UNIT 9 PHYSICAL FILE DESIGN AND DATA BASE DESIGN

Structure **Page Nos.**

9.0 INTRODUCTION

Once the logical system design has been finalized in consultation with the user, the details of physical design of the system can start. Physical database design involves actual implementation of the logical database in the DBMS. The requirements of physical design of the system require the logical design of the system. The database design involves three levels of design concepts namely Conceptual, Logical and Physical schemas. Conceptual model produces a data model which accounts for the relevant entities and relationships within the target application domain. Logical model ensures via normalization procedures and the definition of integrity rules that the stored database will be non-redundant and properly connected. Physical model specifies how database records are stored, accessed and related to ensure adequate performance. A good database design helps efficient storage and retrieval of data.

9.1 OBJECTIVES

After going through this unit, you should be able to understand the:

- advantage of databases over files;
- difference between logical and physical design;
- rules for good database design practices;
- define the concepts of fields, records and database;
- how to design the fields and records in a database table;
- understand various constrains enforced during database design;
- typical database design concepts; and
- explain the concepts of records, record types, and files, as well as the different techniques for placing file records on disk.

9.2 INTRODUCTION TO DATABASE DESIGN

Over a period of time, massive advancements have taken place in the field of databases. Storage of data and retrieval is an integral part of any information system.

9.2.1 Flat Files vs. Database

Traditionally, data are being kept in flat files. Consider an example of flat file which stores pin code of customer's address. Any change in the size of the field will enforce changes in the entire program which uses the data related to customers' address. If the file is being used by more than one application, the problem can become even worse.

9.2.2 Steps in Database Design

Analysis is the process of creating a conceptual data model independent of the target database technology. The typical result is an ER model.

Design is the process of creating a logical data model. This step is dependent on the target technology (relational, hierarchical, or network), but not on the specific database implementation (such as Oracle, MS SQL, DB2 for OS/390 or DB2 Universal Database).

Implementation is the process of creating a physical model or schema for one specific database system, such as Oracle, MS SQL, DB2. The result is an optimized physical design.

9.2.3 E-R Model to Database Design

The process of database design process involves the task of converting logical model to working physical model. The logical model can be arrived at by the application of normal forms. Normal forms are nothing but a set of rules applied sequentially starting from the basic table to the table resultant due to the application of a normal form. As of today, there are a total of five normal forms. The first normal form is applied to the basic table. The resultant table is given as input to the second normal form and so on. It is not mandatory to continue this process until the fifth normal form. Usually, the process is continued until third normal form. Fourth and fifth normal forms are applied only when there are multivalue dependencies and join dependencies respectively. We are not introducing you to the normalization theory in this course. It will be dealt in the course related to databases in detail. Physical database design starts from a given relational model. That is, from the definition of a set of tables and their respective columns. The objective of physical database design is to fulfil the performance requirements of a set of applications by optimizing the use of the DBMS. Key areas include: optimizing the index configuration, data placement and storage allocation.

Entities with one-to-one relationship should be merged into a single entity.

A table models each entity with a primary key and non-key attributes, some of whom may be foreign key(s).

One-to-many relationships are modeled by a foreign key attribute in the table representing the entity on the "many" side of the relationship.

Many-to-one relationship between two entities is modeled by a third table that has foreign keys that refer to the entities. These foreign keys should be included in the primary key of the relationship table, if appropriate.

Many commercially available tools can automate the process of converting a E-R model to a database schema. Figure 9.1 depicts various steps involved in the design of databases.

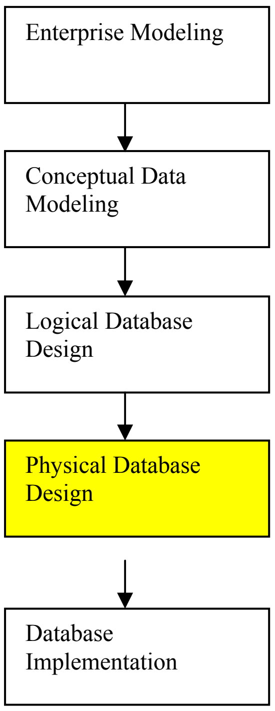


Figure 9.1: Steps in Database Design

9.2.4 Inputs to Physical Database Design

1. Logical structure of Database (Normalized relations).
2. Definition of attributes – data type, integrity control, error handling.
3. Choice of RDBMS
   a. Hierarchical
   b. Network
   c. Relational (DB2, MySQL)
   d. Object relational (Oracle 8i/9i)
4. Estimation of database size growth rate and frequency of usage.
5. Requirements for backup, recovery, response time and retention time.

