U20CSCJ0A - DATABASE INFORMATION SYSTEMS

UNIT 1 - Database Information Systems.

Overview of Database Information Systems - Database system concepts and Architecture - Entity Relationship Model: Types of Attributes - Relationship - Data Modelling.

1. DATABASE INFORMATION SYSTEMS OVERVIEW:

* A database is a collection of related data. A database system is basically a computerized record-keeping system.

* It is a repository or container for a collection of computerized data files. Users of the system can perform a variety of operations involving such files - for example

- Adding new files to the database.
- Inserting data into existing files
- Retrieving data from existing files
- Deleting data from existing files
- Changing data in existing files
- Removing existing files from the database.

1.1 What is a database Information System?

A database Information system is a collection of programs that enables users to create and maintain a database.

Its a general purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.

1.2 Example of database

* Let us consider the university database for maintaining information concerning student, courses and grades in a university environment.

* The following figure 2 shows the database structure and some sample data for such a database. The database is organized as four files, each of which stores data records of same type.

* The STUDENT file stores data on each student. The COURSE file stores data on each course, the SECTION file stores data on each section of a course, the GRADE-REPORT file stores the grades that student receive in the various sections they have completed.

STUDENT

Name	Student_number	class	Major

COURSE

Course-name	Course_number	Credit-hours	Department

SECTION

Section-Identifier	Course-number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section-identifier	Grade

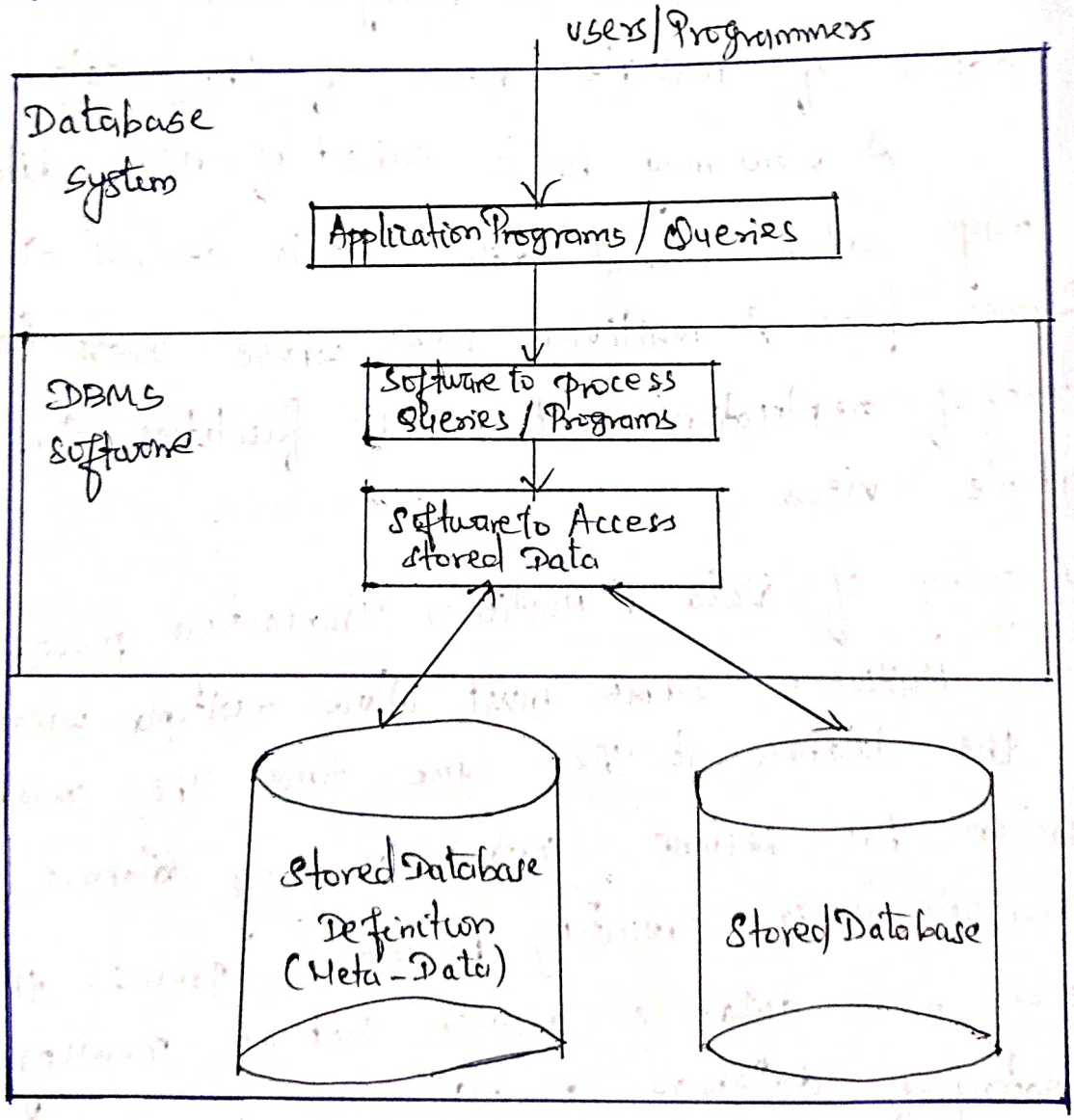
1.3 A simplified database system environment.

* Defining a database involves specifying the data types, structures & constraints of the data to be stored in the database.

* Constructing the database is the process of storing the data on some storage medium that is controlled by DBMS.

* Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes.

* Sharing a database allows multiple users and programs to access the database.



1.4 Characteristics of the Database Approach:

1. Self Describing nature of a Database system.

The definition is stored in the DBMS catalog which contains information such as the structure of each file, the type & storage format of each data item. The information stored in the catalog is called metadata.

2. Relationship between programs & Data, Data Abstraction.

A database gives a logical relationship between its records and data. So a user can access various records depending upon the logical conditions by a single query from the database. The structure of data files is stored in the DBMS catalog separately from the access programs. This property is called program data Indepen

3. Support of multiple views of the data.

A view may be a subset of the database or it may contain virtual data that is derived from the database files. A multiuser DBMS whose users have a variety of applications must provide facilities for defining multiple views.

4. Sharing of Data & Multiuser Transaction processing.

Multiuser DBMS must allow multiple users to access the database at the same time. The concept of a transaction has become central to many database applications. A transaction is an executing program or process that includes one or more database accesses, such as reading or updating of database records.

1.5 Advantages of using the DBMS Approach

1. controlling redundancy.
2. Restricting unauthorized access.
3. Providing Persistent storage for Program Objects
4. Providing storage structures for efficient Query processing.
5. Providing Backup & Recovery.
6. Providing multiple user Interface.

