

Database Management System

Unit-1

Dharmendra Kumar(Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research, Prayagraj

INTRODUCTION TO DATABASE SYSTEM

1 Database

- It is a collection of interrelated data of an enterprise in a particular subject.
- It is a collection of data in an organized manner in a persistent media so that storing and retrieving data will be easier.

For example, a university database might contain information about the following:-

Entities such as students, faculty, courses, and classrooms.

Relationships between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.

2 Database Management System(DBMS)

- It is a collection of programs(software, which is used to create, manipulate(insert, delete, retrieve, update) data in database.
- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

3 Applications of DBMS

Databases are widely used. Here are some representative applications:

- **Banking:** In banking database, we record the information about customers, accounts, loans and banking transactions.
- **Railway reservation system:** we record the information about trains, reservation, passengers.
- **Airlines:** We record reservations and schedule information, flight information.
- **Universities:** We record student information, course registrations, and grades.
- **Credit card transactions:** We record purchases on credit cards and generation of monthly statements.
- **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
- **Sales:** For customer, product, and purchase information.

- **Human resources:** For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.
- **Hospital system:** We record information about doctors, patients, services available, rooms details, employees etc.
- **Hotel system:** We record information about rooms, customers and employees details.

4 Drawbacks of File system

To keep information in such file-processing system, there are a number of major disadvantages:-

Data redundancy: Data redundancy refers to the duplication of data, let's say we are managing the data of a college where a student is enrolled for two courses, the same student details in such case will be stored twice, which will take more storage than needed. Data redundancy often leads to higher storage costs and poor access time.

Data inconsistency: Data redundancy leads to data inconsistency, let's take the same example that we have taken above, a student is enrolled for two courses and we have student address stored twice, now let's say student requests to change his address, if the address is changed at one place and not on all the records then this can lead to data inconsistency.

Difficulty in accessing data: In file system the data is stored in the files. Whenever data has to be retrieved as per the requirements then a new application program has to be written. This is tedious process.

Data isolation: Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

Integrity problems: The data values stored in the database must satisfy certain types of consistency constraints.

For example, the balance of a bank account may never fall below a prescribed amount. These constraints are enforced in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

Atomicity problems: Atomicity of a transaction refers to "All or nothing", which means either all the operations in a transaction executes or none.

For example: Let's say Steve transfers 100\$ to Negan's account. This transaction consists multiple operations such as debit 100\$ from Steve's account, credit 100\$ to Negan's account. Like any other device, a computer system can fail let's say it fails after first operation then in that case Steve's account would have been debited by 100\$ but the amount was not credited to Negan's account, in such case the rollback of operation should occur to maintain the atomicity of transaction. It is difficult to achieve atomicity in file processing systems.

Concurrent-access anomalies: Concurrent access means multiple users can access database simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data.

Consider bank account A, containing \$500. If two customers withdraw funds (say \$50 and \$100 respectively) from account A at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. There is no central control of data in classical file organization. So, the concurrent access of data by many users is difficult to implement.

Security problems: Data should be secured from unauthorised access, for example a student in a college should not be able to see the payroll details of the teachers, such kind of security constraints are difficult to apply in file processing systems. Not every user of the database system should be able to access all the data.

5 Advantage of DBMS over file system

There are several advantages of Database management system over file system. Few of them are as follows:

No redundant data: Redundancy removed by data normalization. No data duplication saves storage and improves access time.

Data Consistency and Integrity: As we discussed earlier the root cause of data inconsistency is data redundancy, since data normalization takes care of the data redundancy, data inconsistency also been taken care of as part of it.

Data Security: It is easier to apply access constraints in database systems so that only authorized user is able to access the data. Each user has a different set of access thus data is secured from the issues such as identity theft, data leaks and misuse of data.

Privacy: Limited access means privacy of data.

Easy access to data: Database systems manages data in such a way so that the data is easily accessible with fast response times.

Easy recovery: Since database systems keeps the backup of data, it is easier to do a full recovery of data in case of a failure.

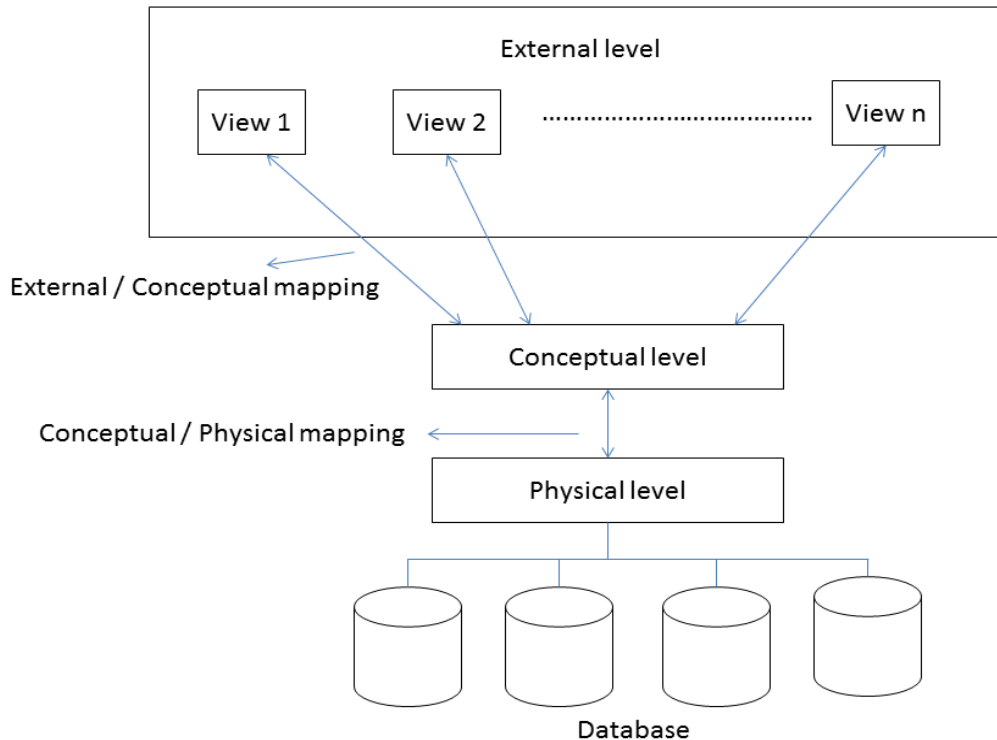
Flexible: Database systems are more flexible than file processing systems.

6 Disadvantages of DBMS

- DBMS implementation cost is high compared to the file system
- Complexity: Database systems are complex to understand
- Performance: Database systems are generic, making them suitable for various applications. However this feature affect their performance for some applications

7 Data abstraction model or Three level data abstraction architecture

Data abstraction means to hide the information. To hide the information from some users, three levels of abstraction is used.



This architecture has three levels:

1. External level
2. Conceptual level
3. Physical level

1. External level

It is also called view level. The reason this level is called “view” is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping. The user doesn’t need to know the database schema details such as data structure, table definition etc. User is only concerned about data which is what returned back to the view level after it has been fetched from database (present at the internal level). External level is the “top level” of the Three Level DBMS Architecture.

2. Conceptual level

It is also called logical level. The conceptual level describes the structure of the whole database. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.

3. Physical level

This level is also known as internal level. This level describes how the data is actually stored in the storage devices. This level is also responsible for allocating space to the data. The internal level describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database. This is the lowest level of the architecture.

8 Instances and Schema

8.1 Schema

The overall design of the database is called the database schema.

Schema is of three types: Physical schema, logical schema and view schema.

Physical schema: The design of a database at physical level is called physical schema, how the data stored in blocks of storage is described at this level.

Logical schema: Design of database at logical level is called logical schema, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

View schema: Design of database at view level is called view schema. This generally describes end user interaction with database systems.

8.2 Instance

The collection of information stored in the database at a particular moment is called an instance of the database.

9 Data Independence

Data independence can be explained using the three-schema architecture.

Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

There are two types of data independence:

9.1 Logical Data Independence

- Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual view.
- If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
- Logical data independence occurs at the user interface level.

9.2 Physical Data Independence

- Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
- If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
- Physical data independence is used to separate conceptual levels from the internal levels.
- Physical data independence occurs at the logical interface level.

10 Data Models

Data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. Data models define how the logical structure of a database is modeled. Data models define how data is connected to each other and how they are processed and stored inside the system.

There are different types of data models are used in DBMS. Here, we are going to explain some data models.

- (1) Entity relationship model
- (2) Relational model
- (3) Network model
- (4) Hierarchical model

Entity-relationship model

The entity-relationship (E-R) data model is based on a perception of a real world that consists of a collection of basic objects, called entities, and of relationships among these objects. An entity is a “thing” or “object” in the real world that is distinguishable from other objects. For example, each person is an entity, and bank accounts can be considered as entities.

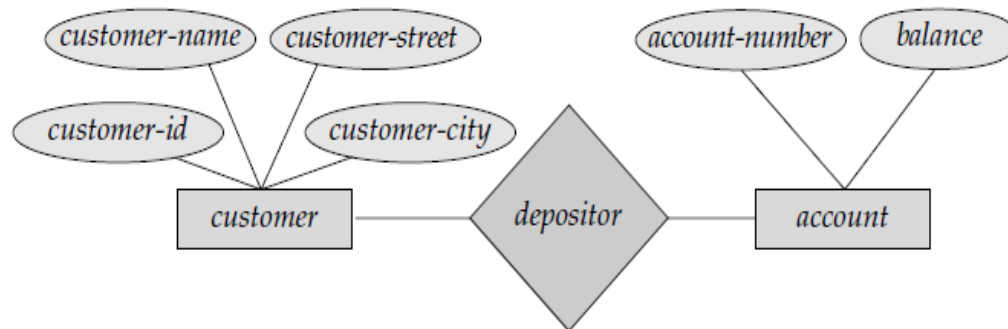
Entities are described in a database by a set of attributes. For example, the attributes account-number and balance may describe one particular account in a bank, and they form attributes of the account entity set. Similarly, attributes customer-name, customer-street address and customer-city may describe a customer entity.

