

B. E.
Fourth Semester Examination, Dec-2007
DATA BASE MANAGEMENT SYSTEM

Note : Attempt only five questions:

Q. 1. What do you mean by DBMS ? Give advantages of DBMS over file Processing System.

Ans. A collection of programs that enables you to store, modify, and extract information from a database. There are many different types of DBMSs, ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications:

- Computerized library systems
- Automated teller machines
- Flight reservation systems
- Computerized parts inventory systems

Advantages of using DBMS :

In DBMS, all files are integrated into one system thus reducing redundancies and making data management more efficient. In addition, DBMS provides centralized control are:

1. Redundancies and inconsistencies can be reduced :

In conventional data systems, an organisation often builds a collection of application programs often created by different programmers and requiring different components of the operation data of the organisation. The data in conventional data systems is often not centralised. Some application may require data to be combined from several systems. These several systems could well have data that is redundant as well as inconsistent (that is different copies of the same data may have different values). Data inconsistencies are often encountered in everyday life. For example, we have all come roll situations when a new address is communicated to an organisation that we deal with (e.g. a bank, or Telecom, or a gas company), we find that some of the communications from that organisation are received at the new address while others involve reduction in redundancy as well as inconsistency. It also is likely to reduce the costs for collection, storage and updation of data.

2. Better service to the users :

A DBMS is often used to provide better service to the users. In conventional systems, availability of information is often poor since it normally is difficult to obtain information that the existing systems were not designed for. Once several conventional systems are combined to form one centralised data base, the availability of information and its up-to-dateness is likely to improve since the data can now be shared and the DBMS makes it easy to respond to unforeseen information requests. Centralizing the data in a database also

often means that users can obtain now and combined information that would have been impossible to obtain otherwise. Also, use of a DBMS should allow users that do not know programming to interact with the data more easily.

3. Flexibility of the system is improved :

Changes are often necessary to the contents of data stored in any system. These changes are more easily made in a database than in a conventional system in that these changes do not need to have any impact on application programs.

4. Cost of developing and maintaining systems is lower :

As noted earlier, it is much easier to respond to unforeseen requests when the data is centralized in a database than when it is stored in conventional file systems. Although the initial cost of setting up a database can be large, one normally expects the overall cost of setting up a database and developing and maintaining application programs to be lower than for similar service using convoluting and maintaining application programs to be programmers can be substantially higher in using non-procedural languages that have been developed with modern DBMS than using procedural languages.

5. Standards can be enforced :

Since all access to the database must be through the DBMS, standards are easier to enforce. Standards may relate to the naming of the data, the format of the data, the format of the data, the structure of the data etc.

6. Security can be improved :

In conventional systems, applications are developed in an ad hoc manner. Often different systems of an organization would access different components of the operational data. In such an environment, enforcing security can be quite difficult. Setting up a database makes it easier to enforce security restrictions since the data is now centralized. It is easier to control who has access to what parts of the database. However, setting up a database can also make it easier for a determined person to breach security. We will discuss this in the next section.

7. Integrity can be improved :

Since the data of the organization using a database approach is centralized and would be used by a number of users at a time, it is essential to enforce integrity controls. If a number of users are allowed to update the same data item at the same time, there is a possibility that the result of the updates is not quite what was intended. For example, in an airline BBMS we could have a situation where the number of bookings made is larger than the capacity of the aircraft that is to be used for the flight. Controls therefore must be introduced to prevent such errors to occur because of concurrent updating activities. However, since all data is stored only once, it is often easier to maintain integrity than in conventional systems.

8. Enterprise requirements can be identified :

All enterprises have sections and departments and each of these units often consider the work of their unit as the most important and therefore consider their needs as the most important. Once a database has been

set up with centralised control, it will be necessary to identify enterprise requirements and to balance the needs of completion units. It may become necessary to ignore some requests for information if they conflict with higher priority needs of the enterprise.

9. Data model must be developed :

Perhaps the most important advantage of setting up a database system is the requirement that an overall data model for the enterprise be built. In conventional systems, it is more likely that files will be designed as needs of particular applications demand. The overall view is often not considered. Building an overall view of the enterprise data, although often an expensive exercise, is usually very cost-effective in the long term.

Q. 2. What do you understand by the term E-R diagram ? Sketch the E-R diagram of Railway reservation system and then reduce this diagram into tables.