II. Database System Concepts & Architecture:

2.1 Database system concepts:

This chapter discusses the data models and defines the concepts of schemas.

DATA MODELS, SCHEMAS & INSTANCES

One fundamental characteristics of the database approach is that it provides level of data abstraction.

Data abstraction refers to the details of data organization and storage as well as the features for understanding the d

A data model is a collection of concepts that can be used to describe the structure of a database provides the necessary details to achieve the abstraction.

Categories of data model.

- High-level or conceptual data model
- low level or physical data model
- Entity Relationship model
- Network & hierarchial models.
- object data model.

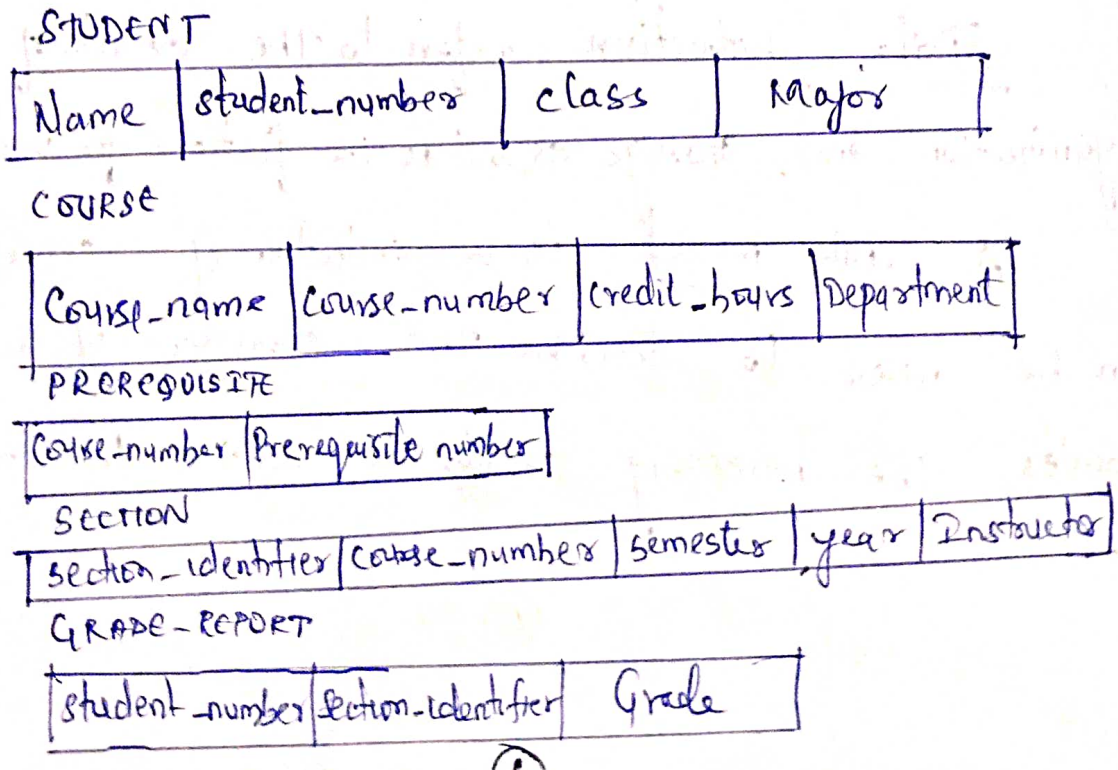
In any model, its important to differentiate the description of the database and the database itself.

The description of the database is called database schema which is specified during database design & should not be changed frequently.

A displayed schema is called schema diagram

The following figure shows schema diagram of student database given in the previous topic.

We represent each object in the schema such as STUDENT or COURSE - a schema construct



2.2 THREE SCHEMA ARCHITECTURE

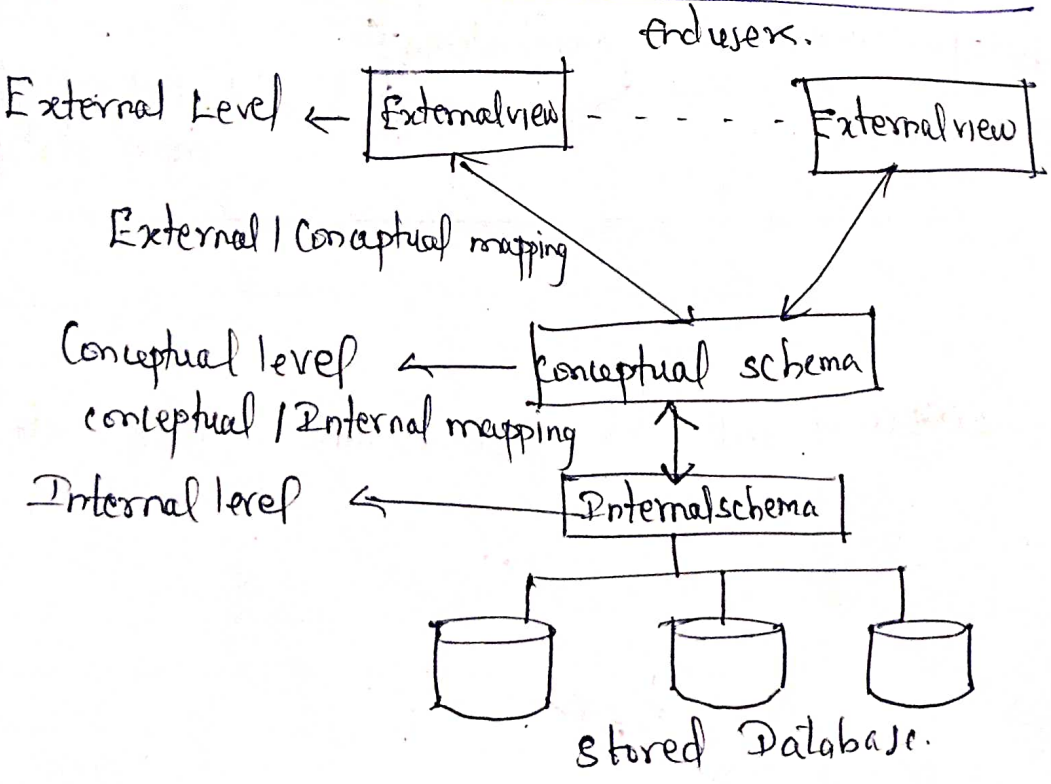
In this architecture, schemas can defined at the following 3 levels, this is used to separate the user application and the physical database.

1- The internal level has an internal schema which describes the physical stage structure of the database. This schema uses a physical data model & describes the complete details of data storage & access paths for the DB.