A relationship is an association among several entities. For example, a depositor relationship associates a customer with each account that she has. The set of all entities of the same type and the set of all relationships of the same type are termed an entity set and relationship set, respectively.

The overall logical structure (schema) of a database can be expressed graphically by an E-R diagram, which is built up from the following components:

- **Rectangles**, which represent entity sets
- **Ellipses**, which represent attributes
- **Diamonds**, which represent relationships among entity sets
- **Lines**, which link attributes to entity sets and entity sets to relationships

For example, consider part of a database banking system consisting of customers and of the accounts that these customers have. Following figure shows the corresponding E-R diagram. The E-R diagram indicates that there are two entity sets, customer and account, with attributes as outlined earlier. The diagram also shows a relationship depositor between customer and account.



E-R diagram

Relational model

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Following figure presents a sample relational database comprising three tables: One shows details of bank customers, the second shows accounts, and the third shows which accounts belong to which customers.

The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type.

The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model. The relational model is at a lower level of abstraction than the E-R model. Database designs are often carried out in the E-R model, and then translated to the relational model.

Network model

A network database consists of a collection of records connected to one another through links. A record is in many respects similar to an entity in the E-R model. Each record is a collection of fields (attributes), each of which contains only one data value. A link is an association between precisely two records. Thus, a link can be viewed as a restricted (binary) form of relationship in the sense of the E-R model.

A schema representing the design of a network database is said to be Data-structure diagram. Such a diagram consists of two basic components:

<i>customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

Customer table

<i>account-number</i>	<i>balance</i>
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

Account table

<i>customer-id</i>	<i>account-number</i>
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

Depositor table

1. Boxes, which correspond to record types
2. Lines, which correspond to links

A data-structure diagram serves the same purpose as an E-R diagram; namely, it specifies the overall logical structure of the database. Every E-R diagrams can be transformed into their corresponding data-structure diagrams.

Example: Consider the following E-R diagram:-

This E-R diagram is consisting of two entity sets, customer and account, related through a binary, many-to-many relationship depositor, with no descriptive attributes. This diagram specifies that a customer may have several accounts, and that an account may belong to several different customers.

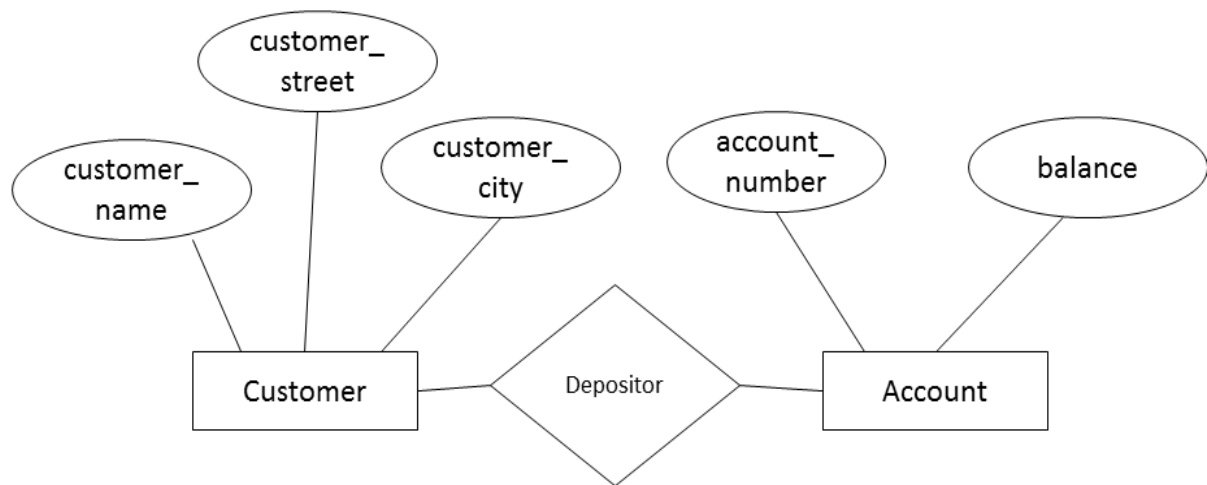
The data-structure diagram for the corresponding E-R diagram is the following:-

The record type customer corresponds to the entity set customer. It includes three fields—customer_name, customer_street, and customer_city. Similarly, account is the record type corresponding to the entity set account. It includes the two fields account_number and balance. Finally, the relationship depositor has been replaced with the link depositor.

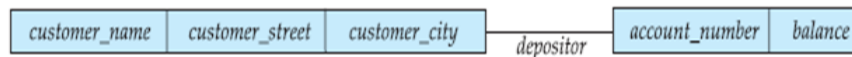
Here, the relationship depositor is many to many. If the relationship depositor were one to many from customer to account, then the link depositor would have an arrow pointing to customer record type. Similarly, if the relationship depositor were one to one, then the link depositor would have two arrows: one pointing to account record type and one pointing to customer record type.

A sample database corresponding to the data-structure diagram of Figure 1.1 is shown in Figure 1.2.

If a relationship includes descriptive attributes, the transformation from an E-R diagram to a data-structure diagram is more complicated. A link cannot contain any data value,



E-R diagram



Data-structure diagram

Figure 1: data-structure diagram

so a new record type needs to be created and links need to be established.

Example: Consider the following E-R diagram.

The data structure diagram of this E-R diagram is shown in the following figure:-

Sample database corresponding to above data-structure diagram is the following:-

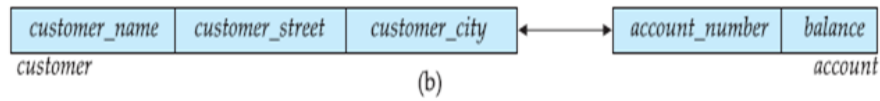
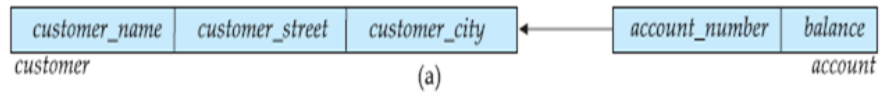
Hierarchical model

A hierarchical database consists of a collection of records that are connected to each other through links. A record is similar to a record in the network model. Each record is a collection of fields (attributes), each of which contains only one data value. A link is an association between precisely two records. Thus, a link here is similar to a link in the network model.

Consider a database that represents a customer-account relationship in a banking system. There are two record types: customer and account. The set of all customer and account records is organized in the form of a rooted tree, where the root of the tree is a dummy node. A hierarchical database is a collection of such rooted trees, and hence forms a forest. We shall refer to each such rooted tree as a database tree.

The schema for a hierarchical database is represented by tree-structured diagram. Such a diagram consists of two basic components:

1. Boxes, which correspond to record types



Two data-structure diagram

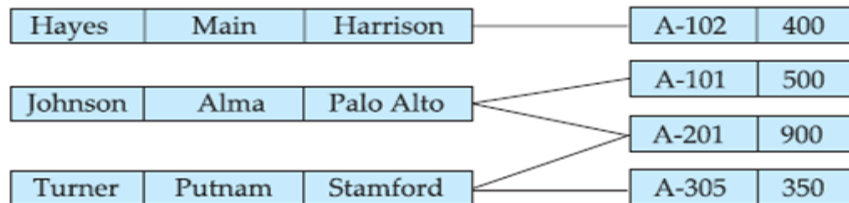


Figure 2: Sample database for data-structure diagram shown in figure 1.1

2. Lines, which correspond to links

A tree-structure diagram serves the same purpose as an entity-relationship (E-R) diagram; namely, it specifies the overall logical structure of the database. A tree structure diagram is similar to a data-structure diagram in the network model. The main difference is that, in the latter, record types are organized in the form of an arbitrary graph, whereas in the former, record types are organized in the form of a rooted tree. The relationships formed in the tree-structure diagram must be such that only one-to-many or one-to-one relationships exist between a parent and a child. The general form of a tree-structure diagram is shown in the following figure:-

The database schema is represented as a collection of tree-structure diagrams. For each such diagram, there exists one single instance of a database tree. The root of this tree is a dummy node. The children of the dummy node are instances of the root record type in the tree-structure diagram. Each record instance may, in turn, have several children, which are instances of various record types, as specified in the corresponding tree-structure diagram. Every E-R diagram can be transformed into tree-structure diagram.

Example: Consider the following E-R diagram:-

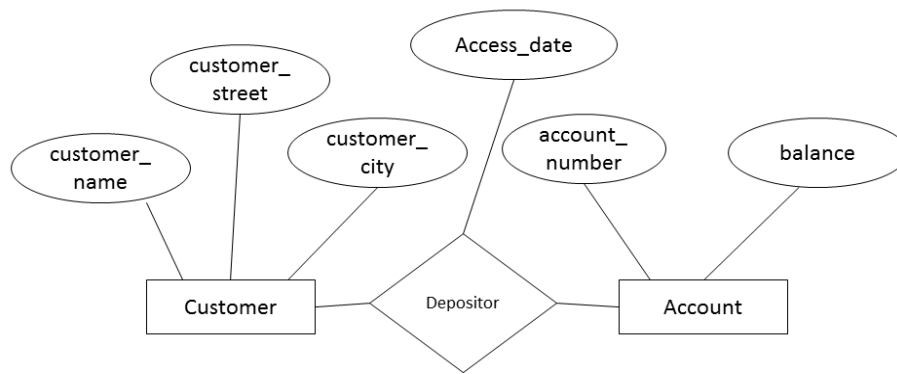
Tree structure diagram for this E-R diagram is the following:-

A sample database corresponding to the above tree-structure diagram is shown in the following figure:-

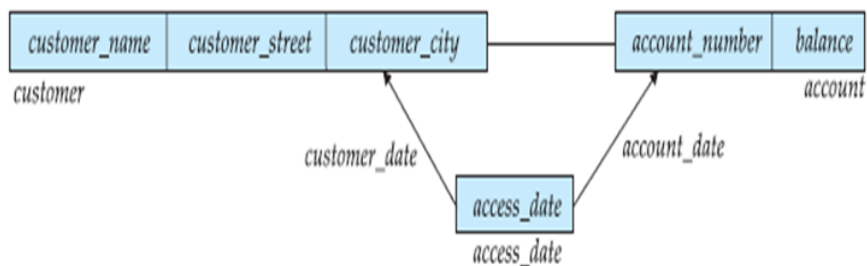
There are two database trees. The first tree (Figure a) corresponds to the tree-structure diagram T1; the second tree (Figure b) corresponds to the tree-structure diagram T2.