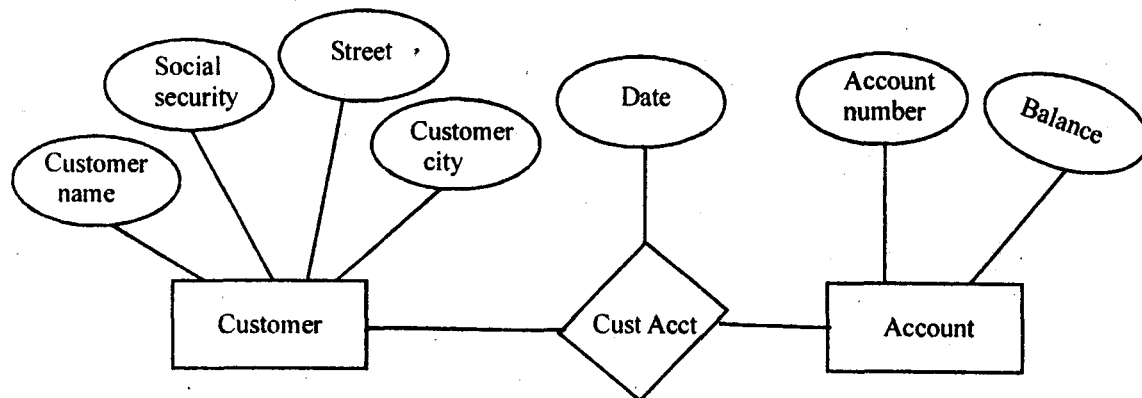
Ans. ER Diagram :

We can express the overall logical structure of a database graphically with an E-R diagram.

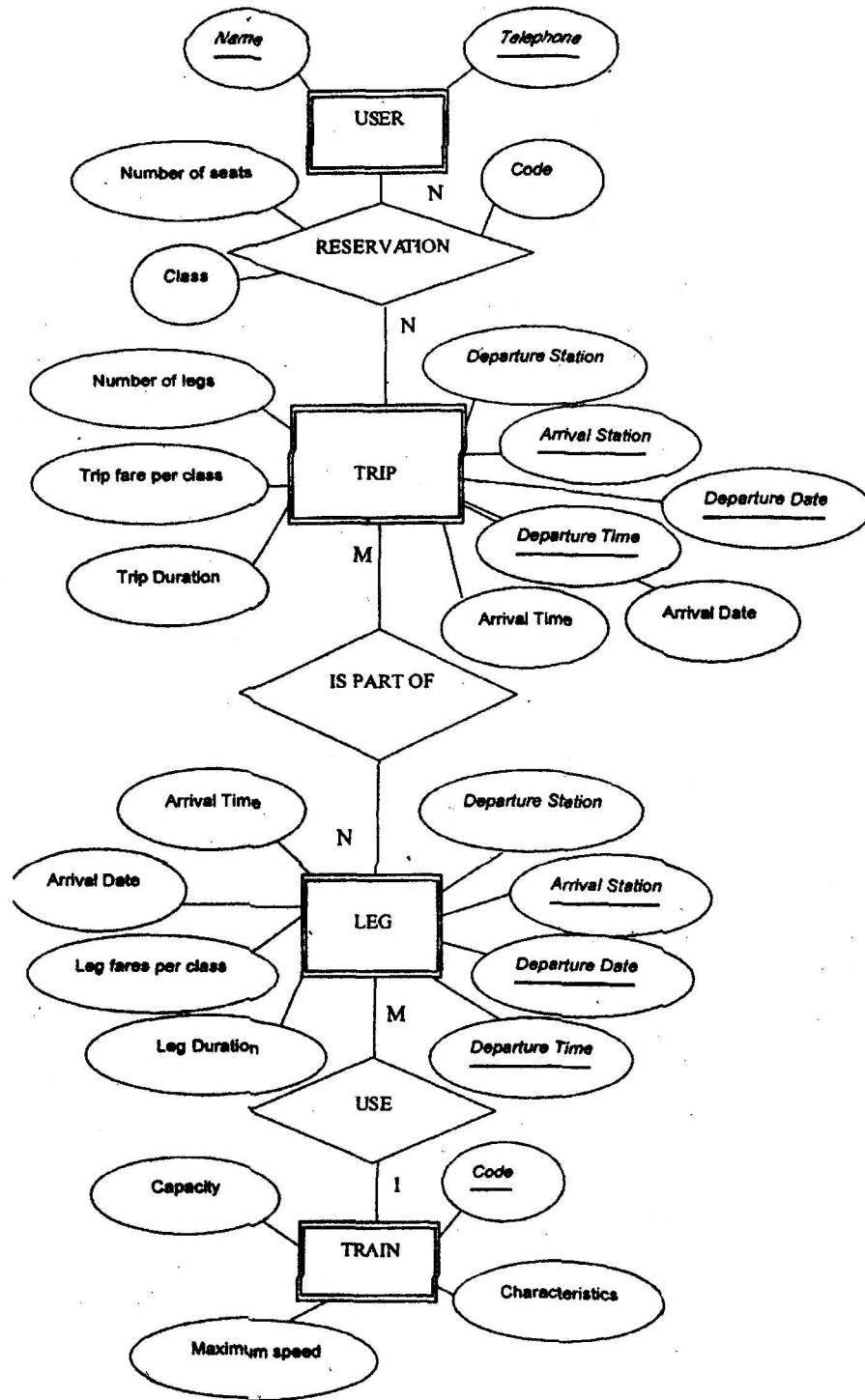
Its components are :