9.2.5 Guidelines for Database Design

The following are various guidelines for Database Design :

- ensure that the data stored in the files (database tables) are atomic. Data stored in the atomic form can be combined later to generate data in specific form;
- every table must have a primary key which identifies each record in the table distinctly. Descriptive and meaningful name is to be used while naming a field in the table (For example, use *product_id* instead of *ID*);
- use single column primary key whenever possible. As most of the join operations are made on primary key and composite primary keys make the operation slower;
- use numeric key whenever possible;
- use primary key name as foreign key for better readability;
- avoid allowing null values to go into the columns that have discrete range of possible values; and
- avoid multiple tables with similar structure when one table is sufficient.

Figure 9.2 depicts various records and fields in a file which consist of details of various employees in a hospital.

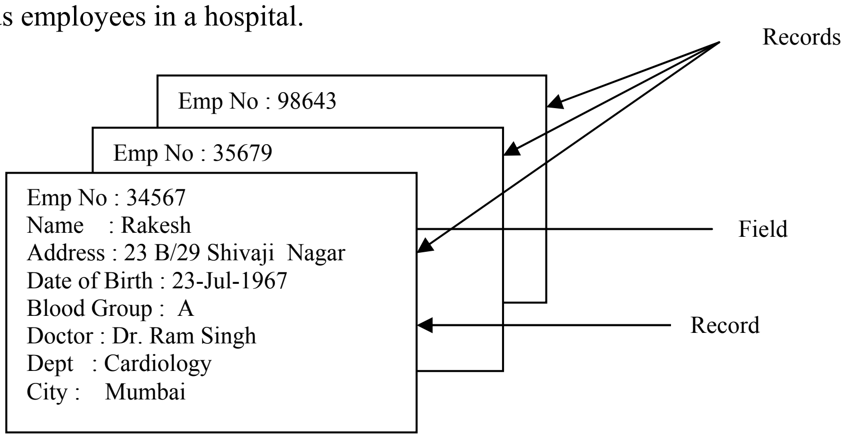


Figure 9.2: Records and Fields

Check Your Progress 1

1. The process of Database Design involves the task of converting logical model to ____________.
2. _______ types of relationship should be merged to a single entity.
3. Commercially available tools can automate the process of converting a ________to a database schema.

9.3 DESIGN OF DATABASE FIELDS

Attributes in E-R model are known as fields in physical data model. A field is the smallest unit of data that is stored and manipulated. Fields are used in conventional files as well as in databases. It is the implementation of attributes and can be termed as smallest unit of meaningful data.

9.3.1 Types of Fields

The following are various types of fields in databases:

Primary key: Any conventional file system or database stores two kind of fields namely descriptive fields and primary key. Descriptive fields comprise the customer names, inventory numbers, item descriptions, and so on, which are used by the application. Keys refer to the primary and foreign key that are used to find database records and relate them to one another.

Keys are fundamental to the concept of relational databases because they enable tables in the database to be related with each other.

A table must have a primary key i.e. an attribute or combination of attributes that are guaranteed to be unique and not null. It is sometimes helpful to introduce a surrogate field to act as a key. This could be a table attribute, which has no business meaning, but simply added to serve as a unique identifier for each record in the table. This is sometimes referred to as *plumbing*.

The requirements for a primary key are very hard. It must conform to the following rules:

- They should exist.
- Be unique in the table.
- The values must not change or become null during the life of each entity instance.
- It must have a not-null value for each instance of the entity.

Surrogate keys are often required because sometimes, real business data does not fulfil the requirement of a primary key. Furthermore, the surrogate key is typically a single field (not a composite key), which simplifies the database schema, particularly when the key is used in other tables as a foreign key.

Most of modern RDBMS are tuned in for queries on integers, so it is advisable to use this datatype as a primary key. Many RDBMS provide a special serial number or sequence number of integer type, which generate a sequence of unique integers as a row is inserted into the table. Declaring a column to be of this type guarantees that a unique key is generated for each inserted row.

Secondary key: Also known as *Alternate key*. This is a field or collection of fields in the table which can be used as primary key in addition to the already existing primary key.

Foreign keys are table attributes, the values of which are the same as those of primary keys of another table. It is often desirable to label foreign key columns explicitly. For instance, by adopting a naming convention. Existence of foreign key enforces the referential integrity constrains (discussed later in this Unit). A referential integrity constraint (references) should be declared as part of the CREATE statement in a DBMS while creating the table.

Descriptive fields: Attributes that are not used as key but store business data.