Example: Consider the following E-R diagram:-

Tree structure diagram for this E-R diagram is the following:-



E-R diagram



A sample database corresponding to the above tree-structure diagram is shown in the following figure:-

11 Database languages

A database system provides a data definition language to specify the database schema and a data manipulation language to express database queries and updates. In practice, the data definition and data manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.

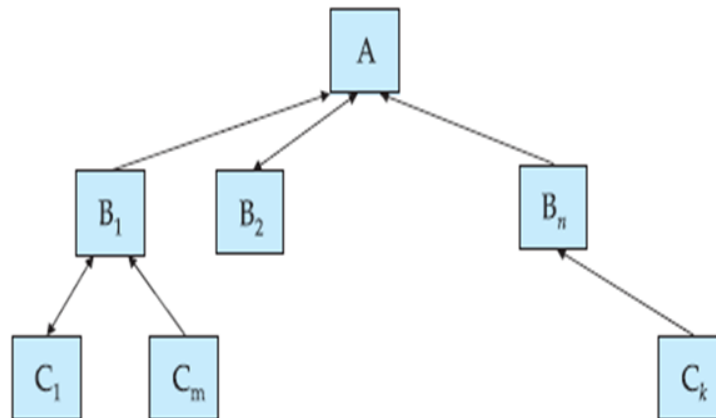
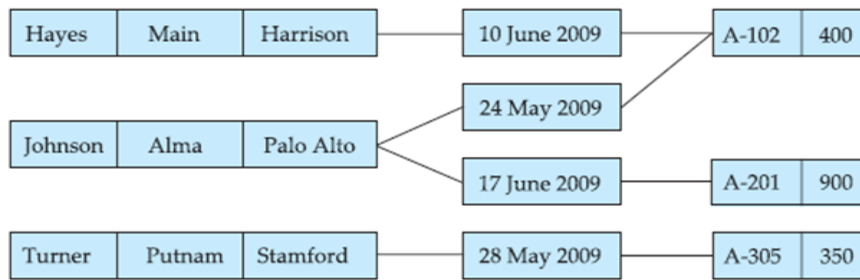
11.1 Data-Definition Language

We specify a database schema by a set of definitions expressed by a special language called a data-definition language (DDL).

It is a Set of SQL Commands used to create, modify, and delete database objects such as tables, views, indices etc. . It is normally used by DBA and Database engineers. It provides command like -

These are all the commands of Data Definition Language. These all Commands are shown in detail with their syntax and example:

1. CREATE :



- The create command is used to create a table.
- A Table name should be unique, i.e., it must not match with existing tables.
- A Table name and column name must start with alphabet, must not match with reserved keywords, and should be combination of A-Z, a-z, 0-9, and '_' (underscore) having maximum length up to 30 characters.
- Each column definition requires name, data type and size for that column.
- Table name and column name are not case sensitive generally . But if they are enclosed within double quotes, then they are case sensitive.
- Each column definition is separated from other by a ',' (comma).
- The entire SQL statement is terminated with ';' (semi colon)

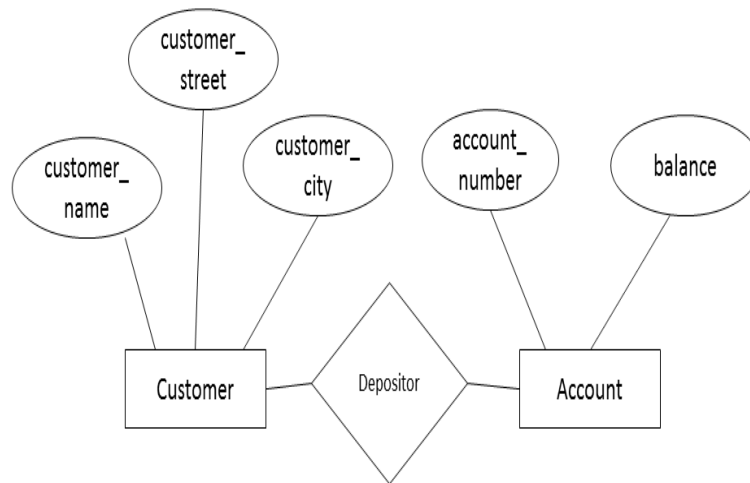
Syntax:

CREATE TABLE tablename(columnName1 datatype(size),columnName2 datatype(size),....., columnNameN datatype(size));

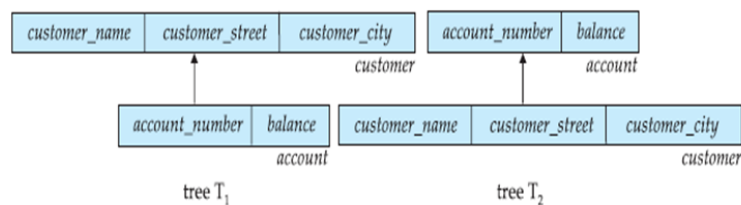
Example: Table creation is shown in the following figure:-

2. ALTER :

- Alter command used to modify structures of a table.
- Alter command can be used to add ,modify, or drop columns in a table.



E-R diagram



- Alter command can be used for this purpose are described below:

A. Adding New Columns :

This command adds a new columns in an existing table. Initially, this column will not contain any data. If required, data can be filled for this column using UPDATE command.

Syntax:

Alter table TableName Add (NewColumnName Datatype(size), NewColumnName Datatype(size)...);

Example: Addition of new column is shown in the following figure:-

B. Dropping Columns :

This command deletes an existing column from the table along with the data held by that column.

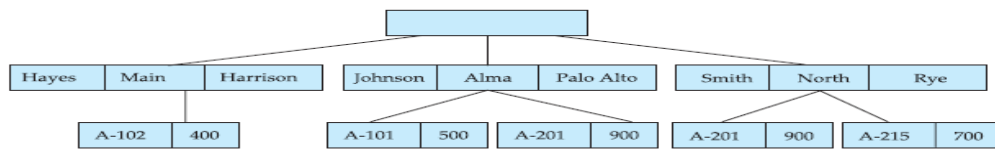
Syntax:

Alter table TableName Drop column columnName;

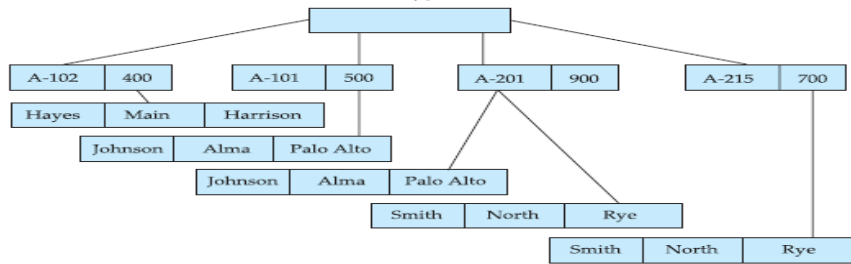
Example: Deletion of a column from a table is shown in the following figure:-

C. Modifying Columns :

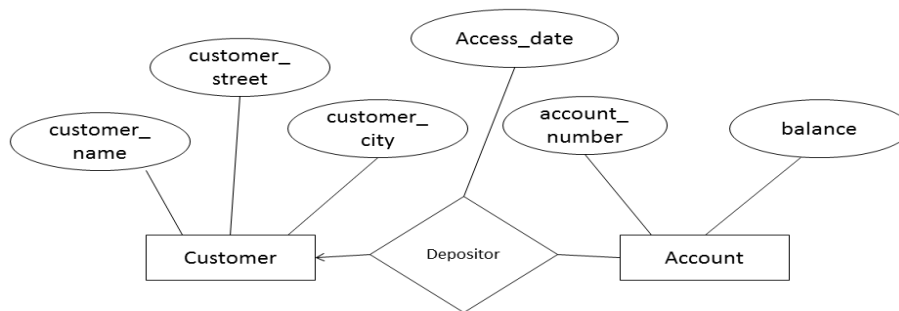
This command sets newDatatype and newSize as datatype and size for specified column respectively. The main aim of this command is to modify or change the datatype and size of the column.



(a)



(b)



E-R diagram

Syntax :

Alter table TableName Modify(columnName newDatatype(newSize));

Example:

3. DROP :

- DROP TABLE command is used to delete or destroy table from a database.
- The DROP TABLE command drops the specified table. This means, all records along with structure of the table will be destroyed.
- Care must be taken while using this command, as all records held within the table are lost and cannot be recovered.

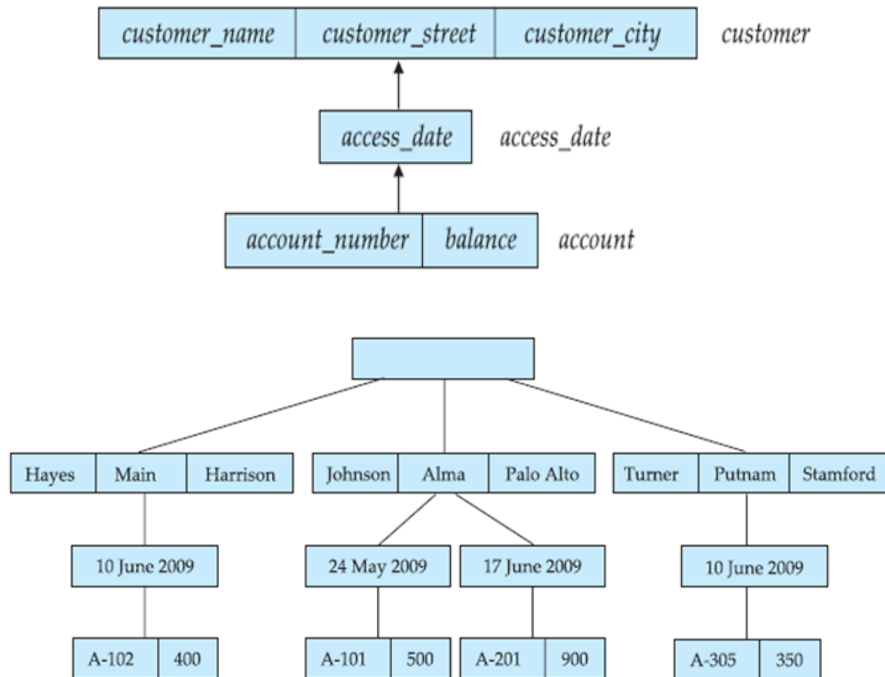
Syntax :

drop table tablename;

Example:

4. TRUNCATE :

- TRUNCATE TABLE is used to delete all data from a table.