2. The conceptual level has a conceptual schema which describes the structure of the whole database for users. This schema hides the details of physical storage structures & concentrate on designing entities, datatypes, relationships, user operations and constraints.

3. The external or new level inclued a number of external schemas or views. Each external schema describes the part of the DB that aparticular user group is interested in & hides rest of DB's.

Figure : Three-schema architecture.



2.3 THE DATA BASE SYSTEM ENVIRONMENT

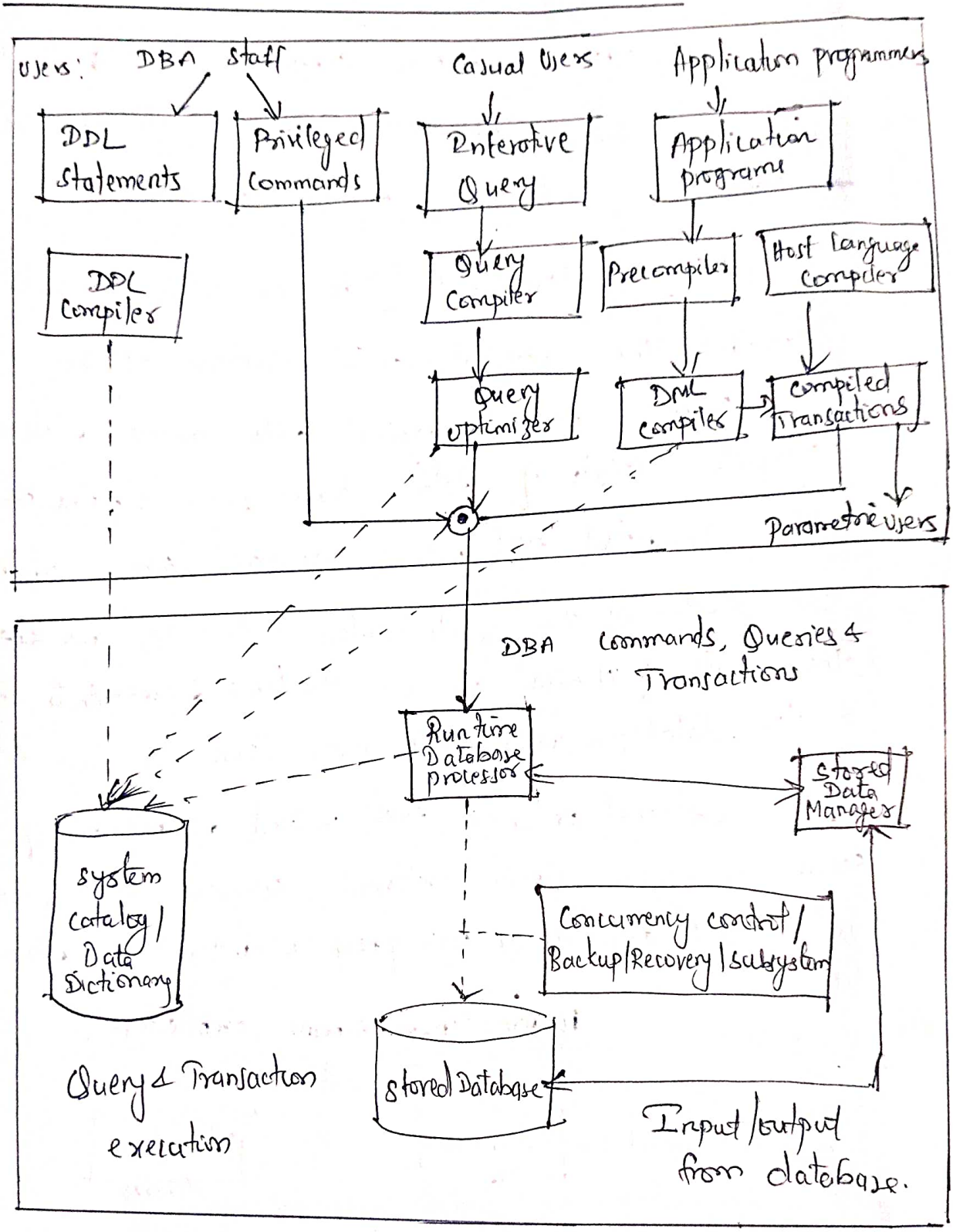


Figure: Component modules of a DBMS & their interactions.

→ The above figure illustrates a typical DBMS component. It is divided into 2 halves. The top half of the figure refers to the various users of the DB environment and their interfaces. The lower half represents the internal of the DB responsible for storage of data & processing of transactions.

* The first half shows interfaces for the DBA staff who work with interactive interfaces to formulate queries, application programmers, parametric users who do data entry work by supplying parameters to predefined transactions.

* The DDL compiler processes schema definitions, specified in the DDL and description of schema in DB catalog.

* casual users and persons with occasional need for information from the database intract using some form of interface

* The Query optimizer is concerned with rearrangement and possible reordering of operations, elimination of redundancies and use of correct algorithm and indexes during execution.

* precompiler extracts DML commands from an application program written in a host programming language. These commands are send to the DML compiler for compilation into object code for database access.

* The second half shows storage of data processing transaction and database system utilities.

Common utilities have the following type of functions.

→ Loading:

A loading utility is used to load existing data files such as text files or sequential files.

→ Backup

A backup utility creates a backup copy of the database, by dumping the entire database into tape. The backup copy can be used to restore the database in case of any failure.

→ Database storage organization.

This utility can be used to reorganize a set of database files into a different file organizations to improve performance.

→ Performance monitoring.

Such a utility monitors database usage & provides statistics to DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files or whether to add or drop indexes to improve performance.

→ The runtime database processor is used to execute privileged commands, the executable query plans and transaction with runtime parameters. It works with the system dictionary and may update it with statistics.

Entity - Relationship Model

* The Entity-relationship (E-R) data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.

* The E-R data model is one of several semantic data models.

* The E-R Model is very useful in mapping the meaning & interactions of real-world enterprises into a conceptual schema.

The E-R Model employs three basic notations. They are

1. Entity Sets
2. Relationship Sets and
3. Attributes.

1. Entity Sets:-

An entity is a "thing" or "object" in the real-world that is distinguishable from all other objects.

eg:- Each person in an enterprise is an entity.

An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.

eg:- A person may have a person_id property whose value uniquely identifies that person.

An entity may be concrete, such as person or a book, or it may be abstract, such as a loan, or a concept.

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

Entity sets do not need to be disjoint. eg:- It is possible to define the entity of all employees of a bank (Employee) and the entity set of all the customers of bank (customer). A person entity may be an employee entity, a customer entity, both, or neither.

An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.

eg:- possible attribute of the loan entity set are loan number & amount.