9.3.2 Rules for Naming Tables and Fields

Names for all database elements should be:

- Unique
- Meaningful
- Short

Restrictions for naming tables:

- Use no acronyms or abbreviations. Should be descriptive to convey meaning.
- Should not imply more than one subject

Restriction for naming fields:

- No acronyms
- Use abbreviations only if clear and meaningful
- Should not imply more than one subject
- Should be singular.

While designing database fields, it is required to set the properties of the fields.

Name: A name is used to refer the attribute in the DBMS that uniquely labels the field. The name of the attribute in the logical data model and the name of the field in the physical data model must be same. For example, *student name* in a *student* table.

Data type: Type of data the field is expected to store. This could be numeric, alphanumeric etc. The data type, supported by various RDBMS varies to a great extent. For example, student_name CHAR(25), indicates that the name of the student is of character data type, 25 indicates the maximum size of the data that can be stored in the field. The data type selected should ensure the following:

- it involves minimum usage of memory and represents all possible values
- supports all types of data manipulation that is expected from the business transaction.

Size: It indicates the size of the database fields. Many RDBMS support sizes that are variable. For example, VARCHAR data type in Oracle.

Null or not Null: specifies whether the field will accept null value. Not null constrains applied in DBMS ensure that null values are not entered to the respective fields. A null value is a special value distinct from 0 or blank. A null value indicates that the value is either missing or unassigned yet. We may specify that customer_name in a customer table to be not null. When a field is declared a primary key DBMS automatically ensures that the field in not null.

Domain: It indicates the range of values that are accepted by the fields. For example: Basic_Pay in a *employee* table can assume any value between the lowest basic_pay and highest basic_pay existing in the company. In such cases, the value of the field can be restricted to the one between the highest and lowest value to avoid entry of non-existing basic_pay.

Default value: It refers to the value that is stored by default in the field. For example, ship_date in a invoice is most of the time same as invoice_date (current date). When a default value is assigned to a field, it reduces a lot of data entry time and reduces the chances of error.

Referential integrity: It refers to a set of rules that avoid data inconsistency and quality problems. Referential integrity ensures that a foreign key value cannot be entered unless it matches a primary key value in another table. RDBMS automatically enforces the referential integrity once the database designer identifies and implements primary and foreign key relationship.

- It prevents orphaned records. i.e. when a row containing a foreign key is created, the referential integrity constrains enforced vide the RDBMS ensure that the same value also exists as a primary key in the related table.
- When a row is deleted, it should be ensured that no foreign key in related tables is the same value as primary key of the deleted row.

Figure 9.3 depicts primary and foreign keys for two tables namely *customer* and *order*.

Customer_id	**Customer_name**	**Customer_city**	**Customer_phone**
5466	John	New Delhi	2345678
5678	David	Mumbai	2567890

Table **Customer**

Order_no	**Order_date**	***Customer_id***	**Amount**
123456	12/3/2004	*5466*	Rs 345
345678	11/3/2003	*5678*	Rs. 567

Table **Order**

Figure 9.3: Primary key and Foreign key relationship

Customer_id is the primary key in **customer** table and is a foreign key in **order** table. This referential integrity constraints ensure that the value *customer_id* in **order** table must exist in the **customer** table. The primary key is shown in bold and the foreign key in intalics.

Check Your Progress 2

1. Attributes in E-R model are known as _______ in physical model of data.
2. __________ are often required because real business data sometimes does not fulfil the requirement of existence of a primary key.

3. A set of rules that avoid data inconsistency and quality problems is called ____________.

9.4 DESIGN OF PHYSICAL RECORDS

A Record is a collection of fields. Records are common to both databases and files. Records are collection of fields in a predefined format. The design of physical record involves putting the collection of fields in a single logical unit so that the fields are stored in adjacent locations for better storage and retrieval.

The main objective of the design of physical records is to store and retrieve them efficiently. Also, the fields should be stored in adjacent locations in such a way that the storage is used efficiently and speed of data processing is appropriate.