- Logically, this is equivalent to DELETE statement that deletes all rows without using WHERE clause.
- TRUNCATE operation drops and re-creates the table. This is must faster than deleting all rows one by one.
- The deleted records cannot be recovered in truncate operation. while in delete operation, deleted records can be recovered using ROLLBACK statement.

Syntax :

truncate table tablename;

Example:

11.2 Data Manipulation Language

Data manipulation is

- The retrieval of information stored in the database
- The insertion of new information into the database
- The deletion of information from the database
- The modification of information stored in the database

A data manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. There are basically two types:

Command	Description
CREATE	to create objects in a database
ALTER	to modify existing data in a table
DROP	to delete objects from the database
TRUNCATE	to remove all records from the table

```

SQL>connect
Enter user-name:system
Enter password:
Connected.

SQL>create table student(enroll_no number(12),name varchar2(10),dept
varchar2(12));
Table created.

SQL> describe student;

Name                                Null?                                Type
-----
ENROLL_NO                           NUMBER(12)
NAME                                VARCHAR2(10)
DEPT                                VARCHAR2(12)

```

- Procedural DMLs require a user to specify what data are needed and how to get those data.
- Declarative DMLs (also referred to as non-procedural DMLs) require a user to specify what data are needed without specifying how to get those data.

A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language.

Following commands in SQL are used to manipulate database.

These all Commands are shown in detail with their syntax and example:

Command	Description
INSERT	insert data into table
SELECT	retrieve data from the table
UPDATE	modify existing data in the table
DELETE	delete records from the table

1. INSERT :

- The INSERT command is used insert the data into a table or create a new row in table.
- To insert user data into tables, "INSERT INTO ..." SQL statement is used. and stores the inserted values into respective columns.

Syntax :

insert into tablename (column1, column2, columnN) Values (expression1, expression2, ...

```
Run SQL Command Line

SQL>alter table student add(age number(10));

Table altered.

SQL> describe student;

Name                          Null?                          Type
-----
ENROLL_NO                     NUMBER(12)
NAME                          VARCHAR2(10)
DEPT                          VARCHAR2(12)
AGE                           NUMBER(10)
```

```
Run SQL Command Line

SQL>alter table student drop column age;

Table altered.

SQL> describe student;

Name                          Null?                          Type
-----
ENROLL_NO                     NUMBER(12)
NAME                          VARCHAR2(10)
DEPT                          VARCHAR2(12)
```

, expressionN); Example:

Example:

2. SELECT :

- The SELECT command is used to retrieve selected rows from one or more tables and displays on the screen. It is most widely used and required statement among all others in SQL.
- Once a table is loaded with user data, these data can be retrieved in number of different manners.
- Here, an asterisk (' * ') is used as the meta character, and it indicates all the columns of a given table.

```
SQL>alter table student modify(name varchar2(15));

Table altered.

SQL> describe student;

Name                               Null?                                Type
-----
ENROLL_NO                           NUMBER(12)
NAME                                VARCHAR2(15)
DEPT                                VARCHAR2(12)

SQL>drop table student;

Table dropped.

SQL> describe student;

ERROR:
ORA-04043: object student does not exist
```

- This statement retrieves all the columns and all the rows of the table.

Syntax :

select * from tablename;

Example:

There are three ways of table data filtering as given below:

- (A) Selected columns, All Rows
- (B) Selected Rows, All Columns
- (C) Selected columns, Selected Rows

Variations of the basic SELECT statement can be used to retrieve selected data as described below:

A. SELECTED COLUMNS, ALL ROWS

- This Statement retrieves only selected columns as specified with SELECT clause.
- This Statement retrieves all the row of the table.

```
SQL>truncate table student;

Table truncated.

SQL>Select * from student;

no rows selected.

SQL>insert into student values(7512,'parimal','computer');

1 row created.

SQL>insert into student values(7503,'preet','computer');

1 row created.
```

Syntax :

select column1, column2, ..., columnN from tablename;

Example:

B. SELECTED ROWS, ALL COLUMNS

- This Statement retrieves all the columns of the table.
- This statement retrieves only specific rows that specify the condition given with WHERE clause.
- Multiple conditions can be combined with logical operators such as AND and OR.

Syntax :

select * from tablename WHERE condition;

Example:

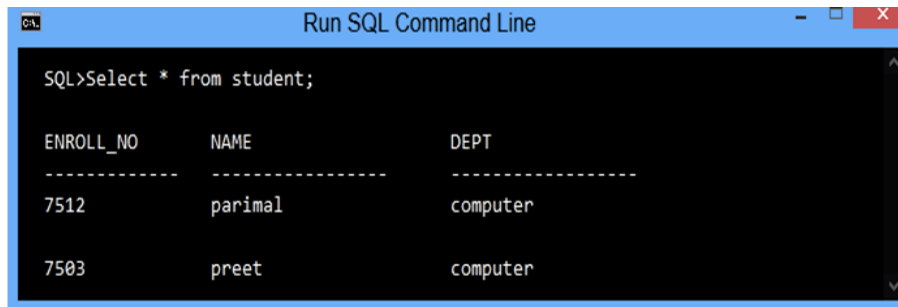
C. SELECTED ROWS, SELECTED COLUMNS

- This Statement retrieves only Selected columns as specified with SELECT clause.
- Also, retrieves only specific rows that specify the condition given with WHERE clause.

Syntax :

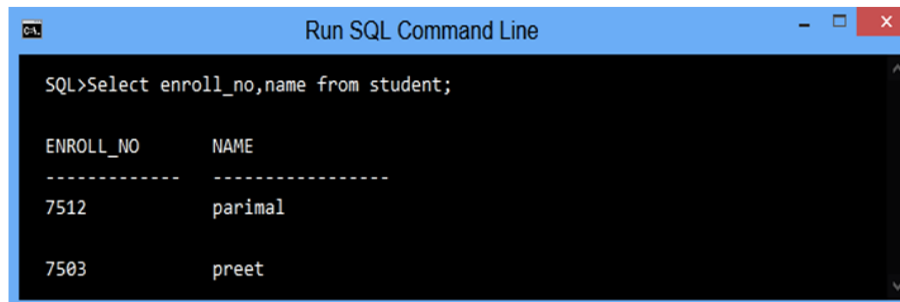
select column1, column2, ..., columnN from tablename WHERE condition;

Example:



```
SQL>Select * from student;
```

ENROLL_NO	NAME	DEPT
7512	parimal	computer
7503	preet	computer



```
SQL>Select enroll_no,name from student;
```

ENROLL_NO	NAME
7512	parimal
7503	preet

3. UPDATE :

- The UPDATE command can be used to change or modify the data values in a table.
- It can be used to update either all rows or a set of rows from a table.
- This update command updates all rows from the table, and displays message regarding how many rows have been updated.
- The SET clause specifies which column data to modify.
- An expression can be a constant value, a variable, or some expression and it specifies the new value for related columns.
- You can update specific rows by the WHERE clause, and displays message regarding how many rows have been updated.

Syntax :

update tablename set column1=expression1,column2=expression2 where condition;

Example:

4. DELETE :

- The DELETE command can be used to remove either all rows of a table, or a set of rows from a table.
- The DELETE command deletes all rows from the table, and displays message regarding how many rows have been deleted.
- The DELETE command deletes rows from the table that satisfy the condition provided by WHERE clause. It also displays message regarding how many rows have been deleted.

```

SQL>Select * from student where enroll_no=7503;

ENROLL_NO    NAME    DEPT
-----
7503         preet   computer

SQL>Select name from student where dept='computer';

NAME
-----
parimal
preet

```

Syntax :

delete from tablename;

Example:

11.3 Data Control Language

Data Control Language(DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables or views, a user needs privileges.

In DCL, we have following two commands:

GRANT: It is used to provide any user access privileges or other privileges for the database.

REVOKE: It is used to take back permissions from any user.

11.4 Transaction Control Language

Transaction Control Language(TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements. It also allows statements to be grouped together into logical transactions.

In TCL, we have following three command-

COMMIT Command: COMMIT command is used to permanently save any transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed. The changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Following is commit command's syntax,

```
SQL>Update student set enroll_no=03 where name='preet';

1 row updated.

SQL>select * from student;

ENROLL_NO      NAME      DEPT
-----
7512      parimal      computer
03      preet      computer

SQL>delete from student;

2 rows deleted.

SQL>select * from student;

no rows selected.
```

COMMIT;

ROLLBACK command

This command restores the database to last committed state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction. If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not committed using the COMMIT command.

Following is rollback command's syntax,

```
ROLLBACK TO savepoint_name;
```

SAVEPOINT command

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Following is savepoint command's syntax,

```
SAVEPOINT savepoint_name;
```

Example:

```
INSERT INTO class VALUES(5, 'Rahul');
COMMIT;
UPDATE class SET name = 'Abhijit' WHERE id = '5';
SAVEPOINT A;
INSERT INTO class VALUES(6, 'Chris');
SAVEPOINT B;
INSERT INTO class VALUES(7, 'Bravo');
SAVEPOINT C;
```

```
SELECT * FROM class;
```

12 Database Users and Administrators

People who work with a database can be categorized as database users or database administrators.

12.1 Database Users

There are four different types of database-system users, differentiated by the way they expect to interact with the system. Different types of user interfaces have been designed for the different types of users.

Naive users/Parametric users

Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a bank teller who needs to transfer \$50 from account A to account B invokes a program called transfer.

Application programmers

Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.