- Rectangles representing entity sets.
- Ellipses representing attributes.
- Diamonds representing relationship sets.
- Lines linking attributes to entity sets and entity sets to relationship sets.

In the text, lines may be directed (have an arrow on the end) to signify mapping cardinalities for relationship sets.



ER diagram for Railway Reservation system



Q. 3. What do you mean by File Organization ? Describe various ways to organize a file.

Ans. File organization :

1. A file is organized logically as sequence of records.
2. Records are mapped onto disk blocks.
3. Files are provided as a basic construct in operating systems, so we assume the existence of an underlying file system.
4. Blocks are of a fixed size determined by the operating system.
5. Record sizes vary.
6. In relational database, tuples of distinct relations may be of different sizes.
7. One approach to mapping database to files is to store records of one length in a given file.
8. An alternative is to structure files to accommodate variable-length records. (fixed-length is easier to implement.)

Organization of records in file :

There are several ways of organizing records in files.

- Heap file organization. Any record can be placed anywhere in the file where there is space for the record. There is no ordering of records.
- Sequential file organization. Records are stored in sequential order, based on the value of the search key of each record.
- Hashing file organization. A hash function is computed on some attribute of each record. The result of the function specifies in which block of the file the record should be placed—to be discussed in chapter 11 since it is closely related to the indexing structure.
- Clustering file organization. Records of several different relations can be stored in the same file. Related records of the different relations are stored on the same block so that one I/O operation fetches related records from all the relations.

Sequential File organization :

1. A sequential file is designed for efficient processing of records in sorted order on some search key.
 - Records are chained together by pointers to permit fast retrieval in search key order.
 - Pointer points to next record in order.
 - Records are stored physically in search key order (or as close to this as possible).
 - This minimizes number of block accesses.
 - Figure 10. 15 shows an example, with name as the search key.
2. It is difficult to maintain physical sequential order as records are inserted and deleted.
 - Deletion can be managed with the pointer chains.
 - Insertion poses problems if no space where new record should go.

- If space, use it, else put new record in an overflow block.
 - Adjust pointers accordingly.
 - Figure 10. 16 shows the previous example after an insertion.
 - Problem: we now have some records out of physical sequential order.
 - If very few records in overflow blocks, this will work well.
 - If order is lost, reorganize the file.
 - Reorganizations are expensive and done when system load is low.
3. If insertions rarely occur, we could keep the file in physically sorted order and reorganize when insertion occurs. In this case, the pointer fields are no longer required

Clustering File Organization :

1. One relation per file, with fixed-length record, is good for small databases, which also reduces the code size.
2. Many large-scale DB systems do not rely directly on the underlying operating system for file management. One large OS file is allocated to DB system and all relations are stored in one file.
3. To efficiently execute queries involving **depositor customer** one may store the depositor tuple for each name near the customer tuple for the corresponding name, as shown in Figure 10.19.
4. This structure misses together tuples from two relations, but allows for efficient processing of the join.
5. If the customer has many accounts which cannot fit in one block, the remaining records appear on nearby blocks. This file structure, called clustering, allows us to read many of the required records using one block read.
6. Our use of clustering enhances the processing of a particular join but may result in slow processing of other types of queries, such as selection on customer.

For example, the query

aaaaaaaaaaaa – select * from customer

Now requires more block accesses as our customer relation is now interspersed with the deposit relation.

7. Thus it is a trade-off, depending on the types of query that the database designer believes to be most frequent. Careful use of clustering may produce significant performance gain.

Q. 4. What do you mean by Functional dependencies and Normalization ? Explain various Kinds of Normalization.

Ans. Functional Dependency

1. Functional dependencies :

- Functional dependencies are a constraint on the set of legal relations in a database.
- They allow us to express facts about the real world we are modeling.
- The notion generalizes the idea of a superkey.