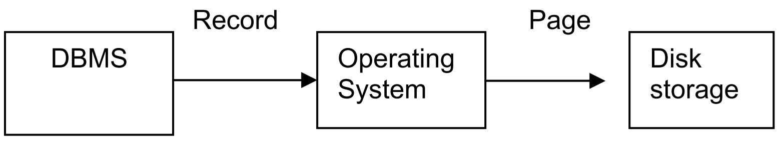


Figure 9.4: Process of storing a record physically

Physical pages or blocks are units of information moved between disk and memory buffers. They hold not only records, or table entries, but other information such as the amount of free space currently available in the block, the starting position of each record, etc. Blocks of data (pages) are normally read or written by the operating system. Page is referred to as the amount of data written in one I/O operation of operating system.
Blocking factor refers to the number of physical records per page. If a record size is 1340 bytes and the page size is 2048 bytes, then 708 bytes are wasted if DBMS does not allow physical records to span different pages. Selecting a block size involves a trade-off. In principle, the larger the block size, the fewer read-write operations need be performed to access a file by the operating system and therefore the more efficient is the processing. However, it requires a correspondingly large allocation of buffer space in memory. Since this is limited (and perhaps shared by many users), there is in practice, an upper bound. Moreover, large block sizes are primarily advantageous for sequential access.
Denormalization is the process of transforming normalized relations into unnormalized physical record specifications. The motivation behind de-normalization is poor performance of normalized table. The following may be of use for denormalization.

- Combine two entities with one-to-one relationship to one entity. This avoids the cost of joining two tables when the data are required from both the tables.
- Another form of de-normalization is to repeat the non key attribute (field) of one table in another table to facilitate the execution of query faster. However, it depends on the application at hand. Figure 9.5 depicts denormalization for optimized query processing.

Order_no	**Order_date**	***Customer_id***	**Amount**	**Customer_name**
123456	12/3/2004	*5466*	Rs 345	John
345678	11/3/2003	*8909*	Rs. 567	David

Figure 9.5: Denormalizations for Optimized Query Processing

In a particular application it is seen that queries about order also require the cutomer_name. In case of normalized table, this would always require joining **Customer** table and **order** table each time the query is processed. We have modified the **order** table by adding back the customer_name from the **customer** table in **order** table. Now all queries will require only the order table as all relevant information are available in this table.

Activities to enhance performance

1. Combining tables to avoid joins
2. Horizontal partitioning refers to placing different rows of a table into separate files. For example, in an **order** table order pertaining to different regions can be kept in a separate table for efficient retrieval of records.
3. Vertical partitioning refers to placing different columns of a table into separate files by repeating the primary key in each of the files.
4. Record partitioning refers to a combination of both horizontal and vertical partitioning as in distributed database processing
5. Data replication refers to the same data being stored in multiple places in the database.

Figures 9.6 and 9.7 depict table's structure in its original and improved versions.

Process of Denormalization of Tables

- Select the dominant process based on frequency of execution and frequency of data access;
- Define the join table for dominant process;
- Evaluate the cost of query, updates and storage for the schema. Consider possibility of renormalization to avoid table joins.

Fixed length records and variable length records

In fixed length record format, the length of record is always the same. The fixed length records are easier to design and implement but are more wasteful than variable length record formats when comes to utilization of space.

In variable length record format, the records are of variable length. To identify the end of a record, variable length records use end of record marker. It can be a special character.

Name	Date of Birth	Address	Phone
Rakesh	23-JUL-1967	12/B Patel Nagar, New Delhi - 110023	2234567
Ahbinav	8-NOV-1970	1201 Erose Appt., New Delhi - 110001	2236780
Roy	1-APR-1954	M-1034 Vanasthalipuram, Hyderabad-500001	2438723
Venkat	5-OCT-1969	24/A Gandhinagar, Vijayawada-520003	2658913
Ajay	5-AUG-1975	56 Canal Road, Patna- 600005	2753219
Sachin	8-SEP-1969	234 Mount Road, Varanasi-546987	2658703

Figure 9.6: Structure of **Students** table

Check Your Progress 3

1. The number of physical records per page is called __________.
2. Placing different rows of a table into separate files is called ___________.
3. The process of transforming normalized relations into un normalized table is called _____________.

Name	Date of Birth	Address	City	Pin Code	Phone
Rakesh	23-JUL-1967	12/B Patel Nagar	New Delhi	110023	2234567
Ahbinav	8-NOV-2001	1201 Erose Appt	New Delhi	110001	2236780
Roy	1-APR-1954	M-1034 Vanasthalipuram	Hyderabad	500001	2438723
Venkat	5-OCT-1969	24/A Gandhinagar	Vijayawada	520003	2658913
Ajay	5-AUG-1975	56 Canal Road	Patna	600005	2753219
Sachin	8-SEP-1969	234 Mount Road	Varanasi	546987	2658703

Figure 9.7: Improved Table Structure of Figure 9.6

9.5 DESIGN OF PHYSICAL FILES

Files are collection of logically connected records. In RDBMS, files are called tables. However, the way the files are stored in memory depend on the operating system. Many operating systems allow files to be split into pieces but the same is transparent to the user.

