Date
Page

L-1 Introductory concepts of DBMS.

What is DBMS?

-> Database Management system is a collection of inter-related data in a set of programms to access those data.

VIMP

* Disadvantages of file system compare to the DBMS OR Disadvantages of conventional file system compare to the DBMS.

(i) Data Redundancy.

-> It is possible that the same information may be duplicated in different files, this leads to the Data redundancy.

-> Data redundancy results in memory wastage.

-> For e.g.: Consider that some customers have both kinds of account- saving and current. In this case data about customers such as name, address, email & contact number will be duplicated in both files saving account file & current account file.

→ In other words, same information will be store in two different location (files) & it waste memory.

(ii) Data Inconsistency:-

→ Due to the data redundancy, it is possible that data may not be in consistant state.

→ For e.g.: Consider that an address of some customer changes and that customers has both kinds of accounts now it is possible that this changed address is updated in only one file from a leaving the updated add. in other file. As a result of this same customer will have two differ addresses in two different file making data inconsistant.

(iii) Limited Data sharing:-

→ Data are scatted in various files.

→ Different files may have different formates & this files may be stored in different folders (directories). May be a different computer a different departments.

→ So due to this data isolation, it is difficult to share data amount different app.

(iv) Integrity Problem:

→ Data integrity means that the data content in the db is both corrects & consistent for this purpose the data stored in the db must satisfied certain types of rules.

→ For e.g.: A balance for any account must not be zero such rules are inforced in the system by adding appropriate code in the app. prog.

→ But when new rules added such as balance should not be <₹5000 app. prog. need to be changed but it is not an easy task to change the program whenever required.

(v) Atomicity Problem:

→ Any operation on db must be atomic. There means that operation complete either 100% or 0%.

-> For e.g. : A fund transfer from one account to another accounts must happen fully, but computer system are vulnerable to failure such as system crash, virus attack, etc.

-> If a system failure occurs during the fund transfer operation. It may possible that amount to be transferred (for e.g. ₹500) is debited from one account but it is not credited to another account.

-> This leaves db in consistant state but it is difficult to ensure atomicity in a file processing system.

(vi) Concurrent Access Anomalies:

-> Multiple users are allowed to access data concurrently this is for the better performance & faster response.

-> Consider an operation to debit an account the prog. reads the old balance & calculates the new balance & writes the new balance back to the db.

→ Suppose an account has a balance of ₹5000. now a concurrent withdrawal of ₹1000 & ₹2000 may leave the balance ₹4000 or ₹3000 depending upon there completion time rather than the current value of ₹2000.

→ Here concurrent data access should be allowed under some supervision.

→ But due to lake of co-ordination among the different app. program, this is not possible in file processing system.

(vii) Security Problem :-

→ DB should be accessble to the users in a limited way.

→ Each user should be allowed to access data concerning his app. only.

→ For e.g. : A customer can check balance only for his or her on account. He/she should not have acress the information about others account.