Each entity has a value for each of its attributes. eg: a particular customer entity may have the values 112-214 for customer_id, the value Hari for customer_name.

A database thus includes a collection of entity sets, each of which contains any no. of entities of the same type.

The following figure shows part of a bank database that consist of two entity set: customer & loan

112-214	Hari	Tambaram
315-227	Vijay	Selaiyur
422-731	Karthick	K.K. Nagar

customer

L-17	10000
L-23	7000
L-11	9000

Loan

2. Relationship Sets :-

A relationship is an association among several entities. eg:- we can define a relationship that associates customer vijay with loan L-11. This relationship specifies that Vijay is a customer with loan number L-11.

A relationship set is an set of relationships of the same type. Formally, it is a mathematical relation on $n \geq 2$ (possibly nondistinct) entity sets.

If $E_1, E_2 \cdots E_n$ are entity sets, then a relationship set R is a subset of

$$\{(e_1, e_2, \cdots e_n) \mid e_1 \in E_1, e_2 \in E_2 \cdots e_n \in E_n\}$$

where $(e_1, e_2, \cdots e_n)$ is a relationship.

Consider the two entity sets customer & loan in Previous fig. We define the relationship set borrower to denote the association between a bank loans and customer of that loans.

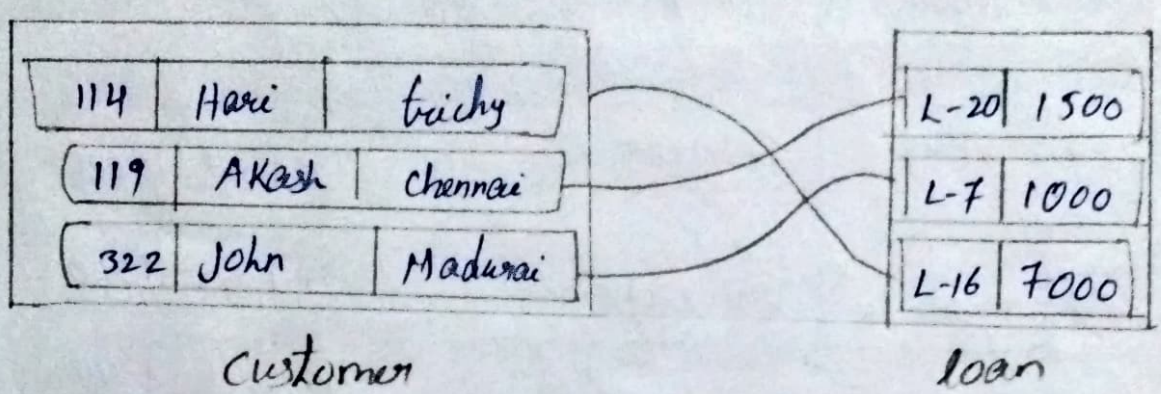


Fig:- Relationshipset borrower

The association between entity set is referred to as participation; that is, the entity sets $E_1, E_2, \ldots E_n$ participate in relationship set R.

A relationship instance in an E-R schema represents an association between the named entities in the real-world enterprise that is being modeled.

The function that an entity plays in a relationship is called the entity's role, Roles are implicit and are not usually specified.

In one type of relationship set, called recursive relationship set, explicit role names are necessary to specify, if the same entity set participate is a relationship set more than once, in different roles.

A relationship may also have attributes called descriptive attributes.

The following figure shows the relationship set depositor with a descriptive attributes access_date ; to keep the figure simple, only some attributes of the 2 entity sets are shown.

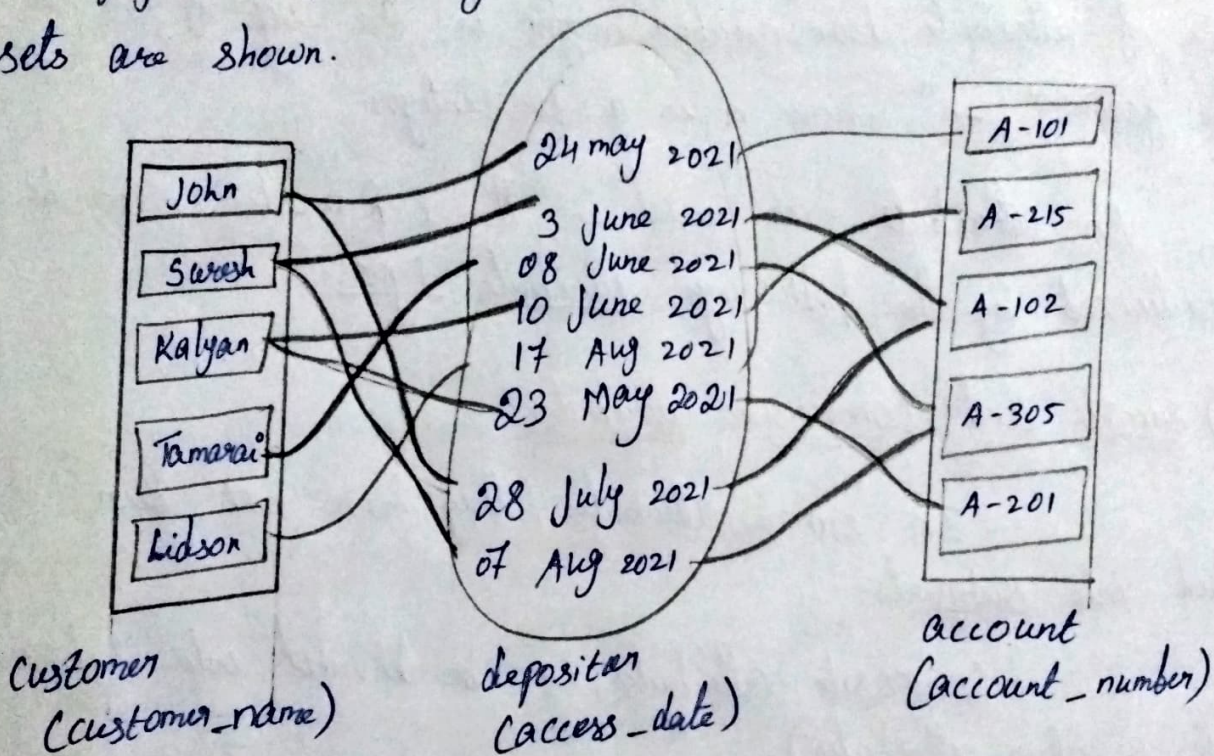


Fig:- Access_date as Attribute

The relationship set borrower & loan_branch provide an example of a binary relationship set - that is, one that involve two entity sets.

The number of entity sets that participate in a relationship set is also the degree of the relationship set.