9.5.1 Types of Files

A **Master file** is a permanent file of all the data needed by a business. Some of the fields may be regularly updated from transactions. For example, a file consisting of information about customers.

A **Transaction file** is a temporary file of all the transactions (items bought, sold, deleted etc.) that have taken place in a given period. It stores data related to day to day business transactions. For example, data related to daily sales activity. The contents of these files are used to update the master file. Daily sales are used to update balance in the customer file.

An **Archive file** is a file of data in permanent storage (usually, the data is stored for legal reasons or to perform trend analysis). Archive files will contain all information about the past dealings of the business and would normally be stored on a different site to facilitate recovery in case of a disaster such as fire.

An **Audit file** is a file that does not store business data but data related to transaction log. For example, data and time of access, modification etc. of data , values of fields before and after modification etc.

A **Work file** is file temporarily created to hold intermediate result of the data processing. For example, a sorted file of list of customers.

9.5.2 File Organization

File organization is the physical organization of records on the disk. There are different types of file organizations depending on the organization of records in the disk and other secondary storage.

Before deciding on a specific file organization, we should ensure that its application leads to the following:

- Fast retrieval of records

- Reduce disk access time
- Efficient use of disk spaces.

Serial file organization

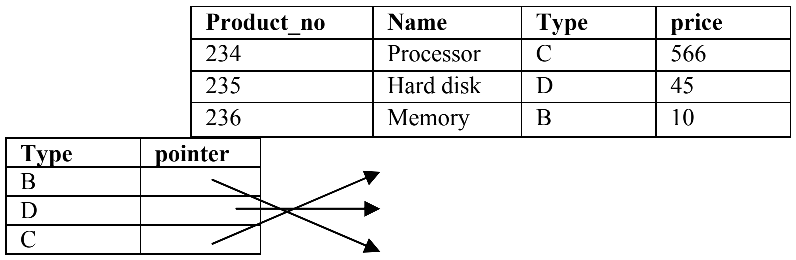
A serial file is created by placing the record as it is created. It leaves no gap between the records that are stored on the disk. The utilization of space called packing density approaches 100 percent in this case. Examples of serial files are print file, dump file, log files, and transaction files. These files are created once and are not used for addition or deletion or any kind of record searching operation.

Sequential file organization

In this organization, the records are physically ordered by primary key. To locate a particular record, the program starts searching from the beginning of the file till the matching primary key is found. Alphabetic list of customers is a common example of sequential file organization. Deletion of record may cause wastage of space and adding a new record requires rewriting of the file. This type of file organization is suitable for master files and is not used where fast response time is required.

Indexed sequential file organization

In this organization, records are not physically ordered. Index is created to facilitate searching of records. Index Records give physical location of each data record. Indexes are separate files with a link to the main file. This type of file organization is used where faster response time is required. Figure 9.8 depicts Indexed sequential file organization.



Product_no	Name	Type	price
234	Processor	C	566
235	Hard disk	D	45
236	Memory	B	10

Type	pointer
B	
D	
C	

Figure 9.8: Indexed Sequential File Organization

Hashed file organization

In this organization, records are physically ordered according to hashing algorithm. The address where each record is stored is determined using hashing algorithms. Figure 9.9 depicts an example of a Hashed file organization.
The following is a typical hashing algorithm :

1. Uses a field in record called the hash field (generally the key filed).
2. Divides by prime number known as hash function.
3. Produces record address called the hash address.

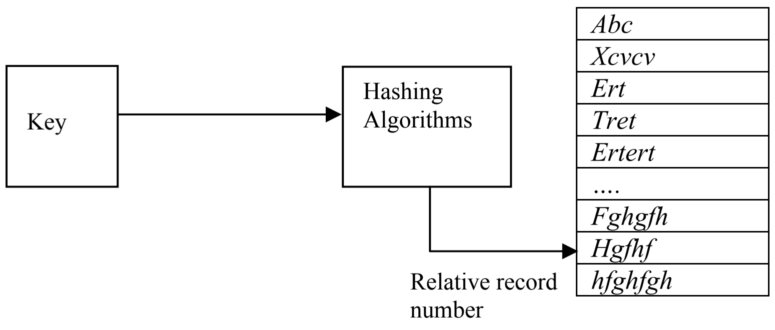


Figure 9.9 : Hashed File Organization

Check Your Progress 4

1. The Records are physically ordered by primary Key in __________ file organization.
2. ________ type of file organization is suitable for master files.

9.6 DESIGN OF DATABASE

Database design is similar to the pillars of a building. Any negligence, errors in Database design may lead to degraded performance of the software. In some cases such as real time applications, it may also lead to disasters.