Sophisticated users

Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands. Analysts who submit queries to explore data in the database fall in this category. They use some of the tools like Online Analytical Processing(OLAP), Data mining.

Specialized users

Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

Casual end users

These are the users who occasionally access the database but they require different information each time. They use a sophisticated database query language basically to specify their request and are typically middle or level managers or other occasional browsers. For example: High level Managers who access the data weekly or biweekly.

12.2 Database Administrator

A person who has such central control of both the data and the programs that access those data over the system is called a database administrator (DBA). The functions of a DBA are the followings:-

- **Schema definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure and access-method definition:** DBA decides what structure to be used to store the data and what method to be used to access that.
- **Schema and physical-organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
- **Granting of authorization for data access:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
- **Routine maintenance:** Examples of the database administrator's routine maintenance activities are:
 - Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
 - Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.
 - Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

13 Database System Structure

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

13.1 Storage Manager

A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for storing, retrieving, and updating data in the database. The components of storage manager includes the following modules:

Authorization and integrity manager: It checks the integrity constraints and the authority of users to access data.

Transaction manager: It ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed

without conflicting.

File manager: It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

Buffer manager: It is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

The storage manager implements several data structures as part of the physical system implementation:

- **Data files**, which store the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices**, which provide fast access to data items that hold particular values.

13.2 Query Processor

The query processor includes the following components:-

DDL interpreter: It interprets DDL statements and records the definitions in the data dictionary.

DML compiler: It translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. The DML compiler also performs query optimization.

Query evaluation engine: It executes low-level instructions generated by the DML compiler.

14 Application Architectures

Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines, on which remote database users work, and server machines, on which the database system runs. Database applications are usually partitioned into two or three parts, as in Figure .

In a **two-tier architecture**, the application is partitioned into a component that resides at the client machine, which invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.

In contrast, in a **three-tier architecture**, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates

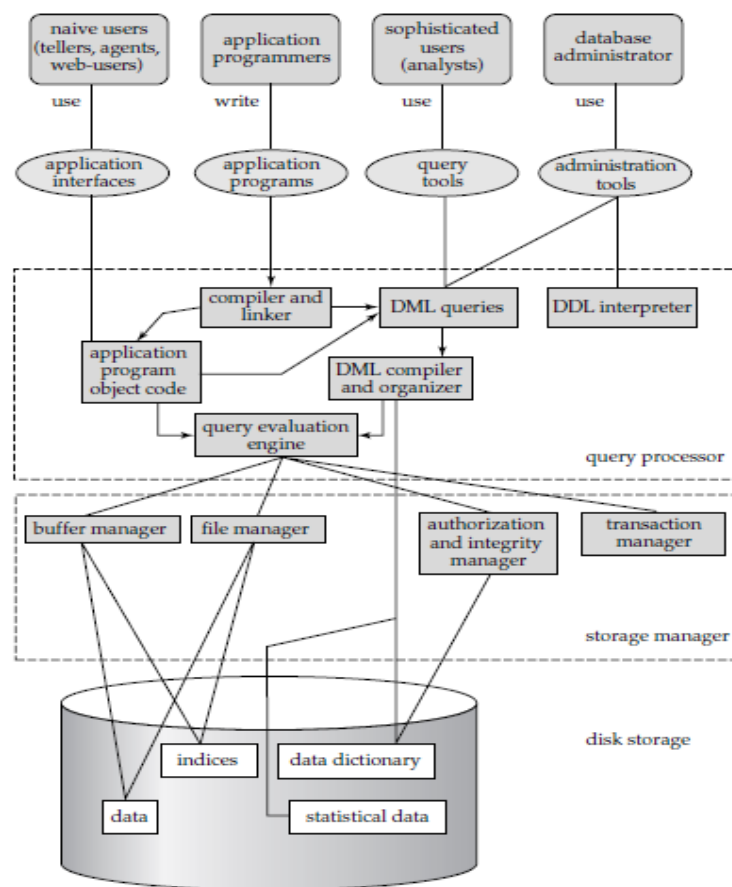


Figure 3: Database system structure

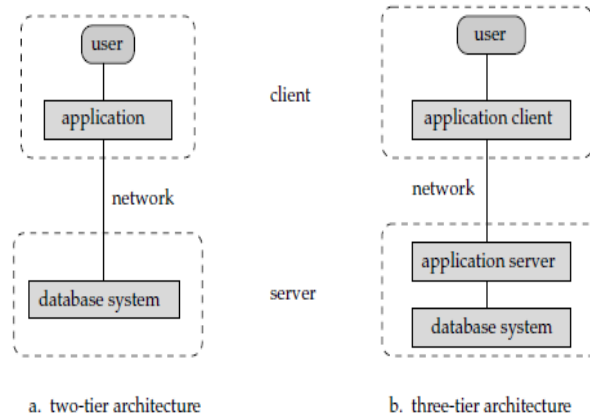


Figure 4: Two-tier and three-tier architectures

with an application server, usually through a forms interface. The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients.

Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.

ENTITY-RELATIONSHIP MODEL

1 E-R Model

ENTITY RELATIONSHIP (ER) MODEL is a high-level conceptual data model diagram. ER modeling helps you to analyze data requirements systematically to produce a well-designed database. The Entity-Relationship model represents real-world entities and the relationship between them. It is considered a best practice to make ER model before implementing your database.

The E-R data model uses three important components: entity sets, relationship sets, and attributes.

1.1 Entity Sets

An entity is a “thing” or “object” in the real world that is distinguishable from all other objects. For example, each person in an enterprise is an entity.

An entity set is a set of entities of the same type that share the same properties, or attributes. The set of all persons who are customers at a given bank, for example, can be defined as the entity set customer.

1.2 Attributes

Attributes are descriptive properties possessed by each member of an entity set. For example, possible attributes of the customer entity set are customer-id, customer-name, customer-street, and customer-city.

In real life, there would be further attributes, such as street number, apartment number, state, postal code, and country.

Possible attributes of the loan entity set are loan-number and amount.

For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute. The domain of attribute customer-name might be the set of all text strings of a certain length.

An attribute, as used in the E-R model, can be characterized by the following attribute types.

Simple and composite attributes

An attribute is said to be simple attribute if it can not be divided into parts. An attribute which can be divide into parts is said to be composite attribute.

For example, an attribute **name** and address are the composite attributes. An attribute **roll-number** is a simple attribute.

Single-valued and multi-valued attributes

The attribute for which we have a single value for each entity is called a single valued attribute and the attributes for which we have a set of values for a particular entity is

called a multi-valued attribute.

For example, loan-number, age are single valued attributes whereas mobile-no, dependent-name are multi-valued attributes.

Derived attribute

If the value for the attribute is derived from the values of other related attributes, then this attribute is said to be derived attribute.

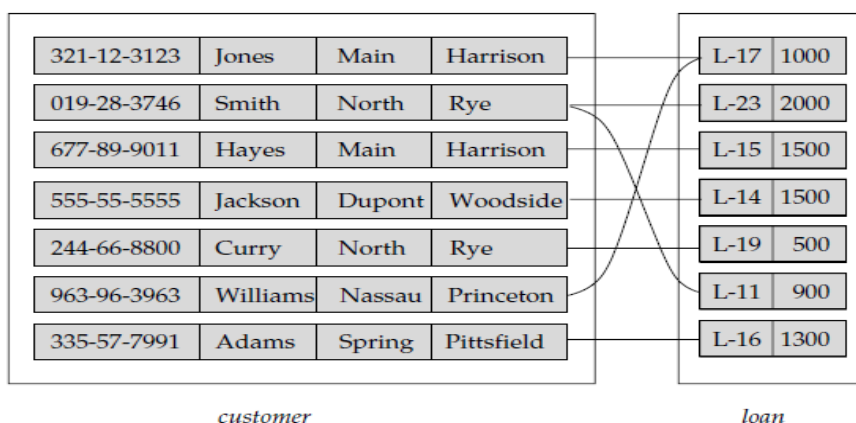
For example, Age attribute is derived from attribute date-of-birth, Experience attribute deried from date-of-joining etc. The value of the derived attribute is not stored but is computed when required.

Null values: An attributes takes null value when an entity does not have a value for it(for example middle-name) or if the value is not known.

1.3 Relationship set

- **A relationship** is an association among several entities.
- **For example,** we can define a relationship that associates customer Hayes with loan L-15. This relationship specifies that Hayes is a customer with loan number L-15.
- A relationship set is a set of relationships of the same type.
- It is a mathematical relation on more than one entity sets. If E_1, E_2, \dots, E_n are entity sets, then a relationship set R is a subset of $\{(e_1, e_2, \dots, e_n) | e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$ where (e_1, e_2, \dots, e_n) is a relationship. Here, entity sets E_1, E_2, \dots, E_n participate in relationship R.

Example: Consider the two entity sets customer and loan in following Figure. We define the relationship set borrower to denote the association between customers and the bank loans that the customers have.



Descriptive attributes: A relationship set may have some new attributes which are

not presents in the entity sets are called descriptive attributes. It describes about the relationship set.

Example: Consider a relationship set depositor with entity sets customer and account. We could associate the attribute access-date to that relationship to specify the most recent date on which a customer accessed an account.

Ternary relationship: The relationship between three entity sets is known as a ternary relationship. For example, the relationship works-on between the entity sets employee, branch and job can be a ternary relationship.

Degree of relationship: The number of entity sets that participate in a relationship set is known as degree of relationship.

2 Constraints

An E-R enterprise schema may define certain constraints to which the contents of a database must conform. In this section, we examine mapping cardinalities and participation constraints, which are two of the most important types of constraints.

