constraints can either be total or partial. A total participation constraint says that every department has a manager. A partial participation constraint says that every employee does not have to be a manager.

Overlap constraint - within an ISA hierarchy, an overlap constraint determines whether or not two subclasses can contain the same entity.

Covering constraint - within an ISA hierarchy, a covering constraint determines where the entities in the subclasses collectively include all entities in the superclass.

For example, with an Employees entity set with subclasses HourlyEmployee and SalaryEmployee, does every Employee entity necessarily have to be within either HourlyEmployee or Salary Employee?

Weak entity set - an entity that cannot be identified uniquely without considering some primary key attributes of another identifying owner entity. An example is including Dependent information for employees for insurance purposes.

Aggregation - a feature of the entity relationship model that allows a relationship set to participate in another relationship set. This is indicated on an ER diagram by drawing a dashed box around the aggregation.

Role indicator - If an entity set plays more than one role, role indicators describe the different purpose in the relationship. An example is a single Employee entity set with a relation Reports-To that relates supervisors and subordinates.

Q.5 What is Data Model?

Ans: A collection of conceptual tools for describing data, data relationships data semantics and constraints.

Q.6 What is E-R model?

Ans: This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

Q.7 What is Object Oriented model?

Ans: This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

Q.8 What is an Entity?

Ans: It is a 'thing' in the real world with an independent existence.

Q.9 What is an Entity type?

Ans: It is a collection (set) of entities that have same attributes.

Q.10 What is an Entity set?

Ans: It is a collection of all entities of particular entity type in the database.

Q.11 What is an Extension of entity type?

Ans: The collections of entities of a particular entity type are grouped together into an entity set.

Q.12 What is Weak Entity set?

Ans: An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

Q.13 What is an attribute?

Ans: It is a particular property, which describes the entity.

Chapter 4

Entity Relationship Model

Q.1 What are partial, alternate,, artificial, compound and natural key?

Ans:

1. **Partial Key:** It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.
2. **Alternate Key:** All Candidate Keys excluding the Primary Key are known as Alternate Keys.
3. **Artificial Key:** If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.
4. **Compound Key:** If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.
5. **Natural Key:** When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

Q.2 What's the difference between a primary key and a unique key?

Ans: Both primary key and unique enforce uniqueness of the column on which they are defined. But by default primary key creates a clustered index on the column, where unique creates a nonclustered index by default. Another major difference is that, primary key doesn't allow NULLs, but unique key allows one NULL only.

Q.3 What are constraints? Explain different types of constraints.

Ans: Constraints enable the RDBMS enforce the integrity of the database automatically, without needing you to create triggers, rule or defaults. Types of constraints: NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY. For an explanation of these constraints see books online for the pages titled: "Constraints" and "CREATE TABLE", "ALTER TABLE"

Chapter 5

Normalization

Q.1 What is normalization?

Ans: It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties (1).Minimizing redundancy, (2). Minimizing insertion, deletion and update anomalies.

Q.2 Explain Normalization.

Ans: Well a relational database is basically composed of tables that contain related data. So the Process of organizing this data into tables is actually referred to as normalization.

Q.3 What is Functional Dependency?

Ans: A Functional dependency is denoted by $X \rightarrow Y$ between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R . The constraint is for any two tuples t_1 and t_2 in r if $t_1[X] = t_2[X]$ then they have $t_1[Y] = t_2[Y]$. This means the value of X component of a tuple uniquely determines the value of component Y .

Q.4 What is 1 NF (Normal Form)?

Ans: The domain of attribute must include only atomic (simple, indivisible) values.

Q.5 What is Fully Functional dependency?

Ans: It is based on concept of full functional dependency. A functional dependency $X \rightarrow Y$ is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

Q.6 What is 2NF?

Ans: A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

Q.7 What is 3NF?

Ans: A relation schema R is in 3NF if it is in 2NF and for every FD $X \rightarrow A$ either of the following is true

1. X is a Super-key of R .
2. A is a prime attribute of R .

In other words, if every non prime attribute is non-transitively dependent on primary key.

Q.8 What is BCNF (Boyce-Codd Normal Form)?

Ans: A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD $X \rightarrow A$, X must be a candidate key.

Q.9 What is 4NF?

Ans: A relation schema R is said to be in 4NF if for every Multivalued dependency X Y that holds over R, one of following is true.

- 1.) X is subset or equal to (or) $XY = R$.
- 2.) X is a super key.

Q.10 What is 5NF?

Ans: A Relation schema R is said to be 5NF if for every join dependency $\{R_1, R_2, \dots, R_n\}$ that holds R, one the following is true

- 1.) $R_i = R$ for some i.
- 2.) The join dependency is implied by the set of FD, over R in which the left side is key of R.

Q.11 What is Domain-Key Normal Form?

Ans: A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

Q.12 What is denormalization and when would you go for it?

Ans: As the name indicates, denormalization is the reverse process of normalization. It's the controlled introduction of redundancy in to the database design. It helps improve the query performance as the number of joins could be reduced.

Send your requisition at
info@biyanicolleges.org