- Let $\alpha R \beta \subseteq R$
 - Then the functional dependency $\alpha \rightarrow \beta$ holds on R if in any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, it is also the case that $t_1[\beta] = t_2[\beta]$
 - Using this notation, we say k is a superkey of R if $k \rightarrow R$
 - In other words, k is a superkey of R if, whenever $t_1[k] = t_2[k]$ then $t_1[R] = t_2[R]$ (and thus $t_1 = t_2$)
2. Functional dependencies allow us to express constraints that cannot be expressed with superkeys.
 3. Consider the scheme
 4. Loan-info-schema=(bname,loan#, cname, amount)
 5. If a loan may be made jointly to several people (e.g. husband and wife), then we would not expect loan # to be a superkey. That is, there is no such dependency
loan# – cname
- we do however expect the functional dependency
- loan# – amount
- loan# – bname
- to hold, as a loan number can only be associated with one amount and one branch.
6. A set F of functional dependencies can be used in two ways:
 - To specify constraints on the set of legal relations. (Does F hold on R ?)
 - To test relations to see if they are legal under a given set of functional dependencies. (Does r satisfy F ?)
 7. Figure shows a relation r that we can examine.
 8. We can see that $A \rightarrow C$ is satisfied (in this particular relation), but $C \rightarrow A$ is not.
 $AB \rightarrow C$ is also satisfied.
 9. Functional dependencies are called **trivial** if they are satisfied by all relations.
 10. In general, a functional dependency $A \rightarrow B$ is trivial if BCA
 11. In the customers relation of figure 5.4, we see that **Street \rightarrow City** is satisfied by this relation. However, as in the real world two cities can have streets with the same names (e.g. Main, Broadway, etc.), we would not include this functional dependency in our list meant to hold on Customer-Scheme.
 12. The list of functional dependencies for the example database is :
 - On Branch –scheme:
 - bname–assets

- bname-assets
- On Customer-scheme:
- cname-city
- cname-street
- On loan-scheme:
- loan# -amount
- loan #-bname
- On Account-scheme:
- account#-balance
- account#- bname

There are no functional dependencies for Borrower-schema, nor for Depositor-schema.

Normalization : Normalisation is the process of taking data from a problem and reducing it to a set of relations while ensuring data integrity and eliminating data redundancy

- Data integrity – all of the data in the database are consistent, and satisfy all integrity constraints.

- Data redundancy– if data in the database can be found in two different locations (direct redundancy) or if data can be calculated from other data items (indirect redundancy) then the data is said to contain redundancy.

Data should only be stored once and avoid storing data that can be calculated from other data already held in the database. During the process of normalisation redundancy must be removed, but not at the expense of breaking data integrity rules. If redundancy exists in the database then problems can arise when the database is in normal operation:

- When data is inserted the data must be duplicated correctly in all places where there is redundancy. For instance, if two tables exists for in a database, and both tables contain the employee name, then creating a new employee entry requires that both tables be updated with the employee name.

- When data is modified in the database, if the data being changed has redundancy, then all versions of the redundant data must be updated simultaneously, So in the employee example a change to the employee name must happen in both tables simultaneously.

The removal of redundancy helps to prevent insertion, deletion, and update errors, since the data is only available in one attribute of one table in the database, The data in the data base can be considered to be in one of a number of 'normal forms'. Basically the normal form of the data indicated how much redundancy is in that data. The normal forms have a strict ordering :

1. 1st Normal Form
2. 2nd Normal Form
3. 3rd Normal Form
4. BCNF

There are other normal forms, such as 4th and 5th normal forms. They are rarely utilised in system design and are not considered further here. To be in a particular form requires that the data meets the criteria to also be in all normal forms before that form. Thus to be in 2nd normal form the data must meet the criteria for both 2nd normal form and 1st normal form. The higher the form the more redundancy has been eliminated.

First Normal Form :

- First normal form (1NF) deals with the 'shape' of the record type.
- A relation is in 1NF if, and only if, it contains no repeating attributes or groups of attributes.
- Example:
- The Student table with the relating group is not in 1NF
- It has repeating groups, and it is called an 'unnormalised table'.

Relational databases require that each row only has a single value per attribute, and so a repeating group in a row is not allowed. To remove the repeating group, one of two things can be done:

- Either flatten the table and extend the key, or
- Decompose the relation—leading to First Normal Form

Second Normal Form :

Second normal form (or 2NF) is a more stringent normal form defined as:

A relation is in 2NF if, and only if, it is in 1NF and every non-key attribute is fully functionally dependent on the whole key. Thus the relation is in 1NF with no repeating groups, and all non-key attributes must depend on the whole key, not just some part of it. Another way of saying this is that there must be no partial key dependencies (PKDs). The problems arise when there is a compound key, e.g. the key to the Record relation—metric—no, subject. In this case it is possible for non-key attributes to depend on only part of the key—i.e. On only one of the two key attributes. This is what 2NF tries to prevent.