The following are various steps in Database design :

- Selection of database architecture
- Designing database schema
- Selecting indexes
- Estimating capacity of the database.

Selection of database architecture

Selecting database architecture is one of the most challenging parts of database design for any information system**.** Before deciding on the target DBMS where the database is to be implemented, few considerations are required.

Hierarchical database structure is a kind of database management system that links records in tree data structure such that each record has only one owner. For example an order is owned by only one customer. It resembles a tree structure.

Network database structure is more flexible structure than Hierarchical model as well as relational model, but not preferred due to the high processing time. A neural network is an example of network database.

Relational database structure is most commonly used database model. The data is modeled as a mathematical relation. Reference key joins different tables together. Indexes provide rapid access to specific record in the database. For example, DB2, MySQL, Oracle are some RDBMS.

Object Oriented Database management system is based on object oriented paradigm. In object oriented database, data is stored as objects and can be interpreted only by using the methods specified by its class.

A blue print of the database is a physical model. A schema defines the database in terms of tables, keys, indexes and integrity rules. A **relational schema** consists of a relation, name of the attributes in the relations and restrictions on the relations called integrity constraints.

A **database schema** is a set of relation schemas. Changes to a schema or database schema are expensive. So, careful thought must be given to design of a database schema. The following are some guidelines for the design of a database schema.

- Each entity should be implemented as a database table.
- Each attribute should be implemented as a field.
- Each table must have a primary key and an index based on the key.
- Each table may have zero or more secondary keys.
- Appropriate foreign keys.

The following is an example of a database schema:

employee(emp_name, birth_date, social_security_no, dept_name),
department(dept_name, function).
Figure 9.10 depicts the database schema of the above.

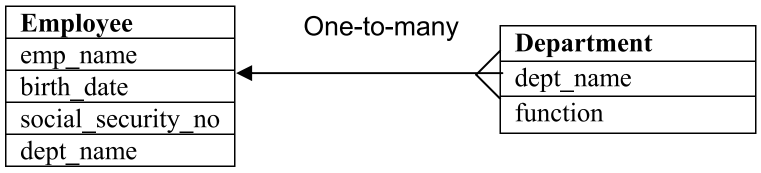


Figure 9.10: A Database Schema

A database schema defines a database in terms of tables, keys, indexes and constraints.

The following are various objects of a schema :

- Fields
- Records
- Tables
- Database

Selecting Indexes: During the process of file and database design one must choose index based on a single field (usually the primary key) or multiple fields. While selecting index, one must keep in mind the performance issues vis-à-vis the issue of inserting and deleting records. While indexes can be used generously on tables primarily used for query purpose with rare necessity to update records, they should be used judiciously in tables that support transaction processing which involve large insertion, updation and deletion operations. The amount of time needed to maintain the indexes in database tables increases with the number of rows stored.

When an index is created on a table, a separate storage area is allocated to store the index structure. A database table can have one or more indexes associated with it.

Figure 9.11: An index created on the EMP_ID field (one.jpg)

For example, consider an **employee** record. Figure 9.11 depicts an index created on the **EMP_ID** field of the **employee** table. The index table contains a sorted list of the employee ID values**.** Indexes may significantly improve the performance of SQL queries. It may not be noticed with small tables but it can be quite significant for larger tables. However, there are disadvantages to having too many indexes on table. Indexes can slow down the speed of some inserts, updates, and deletes when the

DBMS has to maintain the indexes as well as the database tables. Also, indexes take additional disk space as they are stored separately.

Example

In an **employee** database table, one will retrieve records based on employee name, department, or hire date. One may create three indexes-one on the DEPT field, one on the LAST_NAME field, and one on the HIRE_DATE field. The indexes are created on fields that appear in **where** clause of select statements.

If the boolean operator in the where conditions is AND (for example, CITY = 'Mumbai' AND STATE = 'MH'), then a concatenated index on the CITY and STATE fields will help. This index is also useful for retrieving records based on the CITY field.

If the boolean operator in the where condition is OR (for example, DEPT = 'EDP' OR HIRE_DATE > '01/30/03'), an index does not help performance. Therefore, index file need not be created.

Estimating capacity of the database: Database administrator needs to calculate the amount of disk space required for the database.

1. In a given table, calculate the record size by adding the sizes of various fields in it.
2. Also, the number of records that will be present in each table at a particular period of time should be forecast.

   Table size = record size * number of records in the table.

Database size is sum of sizes of all tables in that database. As rule of thumb, add a factor of 50% for indexes and other overheads to get the expected database size. While designing a database, future growth of database should also be kept in mind. Most of the business databases have a liner growth trend. Figure 9.12 depicts the growth of a database in relation to time.