3. Attributes:-

There is a set of permitted values for each attribute called as 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. similarly, the domain of attribute loan_number might be the set of all strings of the form "L-n" where n is a +ve integer.

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

(i) simple and Composite attributes:-

In simple attributes, they have not been divided into subparts.

Composite attributes, can be divided into subparts (that is, other attributes).

eg:- An attribute name could be structured as a composite attribute consisting of first_name, middle_initial, and last_name.

Note also that a composite attribute may appear as a hierarchy. In the composite attribute address, its component attribute street can be further divided into street_number, street_name & Door_number.

The figure is an examples of composite attributes for the customer entity set.

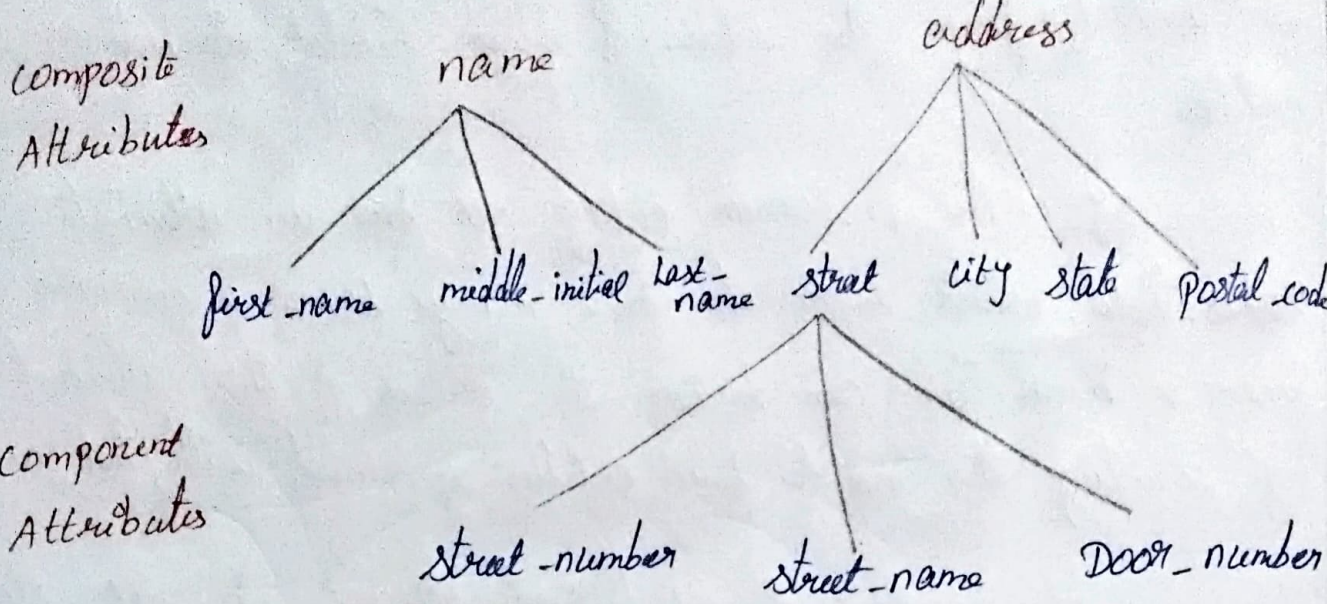


Fig:- Composite attributes customer_name & customer_address

(ii) Single-valued & Multivalued attributes:-

The attributes in our examples all have a single value for a particular entity. For instance, the loan_number attribute for a specific loan entity refers to only one loan number. Such attributes are said to be single valued.

An attribute can have set of values for a specific entity. Consider an employee entity set with the attribute phone_number. An employee may have zero, one, or several phone numbers & different employee may have different no. of phones. This type of attribute is said to be multivalued.

(iii) Derived Attributes:-

The value for this type of attribute can be derived from the values of other related attributes or entities

eg:- The customer entity set has an attribute loans_held, which represents how many loans a customer has from a bank. we can derive the values of this attribute by counting the no. of. loan entities associated with that customer.

The value of a derived attribute is not stored but is computed when required.

An attribute takes a null values when an entity does not have a value for it. The null value may indicate "not applicable" - that is, the value does not exist for the entity.

Mapping Cardinalities

An E-R enterprise schema may define certain constraints to which the contents of a database must conform. Mapping cardinalities is one of the constraints.

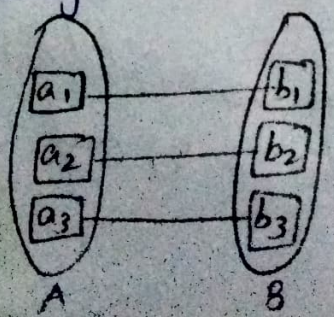
Mapping cardinalities, or cardinality ratios, express the no. of entities to which another entity can be associated via a relationship set.

It is most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than 2 entity sets.

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

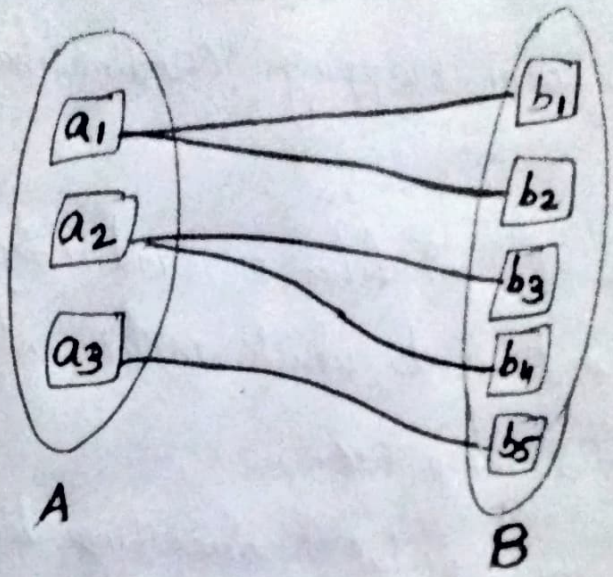
* One - to - one :-

An entity A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



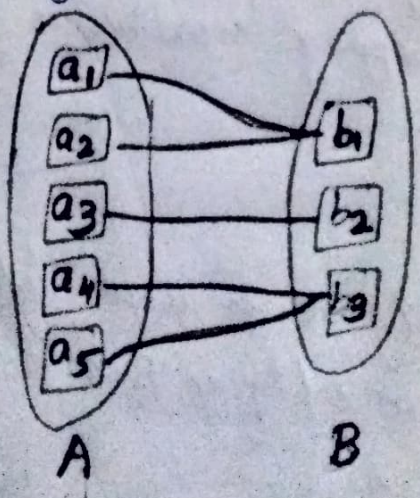
* One-to-Many:-

An entity in A is associated with ~~at~~ any (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:-