Third Normal Form : 3NF is an even stricter normal form and removes virtually all the redundant data :

- A relation is in 3NF if, it is in 2NF and there are no transitive functional dependencies
- Transitive functional dependencies arise:
- when one non-key attribute is functionally dependent on another non-key attribute:
- FD: non-key attribute → non-key attribute
- and when there is redundancy in the database

By definition transitive functional dependency can only occur if there is more than one non-key field, so we can say that a relation in 2NF with zero or one non-key field must automatically be in 3NF.

Q. 5. Explain :

(a) **Parallel Database,**

(b) **Data Mining and Data Warehousing.**

Ans. (a) Parallel Database: Parallelism is natural to DBMS processing

- pipeline parallelism: many machines each doing one step in a multi-step process.

For more study material Log on to <http://www.ululu.in/>

- partition parallelism: many machines doing the same thing to different pieces of data.
- **Both are natural in DBMS!**

DBMSs are the most successful application of parallelism.

- Teradata, Tandem vs. Thinking Machines, KSR.

Every major DBMS vendor has some || server

Workstation manufacturers now depend on || DB server sales.

- Reasons for success:
- Bulk-processing (=partition ||-ism).
- Natural pipelining.

Inexpensive hardware can do the trick!

- Users/app-programmers don't need to think in ||
- Different Types of DBMS ||-ism
- Intra-operator parallelism
- get all machines working to compute a given operation (scan, sort, join)
- Inter-operator parallelism
- each operator may run concurrently on a different site (exploits pipelining)
- *Inter-query parallelism*
- different queries run on different sites

Ans. (b) Data Mining and Data warehousing: Data mining is the exploration and analysis of large quantities of data in order to discover valid, novel, potentially useful, and ultimately understandable patterns in data.

Valid: The patterns hold in general.

Novel: We did not know the pattern beforehand.

Useful; We can devise actions from the patterns.

Understandable: We can interpret and comprehend the patterns.

The Knowledge Discovery Process Steps:

1. Identify business problem
2. Data mining
3. Action
4. Evaluation and measurement
5. Deployment and integration into business processes

Data Mining Step in Detail :

1 Data preprocessing :

- Data selection: Identify target data sets and relevant fields – Data cleaning
- Data cleaning
- Remove noise and outliers
- Data transformation
- Create common units
- Generate new fields

2 Data mining model construction

3 Model evaluation

Data Warehousing :

- Integrated data spanning long time periods, often augmented with summary information.
- Several gigabytes to terabytes common.
- Interactive response times expected for complex queries; ad-hoc updates uncommon. Warehousing issues.
- Semantic Integration: When getting data from multiple sources, must eliminate mismatches, e.g., different currencies, schemes.
- Heterogeneous Sources: Must access data from a variety of source formats and repositories.
- Replication capabilities can be exploited here.
- Load, Refresh, Purge: Must load data, periodically refresh it, and purge too-old data.
- Metadata Management: Must keep track of source, loading time, and other information of all data in the warehouse.

Q. 6. Employee (FNAME, MINT, LNAME, SSN, BDATE, ADDRERSS, SEX, SOLAR, SUPERS, DNO)

DEPT-LOCATION (DNUMBER, DLOCATION)

DEPARTMENT (DNAME, DNUMBER, MGRSSN, MGRSTARTDATE)

WORKS-NO (ESSN, RNO, HOURS)

PROJECT (PNAME, PNUMBER, PLOCATION, DNUM)

DEPENDENT (ESSN, DEPENDENT=NAME, SEX, BDATE, RELATIONSHIP)

Consider the COMPANY relational database schema, where the primary keys are underlined (given above). Construct the following SQL queries:

(a) Retrieve the name and address of all employees who work for the 'Research' department.

Ans. Select e.name, e.address from employees e inner join department d

Where d.dname = "Research "

Q. 6. (b) For sever project located in 'Stafford', list the project No, the controlling department–number, and the department manager's last name, address and birthday.

Ans. Select P.project, D.Dnumber, E.Landname, F.address, E.BDate from project P
 Inner join Department D
 Where D.Dlocation = "stafford"
 Innerjoin employee E.