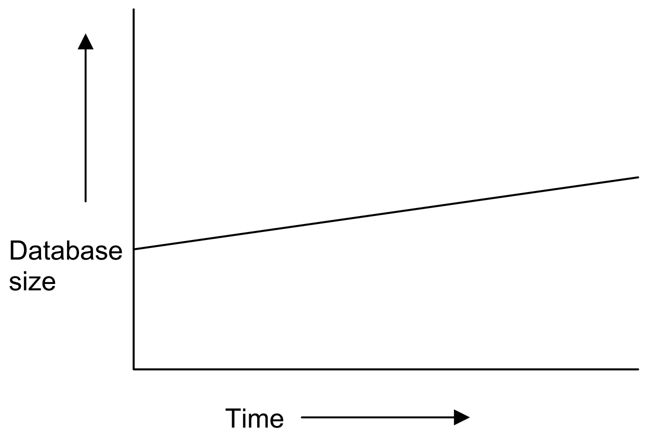


Figure 9.12 : Growth of Database

Check Your Progress 5

1. ________ is a blue print of the database.
2. A neural network is an example of __________.
3. Indexes take additional disk space? Yes/No.

9.7 CASE STUDY

The physical database design is the process of transforming a logical data model into an actual working physical database. A logical data model is required before designing a physical database. We will assume that the logical data model is complete, though. We will consider physical database design targeting a relational DBMS.

The very first step is to create an initial physical data model by transforming the logical data model into a physical implementation based on the target DBMS to be used for deployment. In reality, the physical data model depends largely on the target DBMS to be used for implementing the database. Therefore, to successfully create a physical database design requires a good working knowledge of the features of the DBMS including:

- knowledge of the database objects supported by DBMS, the physical structures and files required to support those objects;
- details regarding how the target DBMS supports indexing, referential integrity, constraints, data types, and other features that augment the functionality of database objects;
- knowledge of the DBMS configuration parameters and limitations; and
- data definition language (DDL) to translate the physical design into actual database objects.

We can create an effective and efficient database from a logical data model,i.e. E-R Diagram. The first step in transforming a logical data model into a physical model is to perform a simple translation from logical terms to physical objects. It may be noted that this simple transformation will not result in a complete and correct physical database design – it is simply the first step and can act as a template to further refining the database design. The transformation consists of the following steps:

- transforming entities in ER diagram into database tables;
- transforming attributes into table columns; and
- transforming domains into data types and constraints to primary key and foreign key relationship.

Deciding a primary key is an integral part of the physical design of entities and attributes. A primary key should be assigned for every entity in the logical data model. One should try to use the primary key as selected in the logical data model. However, if suitable attribute is not available, multiple attributes can be used as primary key. It may be required to choose a primary key other than the attributes in logical design by inserting a column that does not store business data (surrogate key) for physical implementation.

In a physical database, each table column must be assigned a data type supported by the DBMS. Certain data types require a maximum length to be specified, e.g.,a character data type could be specified as CHAR(25), indicating that up to 25 characters can be stored in the column.

Figure 9.13 depicts the E-R diagram of an **employee** database.

After following normal steps of transformation described above, further refinement of relations is possible before implanting on the target DBMS. One such example is shown below.

In the above ER diagram, Department being a multivalued attribute, we will convert this multivalued Attribute into relation namely Department.