2.1 Mapping cardinality

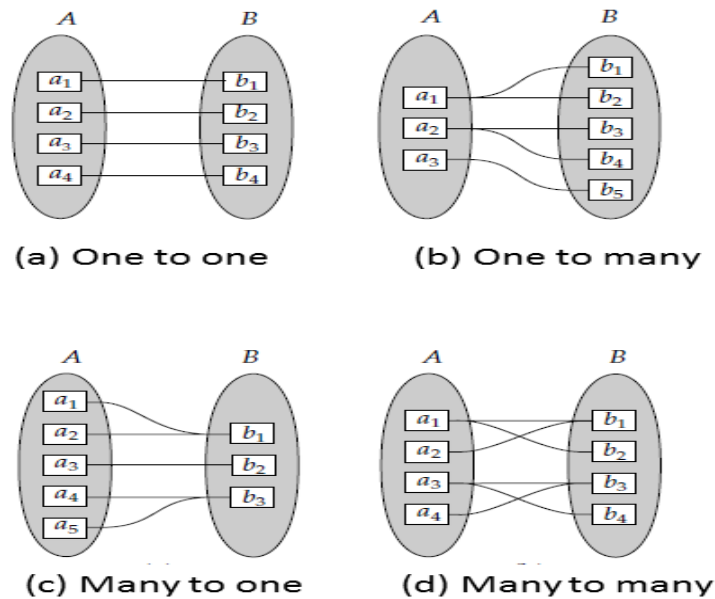
Mapping cardinalities express the number of entities to which another entity can be associated via a relationship set.

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

- **One to one:** An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
- **One to many:** An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.
- **Many to one:** An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.
- **Many to many:** An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

2.2 Participation Constraints

- The participation of an entity set E in a relationship set R is said to be **total** if every entity in E participates in at least one relationship in R.
- If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be **partial**.



Example: Consider relationship borrower between two entity sets customer and loan. Participation of loan entity set in this relationship borrower is the total. And participation of customer entity set in this relationship borrower is the partial.

3 Keys

A key allows us to identify a set of attributes that suffice to distinguish entities from each other. Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

3.1 Keys defined on entity sets

Superkey

A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set.

Candidate key

A superkey is said to be a candidate key if no proper subset of it is a superkey. In another words, a minimal superkeys are a candidate keys.

Primary key

Out of all possible candidate keys, the database designer chooses one as a primary key. That is, a primary key is a candidate key that is chosen by the database designer.

Alternate key

The candidate keys which are not selected as a primary key, are called alternate keys.

Composite key

Any key is said to be composite key if it consists of more than one attributes.

Example: Consider entity set student with attributes (rollNo, name, branch, address,

mobileNo).

Superkeys: $\{\text{rollNo}\}$, $\{\text{rollNo}, \text{name}\}$, $\{\text{rollNo}, \text{address}\}$, $\{\text{mobileNo}\}$, $\{\text{name}, \text{mobileNo}\}$.

Candidate keys: $\{\text{rollNo}\}$, $\{\text{mobileNo}\}$.

Primary key: $\{\text{rollNo}\}$

Alternate key: $\{\text{mobileNo}\}$.

Composite keys: $\{\text{rollNo}, \text{name}\}$, $\{\text{rollNo}, \text{address}\}$, $\{\text{name}, \text{mobileNo}\}$.

3.2 Relationship keys

Let R be a relationship set involving entity sets E_1, E_2, \dots, E_n . Let $\text{primary-key}(E_i)$ denote the set of attributes that forms the primary key for entity set E_i .

- If the relationship set R has no attributes associated with it, then the set of attributes $\text{primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \dots \cup \text{primary-key}(E_n)$ describes an individual relationship in set R .
- If the relationship set R has attributes a_1, a_2, \dots, a_m associated with it, then the set of attributes $\text{primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \dots \cup \text{primary-key}(E_n) \cup \{a_1, a_2, \dots, a_m\}$ describes an individual relationship in set R .

In both of the above cases, the set of attributes $\text{primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \dots \cup \text{primary-key}(E_n)$ forms a **superkey** for the relationship set.

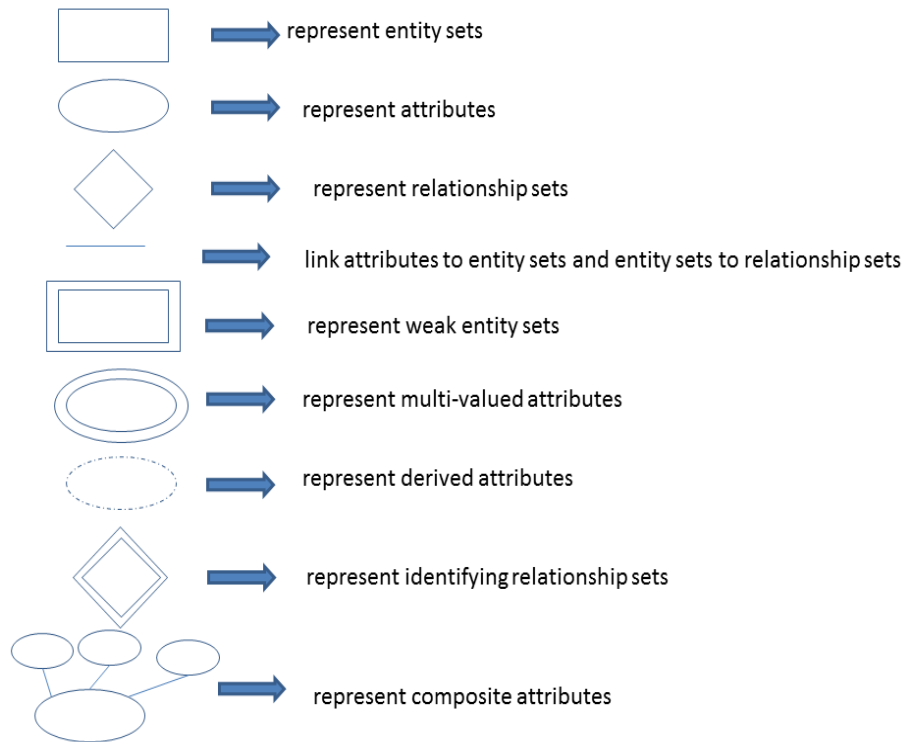
The structure of the **primary key** for the relationship set depends on the mapping cardinality of the relationship set.

Example: consider the entity sets customer and account, and the relationship set depositor, with attribute access-date.

- **Case 1:** If the relationship set is many to many. Then the primary key of depositor consists of the union of the primary keys of customer and account.
- **Case 2:** If the depositor relationship is many to one from customer to account, then the primary key of depositor will be the primary key of customer.
- **Case 3:** If the depositor relationship is many to one from account to customer, then the primary key of depositor will be the primary key of account.
- **Case 4:** If the depositor relationship is one to one from customer to account, then the primary key of depositor will be either the primary key of customer or the primary key of account.

4 Entity-Relationship Diagram

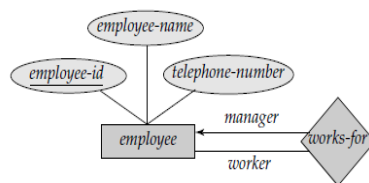
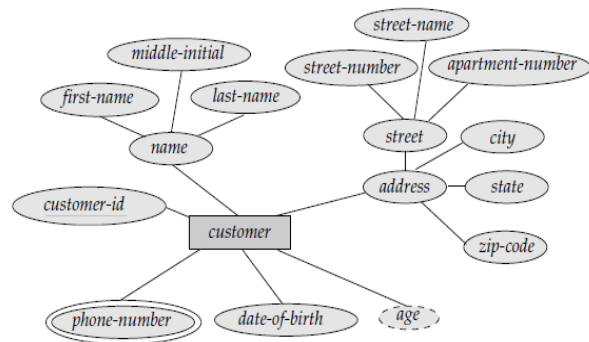
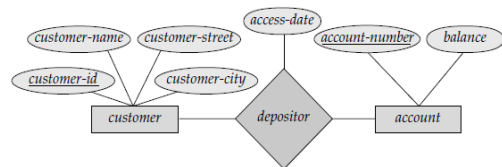
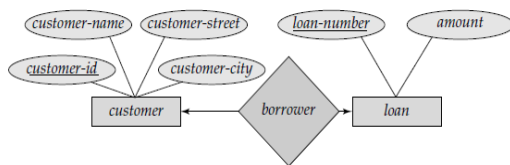
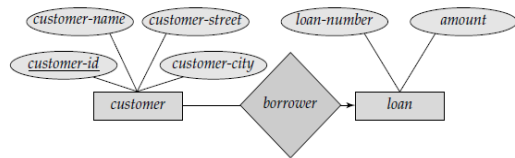
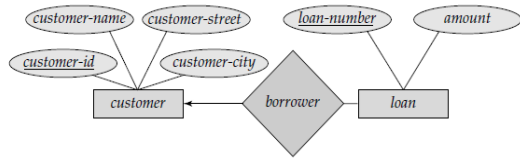
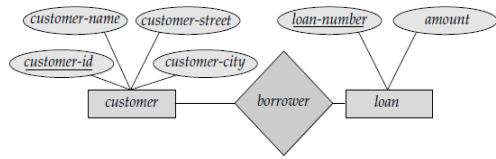
An E-R diagram can express the overall logical structure of a database graphically. Such a diagram consists of the following major components:



4.1 Some E-R diagram examples

1. E-R diagram showing many to many relationship between customer and loan entity set.
2. E-R diagram showing one to many relationship between customer and loan entity set.
3. E-R diagram showing many to one relationship between customer and loan entity set.
4. E-R diagram showing one to one relationship between customer and loan entity set.
5. E-R diagram showing descriptive attributes associated with a relationship set.
6. E-R diagram showing composite, multi-valued and derived attributes.
7. E-R diagram showing role indicator.
8. E-R diagram showing ternary relationship.

We permit at most one arrow out of a relationship set, since an E-R diagram with two or more arrows out of a non-binary relationship set can be interpreted in two ways. Suppose there is a relationship set R between entity sets A_1, A_2, \dots, A_n , and the only arrows are on the edges to entity sets $A_{i+1}, A_{i+2}, \dots, A_n$. Then, the two possible interpretations are:



- (a) A particular combination of entities from A_1, A_2, \dots, A_i can be associated with at most one combination of entities from $A_{i+1}, A_{i+2}, \dots, A_n$. Thus, the primary key for the relationship R can be constructed by the union of the primary keys of A_1, A_2, \dots, A_i .
 - (b) For each entity set $A_k, i < k \leq n$, each combination of the entities from the other entity sets can be associated with at most one entity from A_k . Each set $\{A_1, A_2, \dots, A_{k-1}, A_{k+1}, \dots, A_n\}$, for $i < k \leq n$, then forms a candidate key.
9. E-R diagram showing total participation.
10. E-R diagram showing alternative notation for cardinality limits.
- An edge between an entity set and a binary relationship set can have an associated minimum and maximum cardinality, shown in the form l..h, where l is the minimum and h the maximum cardinality.
 - A minimum value of 1 indicates total participation of the entity set in the relationship set.
 - A maximum value of 1 indicates that the entity participates in at most one relationship, while a maximum value * indicates no limit.
 - Note that a label 1..* on an edge is equivalent to a double line.