Requirement

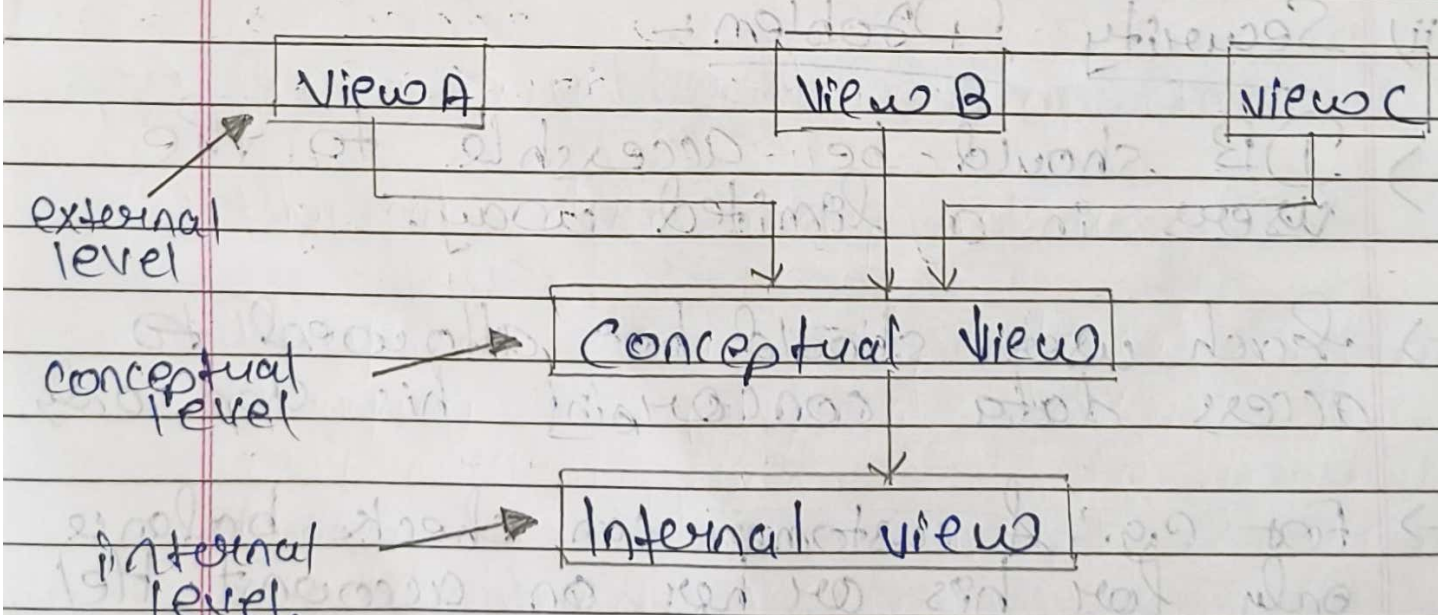
Date
Page

-> But in a file processing system app. programs are added in adhoc manner by different programers, so it is difficult to enforce such a kind of security constraits.

* Three level of data abstraction (OR) Views of the db (OR) three levels of ANSI SPARC db system.

-> The ANSI SPARC architecture divided into the three levels.

(i) External level.
(ii) Conceptual level.
(iii) Internal level.



(i) Internal level:

-> This is the lowest level of data abstraction.

-> It describes how the data are actually stored on storage devices.

-> It is also known as physical level.

-> The internal view is described by internal schema. That consists of defination of stored record, method of representing the data & access method use.

(ii) Conceptual level:

-> This is next higher level of data abstraction.

-> It describes what data are stored in the db and what relationships exists among those data.

-> It is also known as logical level.

-> Conceptual view is define by conceptual schema. And it describes all records & relationship.

(iii) External level:-

→ This is the highest level of data abstraction.

→ It is also known as view level.

→ It describes only part of entire db that a particular user requires.

* Data Independence:

→ Data Independence is the ability to modify a schema defination in one level without affecting a schema defination in the next higher level.

→ There are 2 types of data Independe

(i) Physical data independence:

→ Physical data independence allows changing in physical storage device's or organization of file without change in the conceptual view or external view.

→ Modification at the internal level are occassionally necessary to improve the performance.

→ Physical data independence seperates conceptual level from the internal level.

(ii) Logical data independence:

→ Logical data independence is abilite to modify the conceptual schema without requiring any change in the application program.

→ Conceptual schema can be changed without affecting the existing external schema.

→ Modification at the logical level are necessary whenever the logical structure of the db is alter.

→ Logical data independence seperates external level from the conceptual level.

→ It is difficult to achive logical data independence.

Imp.

Different DB users.

-> There are 4 different db users.

1) Application programers:
-> This users are computer proffesionals who write the app. prog. using some tools
-> For e.g.: Software developers.

2) ~~Sophisticated users~~ Sophisticated users:
-> These users interact with the system without writing the program.
-> They form their request in a db query language.
-> For e.g.: Analyst.

3) Specialized users:
-> These users write specialized db application. thats do not fit into the traditional data processing frame work.
-> For e.g.: db administrator.

4) NAIVE users:

-> These users are unsophisticated users who have very less knowledge of db system.

-> These users interact with the system by using one of the app. programs that have been written previously.

-> For e.g.: clerk in a bank.