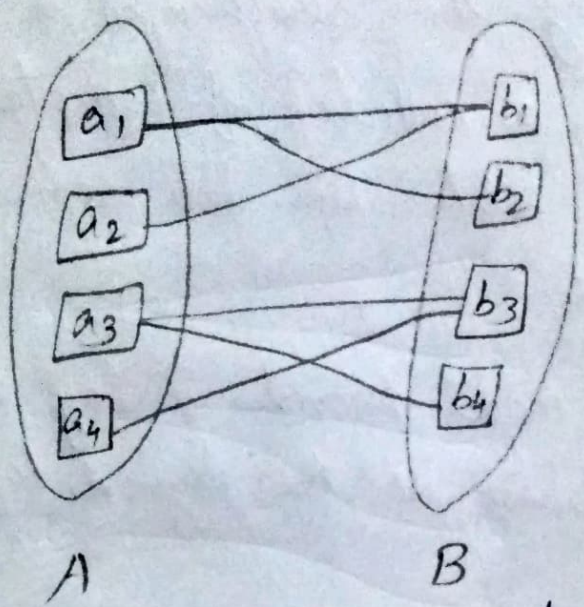
An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any no. of (zero or more) entities in A.



*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.



The appropriate mapping coordinality for a particular relationship set obviously depends on the real-world situation that the relationship set is modeling.

Eg:- Consider the borrower relationship set. If, in a particular bank, a loan can belong to only one customer, and a customer can have several loans, then the relationship set from customer to loan is one-to-many.

If a loan can belong to several customers (jointly by several business partners), the relationship set is many-to-many.

Data Models:-

Underlying the structure of a database is the data model: a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.

A data model provides a way to describe the design of a database at the physical, logical, and view level.

It is classified into 4 categories. they are Relational model, The E-R Model, Object-Based Data model, semistructured Data Model.

* Enity-Relationship Diagrams

An E-R diagram can express the overall logical structure of a database graphically.

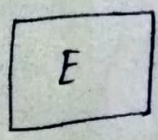
E-R diagrams are simple and clear-qualities that may well account in large part for the widespread use of the E-R model.

such a diagram consists of the following major components:-

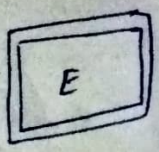
- Rectangle - Represent entity sets
- Ellipses - Represent attributes

- Diamonds - Represent relationship sets
- Lines - which link attribute to entity sets & entity sets to relationship sets.
- Double ellipses - Multivalued attributes
- Dashed ellipses - Derived attributes
- Double lines - Total participation of an entity in a relationship set
- Double rectangle - weak entity sets.

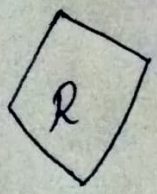
symbols used in the E-R notation



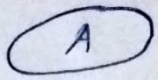
entity set



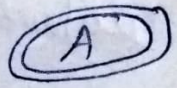
weak entity set



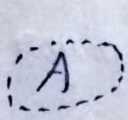
relationship set



attribute

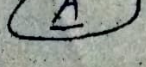


multivalued attribute

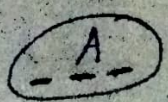


derived attribute

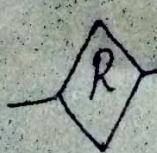
Identifying relationship set for weak entity set



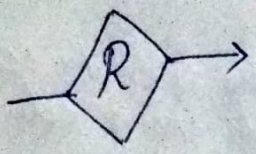
Primary key



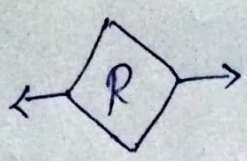
Discriminating attribute of weak entity set



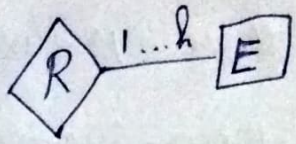
Many-to-many relationship



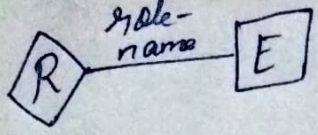
Many-to-one relationship



One-to-one



cardinality limits



role indicator

ISA

total generalization

Disjoint generalization

The two major part of E-R model is entity set & relationship set.

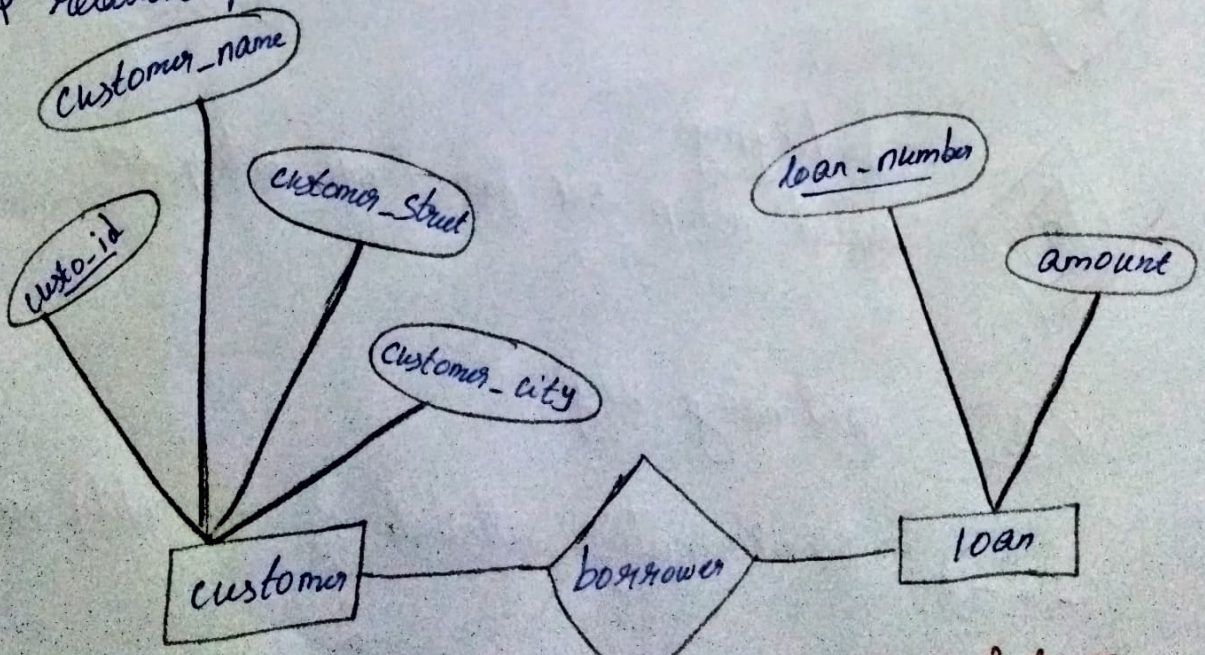


Fig:- E-R diagram - customers & loans.