5 Weak Entity Sets

- An entity set that does not have a primary key is referred to as a **weak entity set**.
- An entity set that has a primary key is termed as a strong entity set.
- The existence of weak entity set depends on the existence of an **identifying or owner entity set**.
- The relationship associating the weak entity set with identifying entity set is called an **identifying relationship set**.
- The identifying relationship is many to one from the weak entity set to the identifying entity set, and the participation of the weak entity set in the relationship is total.
- The **discriminator** of a weak entity set is a set of attributes that distinguishes among all the entities in weak entity set that depends on one strong entity set. It is also said to be **partial key**.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is dependent plus the weak entity set's discriminator.
- Weak entity set is represented by double rectangle.
- The discriminator of a weak entity set is shown with dashed lines.

- The identifying relationship set is represented by double diamonds

In this E-R diagram, payment is a weak entity set and loan is a strong entity set. loan-payment is an identifying entity set. Discriminator of payment is payment-number. Primary key of payment will be {loan-number,payment-number}.

6 Extended E-R Features

In this section, we discuss the extended E-R features of specialization, generalization, higher- and lower-level entity sets, attribute inheritance, and aggregation.

6.1 Specialization

The process of designating subgroupings within an entity set is called specialization.

Example: Consider an entity set person, with attributes name, street, and city. A person may be further classified as one of the following:

- customer
- employee

Each of these person types is described by a set of attributes that includes all the attributes of entity set person plus possibly additional attributes. For example, customer entities may be described further by the attribute customer-id, whereas employee entities may be described further by the attributes employee-id and salary.

In terms of an E-R diagram, specialization is depicted by a triangle component labeled ISA, as shown in the following figure. The label ISA stands for "is a". The ISA relationship may also be referred to as a superclass-subclass relationship. Higher- and lower-level entity sets are depicted as regular entity sets that is, as rectangles containing the name of the entity set.

6.2 Generalization

- Generalization is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets.
- Generalization is a bottom-up design process that combines the number of entity sets that share the same features into higher level entity sets.
- Generalization and specialization are simple inversions of each other. They are represented in an E-R diagram in the same way.
- In E-R diagram, generalization is also represented by triangle symbol "ISA" just like specialization.
- Same E-R diagram is used to represent specialization and generalization.

Example: In generalization process, database designer may have first identified a customer entity set with the attributes name, street, city, and customer-id, and an employee entity set with the attributes name, street, city, employee-id, and salary.

There are similarities between the customer entity set and the employee entity set in the sense that they have several attributes in common. This commonality can be expressed by generalization.

6.3 Aggregation

- Aggregation is an abstraction through which relationships are treated as a higher level entity sets and can participate in relationships.
- Aggregation allows us to indicate that a relationship set participates in another relationship sets.

Example: Consider the following E-R diagram:- There is redundant information in the above figure, however, since every employee, branch, job combination in manages is also in works-on. If the manager were a value rather than an manager entity, we could instead make manager a multi-valued attribute of the relationship works-on.

Using aggregation, the relationship set works-on (relating the entity sets employee, branch, and job) is treated as a higher-level entity set called works-on. Such an entity set is treated in the same manner as is any other entity set. We can then create a binary relationship manages between works-on and manager to represent who manages what tasks. It is shown in the following figure:- **Example:**

Draw the E-R model or diagram for banking enterprise.

Solution: To design E-R model corresponding to any enterprise, we follow the following steps:-

- **Data requirements**
- **Entity Sets Designation**
- **Relationship Sets Designation**
- **Identify attributes**
- **Identify mapping cardinality of each relationship**

E-R diagram for banking enterprise is the following:-

7 Reduction of an E-R Schema to Tables

Every E-R schema or diagram can be converted into set of tables. In this section, we describe how an E-R schema can be represented by tables. The constraints specified in an E-R diagram, such as primary keys and cardinality constraints, are mapped to constraints on the tables generated from the E-R diagram.

7.1 Tabular Representation of Strong Entity Sets

Let E be a strong entity set with descriptive attributes a_1, a_2, \dots, a_n . We represent this entity by a table called E with n distinct columns, each of which corresponds to one of the attributes of E . Each row in this table corresponds to one entity of the entity set E .

Example:

Consider a strong entity set **loan** with attributes loan-number and amount. The table corresponding to loan will be the following

7.2 Tabular Representation of Weak Entity Sets

Let A be a weak entity set with attributes a_1, a_2, \dots, a_m . Let B be the strong entity set on which A depends. Let the primary key of B consist of attributes b_1, b_2, \dots, b_n . We represent the entity set A by a table called A with one column for each attribute of the set:

$$\{a_1, a_2, \dots, a_m\} \cup \{b_1, b_2, \dots, b_n\}$$

Example:

Consider the weak entity set payment with three attributes: payment-number, payment-date, and payment-amount. It depends on strong entity set loan. The primary key of loan is loan-number. Thus, we represent payment entity set by a table with four columns labeled loan-number, payment-number, payment-date, and payment-amount. The table corresponding to payment will be the following

7.3 Tabular Representation of Relationship Sets

Let R be a relationship set, let a_1, a_2, \dots, a_m be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R , and let the descriptive attributes (if any) of R be b_1, b_2, \dots, b_n . We represent this relationship set by a table called R with one column for each attribute of the set:

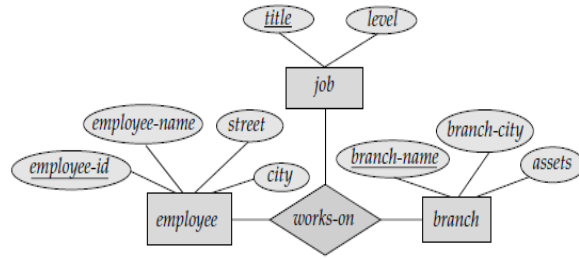
$$\{a_1, a_2, \dots, a_m\} \cup \{b_1, b_2, \dots, b_n\}$$

Example:

Consider the relationship set borrower. This relationship set involves the following two entity sets:

- customer, with the primary key customer-id
- loan, with the primary key loan-number

Assume relationship set borrower has no attributes. Then, the borrower table has two columns, labeled customer-id and loan-number. Borrower table is the following:-



loan-number	amount
L-1	10000
L-2	15000
L-3	20000

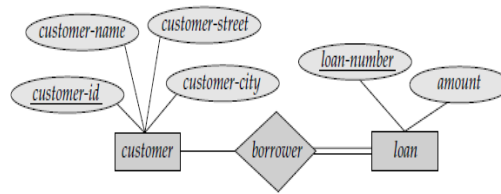
Table 1: loan table

loan-number	payment-number	payment-date	payment-amount
L-1	1	9 May 20018	5000
L-2	3	29 May 20018	6000
L-2	4	20 June 20018	9000

Table 2: payment table

customer-id	loan-number
C-34	L-1
C-45	L-2
C-20	L-3

Table 3: borrower table



7.4 Tabular Representation corresponding to Composite Attributes

We handle composite attributes by creating a separate attribute for each of the component attributes; we do not create a separate column for the composite attribute itself.

Example:

Suppose address is a composite attribute of entity set customer, and the components of address are street and city. The table generated from customer would then contain columns address-street and address-city; there is no separate column for address.

7.5 Tabular Representation corresponding to Multivalued Attributes

For a multivalued attribute M, we create a table T with a column C that corresponds to M and columns corresponding to the primary key of the entity set or relationship set of which M is an attribute.

Example:

Consider the above E-R diagram for banking enterprise. The diagram includes the multivalued attribute dependent-name. For this multivalued attribute, we create a table dependent-name, with columns dname, referring to the dependent-name attribute of employee, and employee-id, representing the primary key of the entity set employee. Each dependent of an employee is represented as a unique row in the table.

7.6 Tabular Representation of Generalization

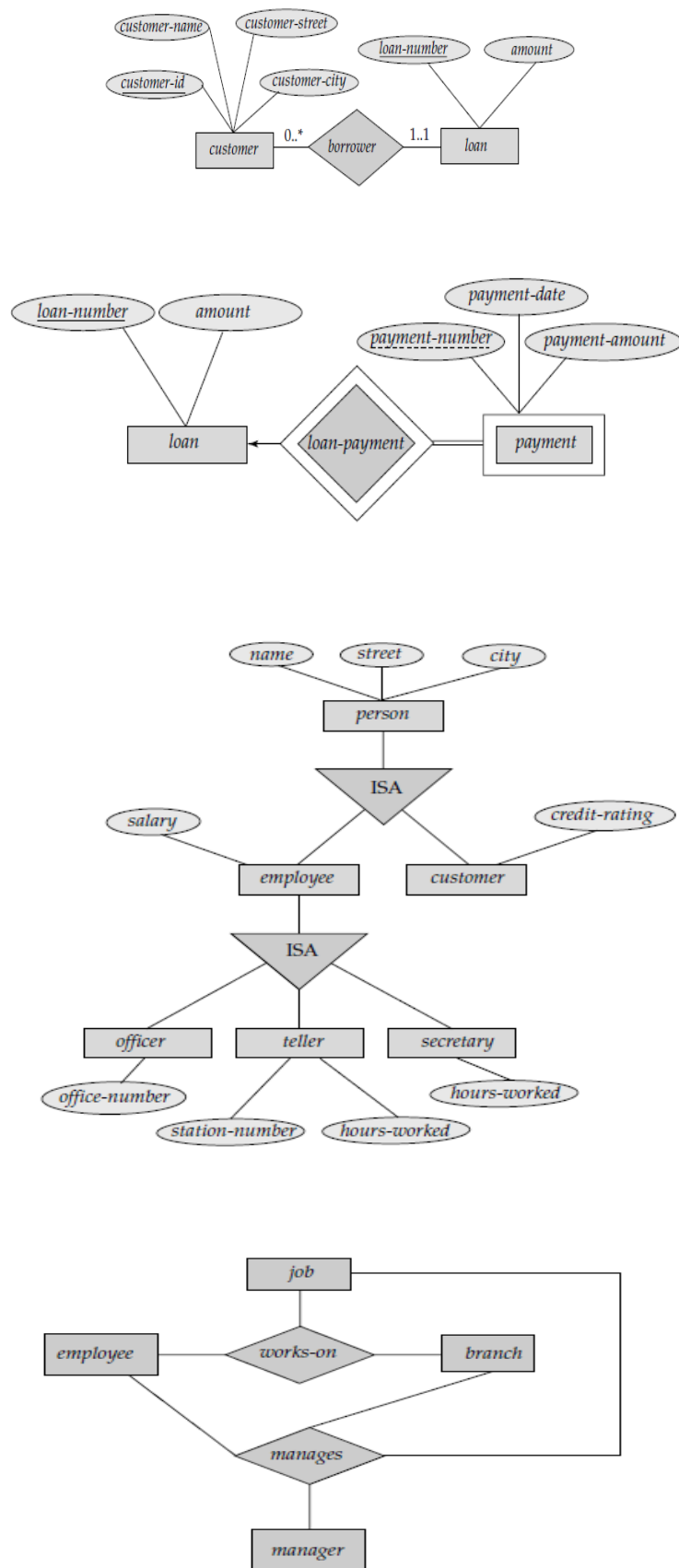
There are two different methods for transforming to a tabular form an E-R diagram that includes generalization. Consider the following E-R diagram:-