Q. 6. (c) Find the name of employee who work on all the projects controlled by department no 5 ?

Ans. Select E.Fname from employee E
 join Department D
 Where D.Dnumber = 5
 join Project P
 Where P.Pnumber! = null.

Q. 6. (d) List the names of managers who have at least one dependent ?

Ans. Select E.Fname from employee E
 join Dependent D
 Where E.SSN = D.Department.

Q. 6. (e) Retrieves the names of employees who have no dependent ?

Ans. Select E.Fname from employee E
 Join Department D
 Where E.SSN! = D.Department

Q. 7. (a) What do you mean by Relational model constraints ? Explain it with it's various kinds.

Ans. Relational model: The **relational model** for database management is a database model based on predicate logic and set theory. The aims that included avoiding, without loss of completeness, the need to write computer programs to express database queries and enforce database integrity constraints. "Relation" is a mathematical term for "table", and thus "relational" roughly means "based on tables". It does not refer to the links or "keys" between tables, contrary to popular belief. Data are operated upon by means of a relational calculus or relational algebra, these being equivalent in expressive power. The relational model of data permits the database designer to create a consistent, logical representation of information. Consistency is achieved by including declared constraints in the database design, which is usually referred to as the logical schema; The theory includes a process of database normalization whereby a design with certain desirable properties can be selected form a set of logically equivalent alternatives. The access plans and other implementation and operation details are handled by the DBMS engine, and are not reflected in the logical model. This contrasts

with common practice for SQL DBMSs in which performance tuning often requires changes to the logical model.

Relational model constraints: Functional dependency is the relational model constraint Functional Dependency

- $x \rightarrow A$ is an assertion about relation R that whenever two tuples of R agree on all the attributes of X, then they must also agree on the attribute A.
- say " $x \rightarrow A$ holds in R."
- Convention: X, Y, Z represent sets of attributes; A, B, C, ... represent single attributes.
- convention: no set formers in sets of attributes, just ABC, rather than {A, B, C}.

Drinkers(name, addr, beersLiked, manf, favBeer)

- Reasonable FD's to assert:

1. $\text{name} \rightarrow \text{addr}$
2. $\text{name} \rightarrow \text{favBeer}$
3. $\text{beersLiked} \rightarrow \text{manf}$

Functional dependency with multiple attributes

- No need for FD's with > 1 attribute on right.
- But sometimes convenient to combine FD's as a shorthand.
- Example: $\text{name} \rightarrow \text{addr}$ and $\text{name} \rightarrow \text{favBeer}$ become
 $\text{name} \rightarrow \text{addr favBeer}$
- $>$ attribute on left may be essential.

Example: $\text{bar beer} \rightarrow \text{price}$

Q. 7. (b) Give the basic features of Network model and Hierarchical model.

Ans. 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. Detain 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 modes 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. 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 hierarchy, where 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.

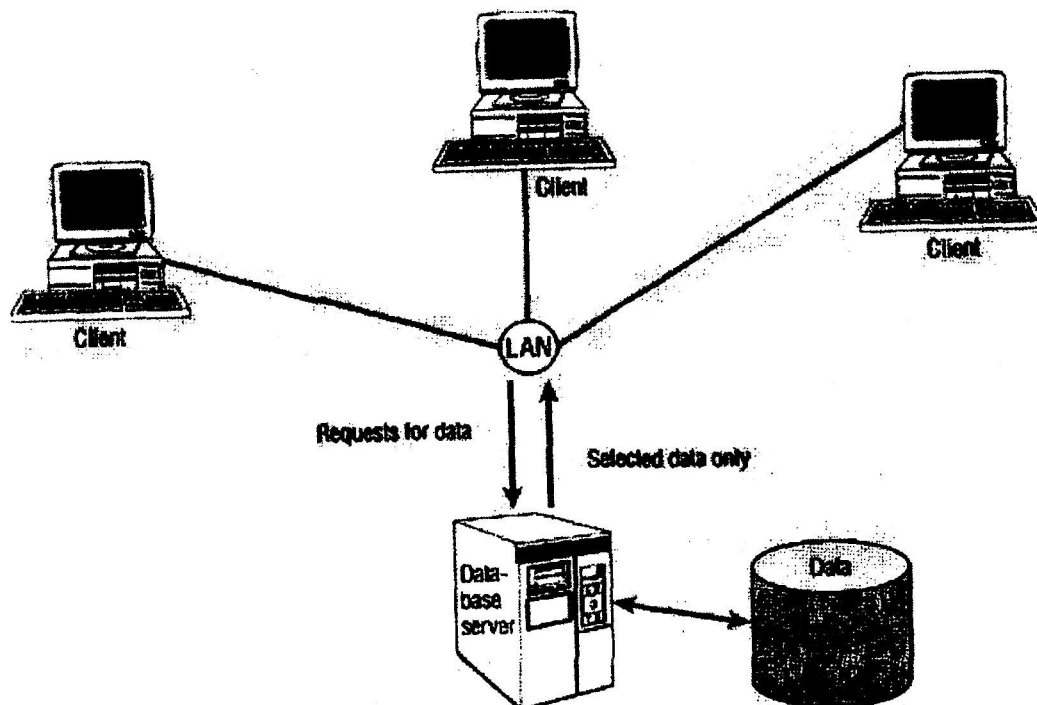
Network Model:

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 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 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.

Q. 8. Write short notes on :

- (a) Client/server architecture,**
- (b) Responsibility of Database Administrator.**
- (c) Primary Key, Candidate Key, Superkey.**

Ans. (a) Client/Server architecture: Client-Server Architecture :

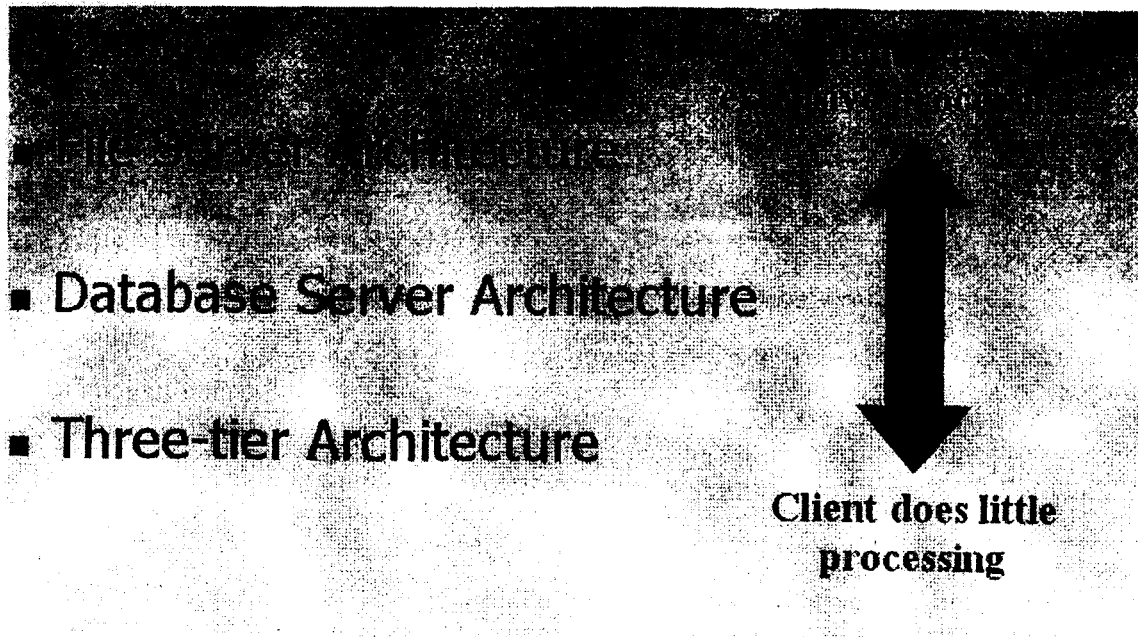


The term client/server was first used in the 1980s in reference to personal computers (PCs) on a network. The actual client/server model started gaining acceptance in the late 1980s. The client/servers software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interoperability, and scalability as compared to centralized, mainframe, time sharing computing.

A client is defined as a requester of services and a server is defined as the provider of services. A single machine can be both a client and a server depending on the software configuration. For details on client/server software architectures. This technology description provides a summary of some common client/servers architectures and, for completeness, also summarizes mainframe and file sharing architectures. Detailed descriptions for many of the individual architectures are provided elsewhere in the document.

System has following properties :

- Networked computing model
- Processes distributed between clients and servers
- Client–Workstation (usually a PC) that requests and uses a service
- Servers–Computer (PC/mini/mainframe) that provides a service
- For DBMS, server is a database server



(b) Responsibility of DBA: Some of the more advanced duties of the Oracle DBA might include the following:

– Data analysis :

The DBA will frequently be called on to analyze the data stored in the database and to make recommendations relating to performance and efficiency of that data storage. This might relate to the more effective use of indexes or the use of some feature such as the Parallel Query option.

– Database design (preliminary) :

The DBA is often involved at the preliminary database-design stages. Through the involvement of the DBA, many problems that might occur can be eliminated. The DBA knows the DBMS and system, can point out potential problems, and can help the development team with special performance considerations.