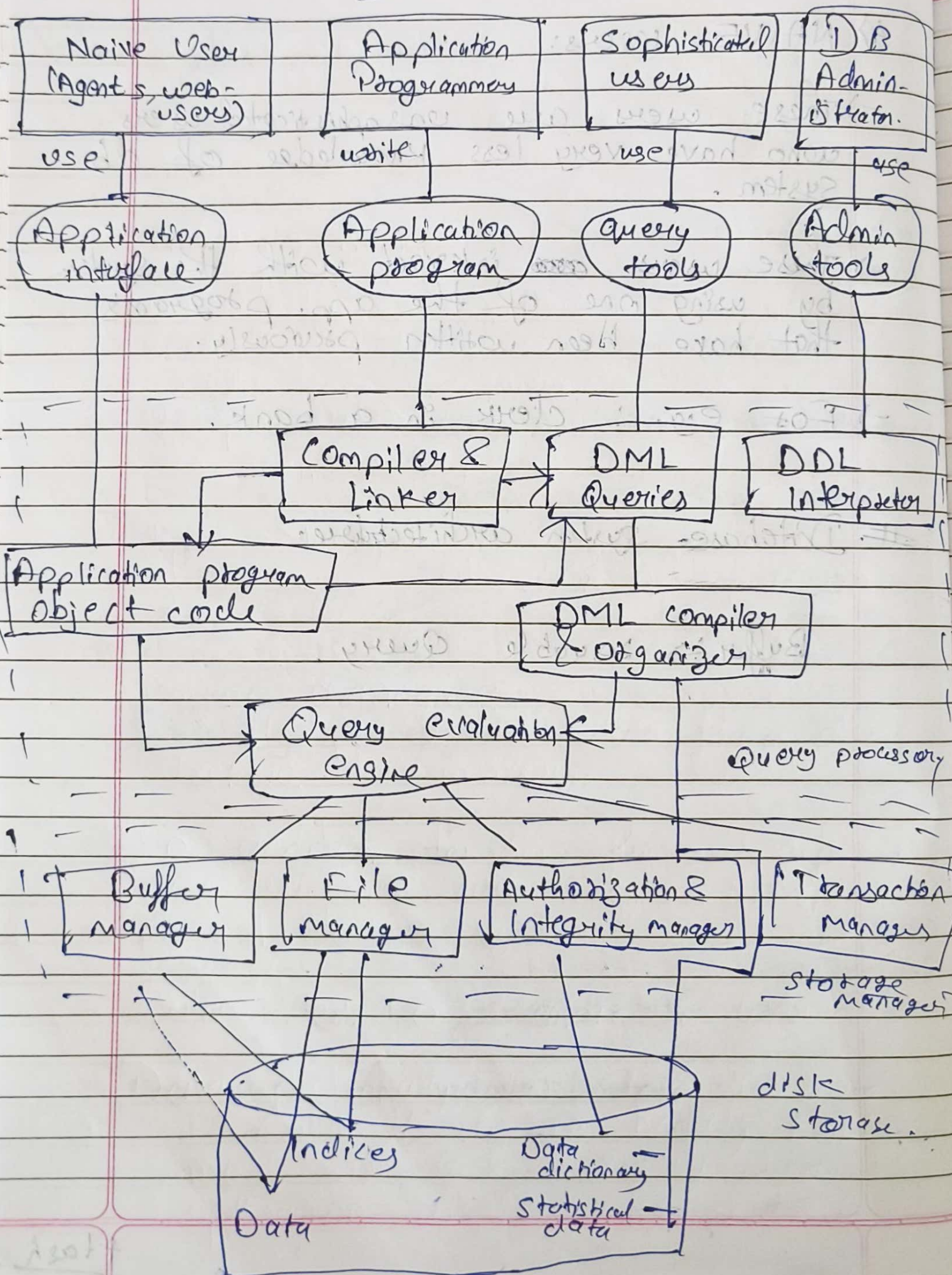
Database system architecture.

Buffer => readable Query.

flash

Date
Page

Architecture.



Date
Page

* Query Processor unit:

-> Query Processor unit deal with the exclusion of DDL (Data defination language) & DML (Data Manipulation Language) statements!

- DDL interpreter: Interprets DDL statement into a set of tables containing in the meta data (data about the data).

- DML compiler: It translate DML statements into the low level instruction that query evaluation engine understand.

- Embed compiler: It converts DML statements embeded in an application programs into a normal processor call in the language.