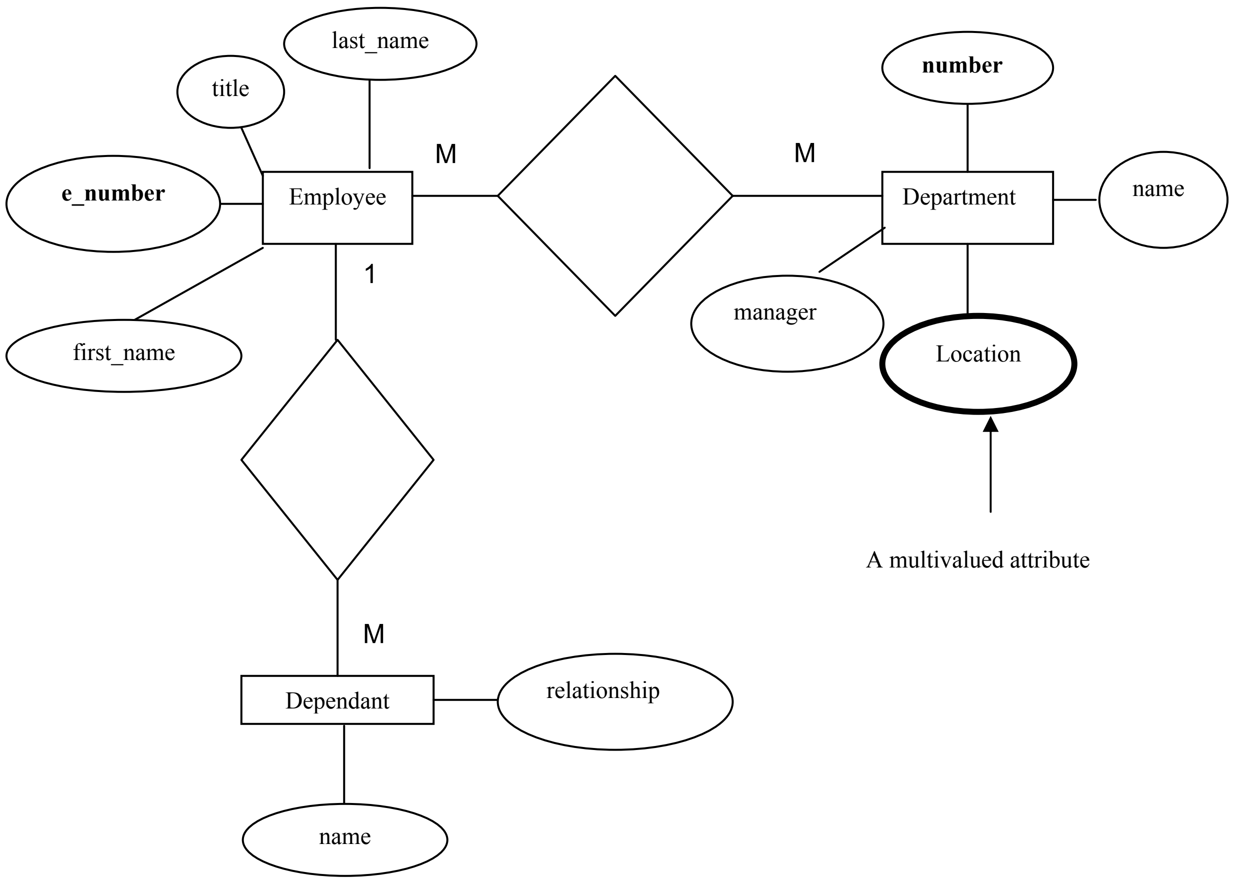


Figure 9.13: E-R Diagram for Employee database

e_number	Title	Last_name	First_name	Supervisor_emp_no

EMPLOYEE

number	Name	Manager

DEPARTMENT

Dept_no	Location

LOCATION

Emp_number	**Name**	relationship

DEPENDENT

Figure 9.14: A Relational Database Schema

Figure 9.14 shows the nearest relational database schema for the E-R diagram depicted in Figure 9.13. The aim of database design should be to create a database plan to fit present and future objectives. A logical data model should be used as the blueprint for designing and creating a physical database. But, the physical database cannot be created properly with a simple logical to physical mapping. Physical database design decisions need to be made by the Data Base Administrator before implementing physical database structures to the target DBMS. Sometimes, this may require deviation from the logical data model.

9.8 SUMMARY

Traditionally, file is being used for keeping data. Use of DBMS has been the standard to store data for today's information systems due to their various advantages. Any physical database design involves steps like choice of target DBMS on which the database is going to be implemented. Relational database is mostly being used unless the application has specific requirements. The physical database design process can be considered as a mapping from logical model to physical working database, which involves design of fields, design of records and finally design of the database. While transforming the logical model to physical model, many implementation issues related to the information system and target DBMS are to be addressed. Database volume estimation is an important part of database design. The present size and future growth of database is to be estimated before implementing the database.

9.9 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. Working physical model
2. One-to-one
3. E-R Model

Check Your Progress 2

1. Field
2. Surrogate keys
3. Referential integrity rules

Check Your Progress 3

1. Blocking factor
2. Horizontal partitioning
3. Denormalization

Check Your Progress 4

1. Sequential
2. Sequential

Check Your Progress 5

1. Database Schema
2. Network database
3. Yes

9. 10 FURTHER READINGS

Jeffrey L. Whitten, Lonnie D. Bentley, Kevin C. Dittman; *Systems Analysis and Design Methods*; Tata McGraw Hill; Fifth Edition;2001.

Joey George, J Hoffer and Joseph Valacich; Pearson Education, *Modern System Analysis and Design*;2001.

Reference Websites

http://seastorm.ncl.ac.uk/itti/create.html#create
http://www.rspa.com