In the above figure, relationship set borrower is many-to-many.

If the relationship set borrower were one-to-many, from customer to loan, then the line from borrower to customer would be directed, with an arrow pointing to the customer entity set.

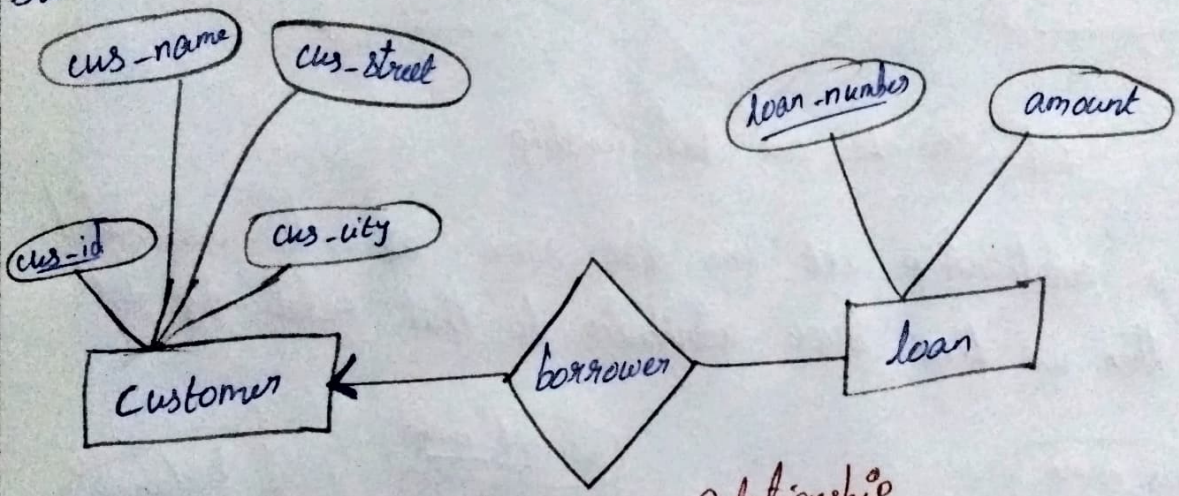


fig:- one-to-many Relationship

If the relationship set borrower were many-to-one from customer to loan, then the line from borrower to loan would have arrow pointing to the loan entity set.

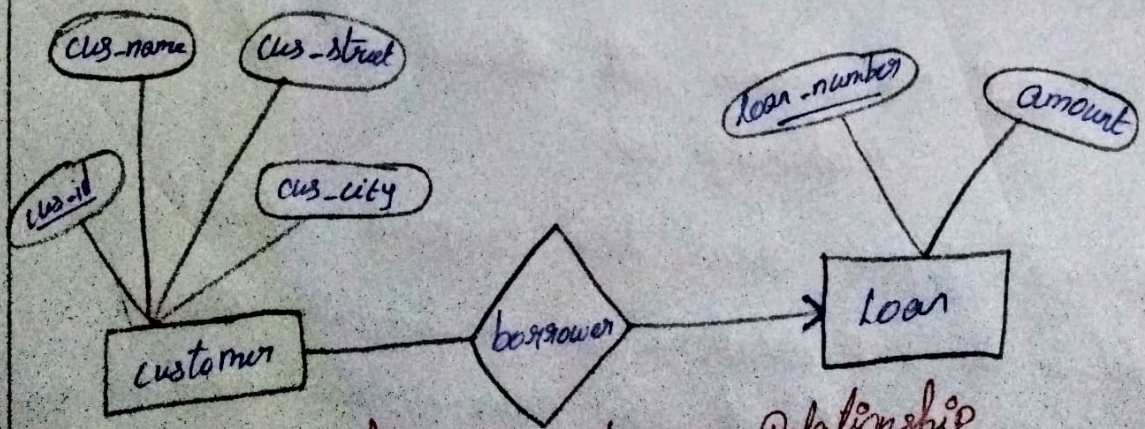


fig:- Many-to-one Relationship

Finally, if the relationship set borrower were one-to-one, then both lines from borrower would have arrows.

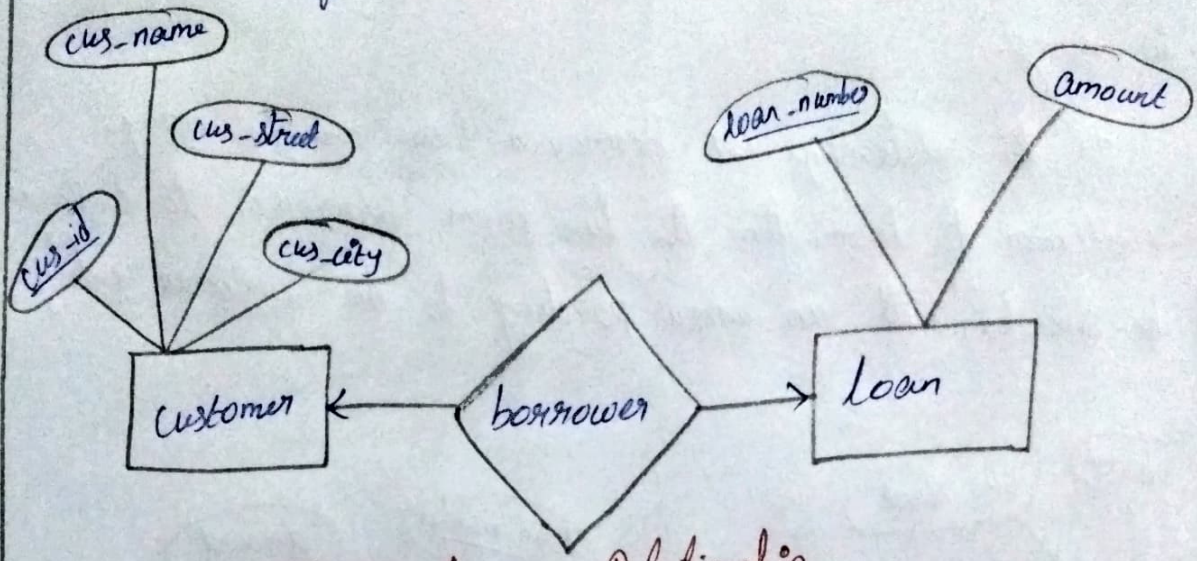


Fig:- One-to-one Relationship

If a relationship set has also some attributes associated with it, then we link these attributes to that relationship set.