– Data coding and optimization—By modeling the data, it is possible to optimize the system layout to take the most advantage of your I/O subsystem.

– Assistance developers with SQL and stored procedure development—The DBA should be prepared to be a resource for developers and users. The DBA is often called on to help with SQL problems as well as to design and write stored procedures.

– Enterprise standards and naming conventions—Because many different groups might perform different roles in developing and deploying applications, it is often the DBA who is called on to help define enterprise standards and naming conventions as well as to ensure that new applications are conforming to these standards.

– Development of production migration procedures—Because the DBA is responsible for the availability and reliability of the DBMS and applications using that DBMS software. This involves evaluating new software or patches as well as testing them. It is up to the DBA to guarantee the stability and robustness of the system.

– Environmental documentation—The DBA should document every aspect of the DBMS environment, including hardware configuration and maintenance records, software updates, changes to the application and DBMS, and all other items related to changes made to the system. The DBA should be able to access these records and fully reproduce the current system as necessary.

– Consult with development team and end users—The DBA is often called on to act as a consultant to the development team as well as to the user community. This might involve personally assisting a single user or developing training courses for the user community as a whole.

– Evaluation of new software—The DBA might be called on to evaluate new software and make recommendations based on that evaluation. This might be related to a software purchase of a scheduled roll out of a new version of software. This evaluation must be done in the context of the stability of the system. It is your responsibility to maintain system stability and reliability.

– **Evaluation of new hardware and software purchases**—There is much consideration involved in purchasing new hardware and software. Much of this consideration involves the functionality and compatibility of the software or hardware as well as the cost of these components. Although the cost of the item is not usually a concern of the DBA, the functionality and compatibility is. The DBA might be asked to make recommendations based on whether these purchases make sense.

– **Capacity planning and sizing**—Determining whether it is necessary to purchase new hardware or software to meet increased loads is often a job for the DBA. Capacity planning and sizing is important to provide the level of service your users require. By anticipation the future needs of your users, you can provide an excellent level of service with no interruptions.

(c) Privacy key :

In relational database design, a unique key or Privacy key is a candidate key to uniquely identify each row in a table. A unique key or primary key composes a single column or set of columns. No two distinct rows in a table can have the same value (or combination of values) in those columns. Depending on its design, a table may have arbitrarily many unique keys but at most one primary key. A unique key must uniquely identify all possible rows that exist in a table and not only the currently existing rows. Examples of unique keys are social security numbers (associated with a specific person) or ISBNs (associated with a specific book). Telephone books and dictionaries cannot use names or words or Dewey Decimal system numbers as candidate keys because they do not uniquely identify telephone numbers or words. A primary key is a special case of unique keys. The major difference is that for unique keys the implicit NOT NULL constraint is not automatically enforced, while for primary keys it is. Thus, the values in a unique key columns may or may not be NULL. Another difference is that primary keys must be defined using another syntax.

Candidate key :

In the relational model, a candidate key of a relvar (relation variable) is a set of attributes of that relvar such that

1. At all times it holds in the relation assigned to that variable that there are no two distinct tuples with the same values for these attributes and
2. There is not a proper subset for which (1) holds.

Since a superkey is defined as a set of attributes for which (1) holds, we can also define a candidate key as a minimal superkey, i.e. a superkey of which no proper subset is also a superkey.

The importance of candidate keys is that they tell us how we can identify individual tuples in a relation. As such they are one of the most important of database constraint that should be specified when designing a database schema. Since a relation is a set (no duplicate elements), it holds that every relation will have at least one candidate key (because the entire heading is always a superkey).

Superkey :

A superkey is defined in the relational model of database organisation as a set of attributes of a relation variable (relvar) for which it holds that in all relations assigned to that variable there are no two distinct tuples (rows) that have the same values for the attributes in this set. Equivalently a superkey can also be defined as a set of attributes of a relvar upon which all attributes of the relvar are functionally dependent.

Not that if attribute set K is a superkey of relvar R, then at all times it is the case that the projection of R over K has the same cardinality as R itself.

Informally, a superkey is a set of columns within a table whose values can be used to uniquely identify a row. A candidate key is a minimal set of columns necessary to identify a row, this is also called a minimal superkey. For example, given an employee table, consisting of the columns employeeID, name, job, and departmentID, we could use the employeeID in combination with any or all other columns of this table to uniquely identify a row in the table. Examples of superkeys in this table would be {employeeID, name}, {employeeID, name, job}, and {employeeID, name, job, departmentID}.