We simplify it by including only the first tier of lower-level entity sets, that is, savings-account and checking-account.

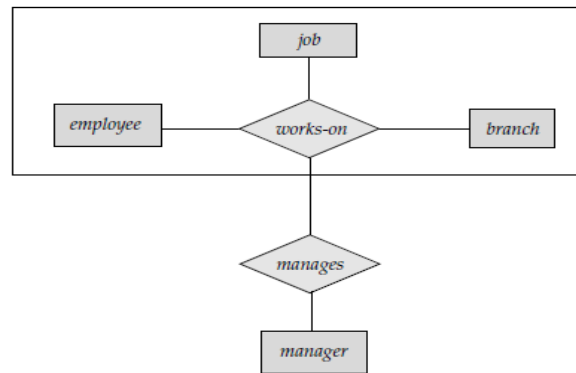
1. Create a table for the higher-level entity set. For each lower-level entity set, create a table that includes a column for each of the attributes of that entity set plus a column for each attribute of the primary key of the higher-level entity set.

For the above E-R diagram, we have three tables:

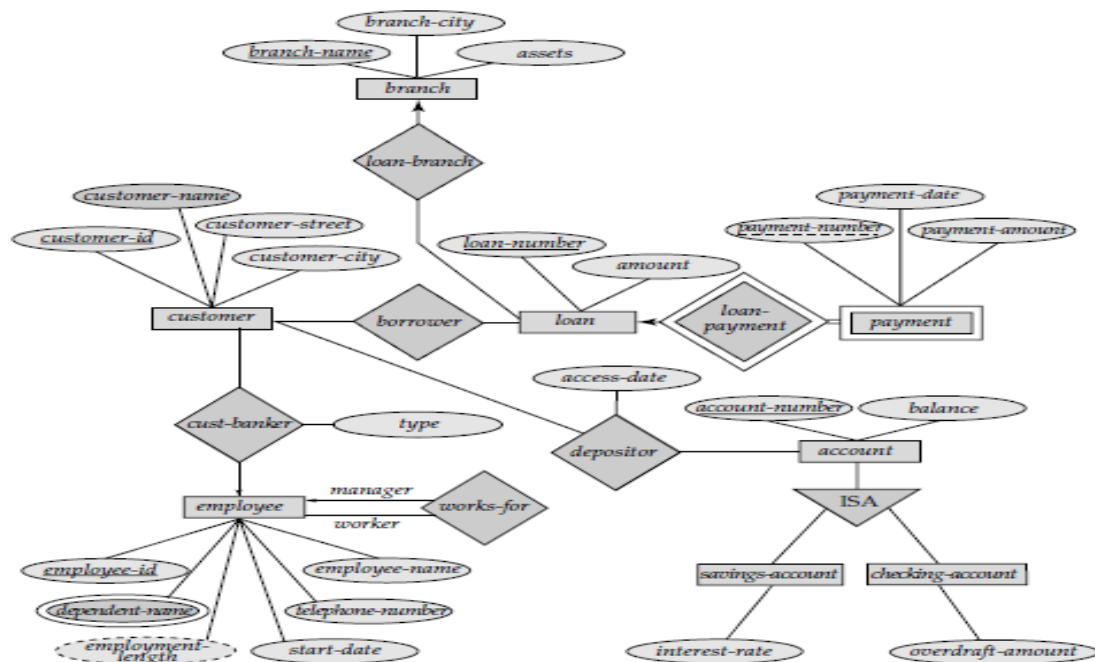
- account, with attributes account-number and balance
- savings-account, with attributes account-number and interest-rate
- checking-account, with attributes account-number and overdraft-amount



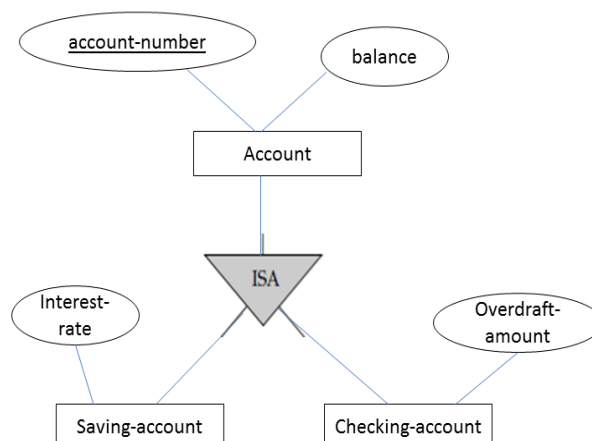
E-R diagram with redundant relationships



E-R diagram with aggregation



E-R diagram for banking enterprise



2. If the generalization is disjoint and complete, that is, if no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher level entity set is also a member of one of the lower-level entity sets. Here, do not create a table for the higher-level entity set. Instead, for each lower-level entity set, create a table that includes a column for each of the attributes of that entity set plus a column for each attribute of the higher-level entity set.

For the above E-R diagram, we have two tables:

- savings-account, with attributes account-number, balance, and interest-rate
- checking-account, with attributes account-number, balance, and overdraft-amount

7.7 Tabular Representation of Aggregation

Transforming an E-R diagram containing aggregation to a tabular form is straightforward. Consider the diagram in the following figure:-

The table for the relationship set manages between the aggregation of works-on and the entity set manager includes a column for each attribute in the primary keys of the entity set manager and the relationship set works-on. It would also include a column for any descriptive attributes, if they exist, of the relationship set manages. We then transform the relationship sets and entity sets within the aggregated entity.

8 Exercise

1. Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.
2. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.
3. A university registrar's office maintains data about the following entities:
 - (a) **courses**, including number, title, credits, syllabus, and prerequisites;
 - (b) **course offerings**, including course number, year, semester, section number, instructor(s), timings, and classroom;
 - (c) **students**, including student-id, name, and program; and
 - (d) **instructors**, including identification number, name, department, and title.

Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.

4. Construct appropriate tables for each of the E-R diagrams in Exercises 1 to 3.
- 5.

8.1 Solution of Exercise

1. An E-R diagram for a car-insurance company is the following:-
2. An E-R diagram for a hospital with a set of patients and a set of medical doctors is the following:-
3. An E-R diagram for the registrar's office is the following:-
The assumptions made are :
 - (a) A class meets only at one particular place and time. This E-R diagram cannot model a class meeting at different places at different times.
 - (b) There is no guarantee that the database does not have two classes meeting at the same place and time.
4. Appropriate tables for each of the E-R diagrams used in exercises 1 to 3 are the following:-

(1) Car insurance tables:

person (driver-id, name, address)

car (license, year, model)

accident (report-number, date, location)

participated(driver-id, license, report-number, damage-amount)

(2) Hospital tables:

patients (patient-id, name, insurance, date-admitted, date-checked-out)

doctors (doctor-id, name, specialization)

test (testid, testname, date, time, result)

doctor-patient (patient-id, doctor-id)

test-log (testid, patient-id)

performed-by (testid, doctor-id)

(3) University registrar's tables:

student (student-id, name, program)

course (courseno, title, syllabus, credits)

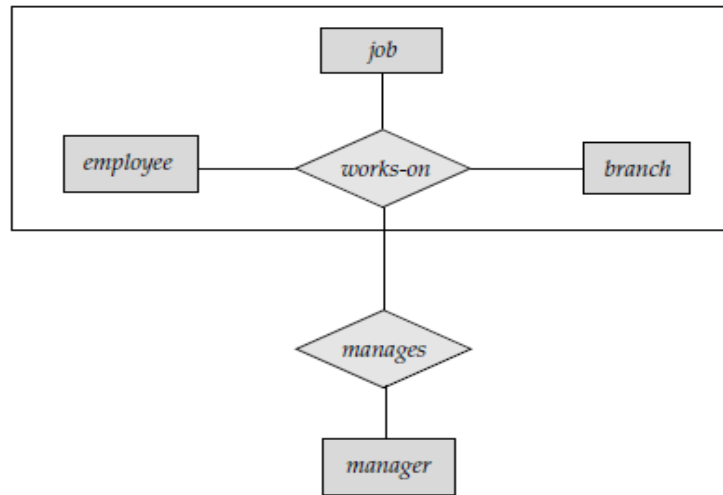
course-offering (courseno, secno, year, semester, time, room)

instructor (instructor-id, name, dept, title)

enrols (student-id, courseno, secno, semester, year, grade)

teaches (courseno, secno, semester, year, instructor-id)

requires (maincourse, prerequisite)



E-R diagram with aggregation