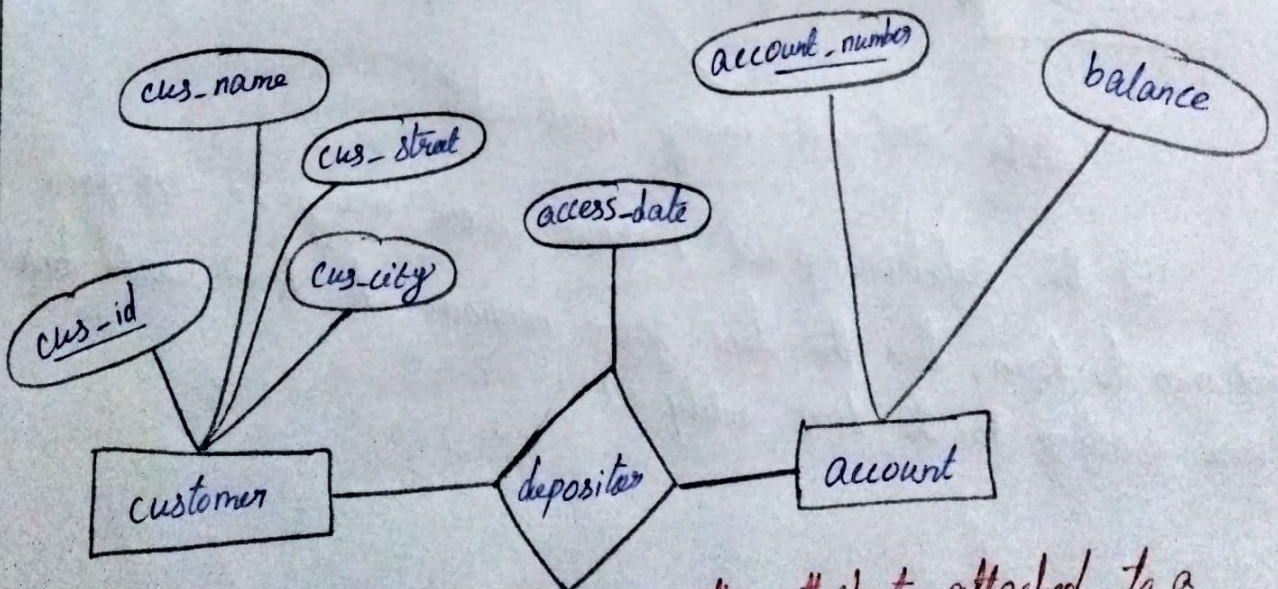


Fig:- E-R diagram with attribute attached to a relationship set.

The following figure shows how composite attributes can be represented in the E-R notations. Here, a composite attribute name, with component attribute first_name, middle_initial & last_name

replaces the simple attribute customer_name of customer & also illustrates a multivalued attributes phone_number, derived attributes age.

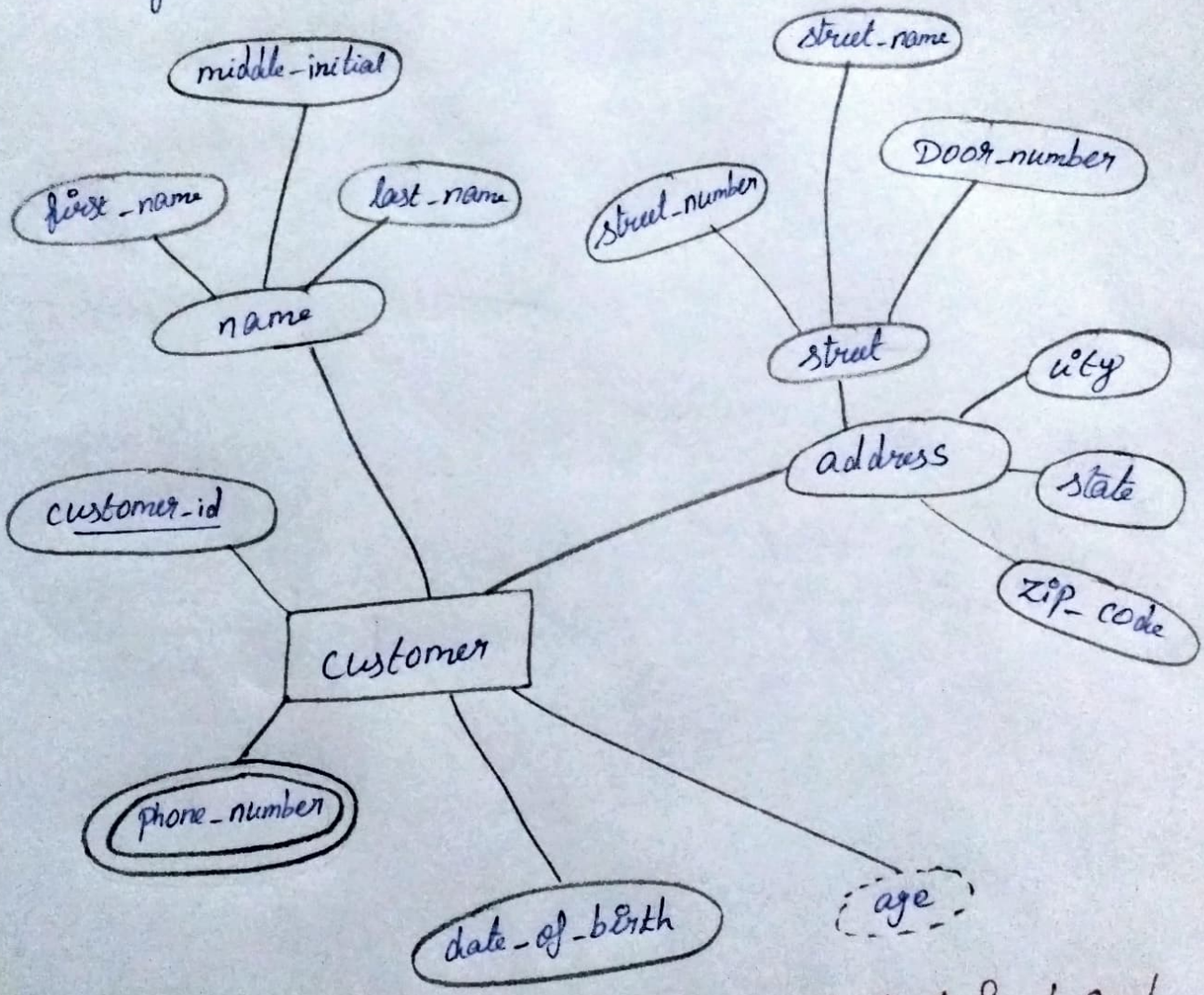


Fig:- E-R diagram with composite, Multivalued & derived attributes.