- Query evaluation Engine: It execute low level instructions generated by DML compiler.

* Storage Manager Units:

Storage Manager units provides the interphase btw the low level data stored in the database & app. programs & query submited to the system.

- Authorization units:
  - It checks the authority of users to access the data.
- Integrity manager:
  - It check for the satisfaction of the intigrity constrat.

Transaction manager:
- It passive reactive automety & controls the concurrency.

File Manager:
- It manages Alogetion of space on disc storage.

Buffer manager:
It pichies the data from disc storage to memory.

=> In addition to this fui unit different data structures are required to empliment physical storage system.

- Data files: To storage the user data.
- Data dictionary: To store the meta data. It is use heavily Rall most for each & every data operation so its should be

access efficiently.

- Indices: To provide the faster access to the data items.

- Statistical data: To store the statistical information about the data in db. This inf. is used by the query processor to select efficient base to exicute the query.

4. Differentiate DA (Data Administrator) v/s DBA (Data base Administrator).

DA (Data Administrator)	DBA.
(i) The data administrator is the person in the organization who controls the data of the db.	The dba is the person in the organization who controls the design & the use of db.
(ii) It determine what data be stored in the db based on the requirement of the organization.	It provides necessary technical spports for implimenting of data.
(iii) DA is involved more in the requirment gathering analysis & designed phases.	DBA is involved more in the designed, development, testing, & operation phases.

(iv)	1) DA is a manager or same senior level person in an organization program who understand organization with respect to data.	DBA is a technical person having the known of DB technology.
(v)	DA is a business flows person but he/she should understand more about the db technology.	DBA is a technical person but he/she should understand more about the business to administer the db efficiently.

L-2 Data Models.

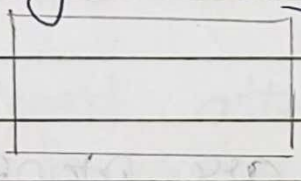
Date
Page

* Entity: An entity is a object/thing/person in the real world that is differentiable from all other objects.

-> For e.g.: Book, Student, Employee, etc.

* Entity sets: An entity set is a set of entities of same type that share the same properties / same attribute.

Note: Symbol for entity & entity set.



-> Rectangle.

-> For e.g.: The set of all students in a clg can be defined as entity set of students.

* Relationship: Relationship is an association (connection) between diff. entities.

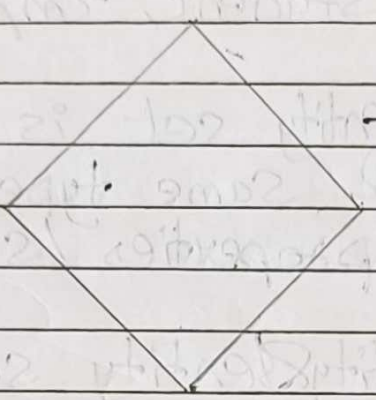
-> For e.g.: Book is issued by the student, where book & student are entities and issued is a relationship.

-> * Relationship btw 2 entities is called binary relationship.

Date
Page

* Relationship set: Relationship set is a set of relationship of the same type.

Note: Symbol for relation & relationship set.



→ diamond.

* Attributes: Attributes are properties hold by each member of entity set.

→ For e.g.. Entity is a student and its attributes are Enrollement no, name, Address, CPI, etc.

⇒ Types of Attributes:

1) Simple attributes: It cannot divided into the sub-parts.

→ For e.g.: CPI, roll no, etc.

2) Composite attributes: It can be divided into the sub-parts.

-) For eg. (i) Name -> First name, Middle name and Last name.

(ii) Address -> Street no., city, district, code, pincode.

3) Singled valued attributes: It has singled data valued.

-) For e.g. Enrollement no., Addhar card no, DOB.

4) Multi valued attribute: It has a multiple data valued.

-) For e.g.: Phone_No.

5) Stored attribute: It's value is stored manually in the db.

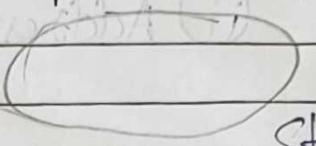
-) For e.g.: DOB.

6) Derived attribute: It's value is derived/ calculated from other attribute.

-) For e.g.: Age (It can be calculated using current date & DOB).

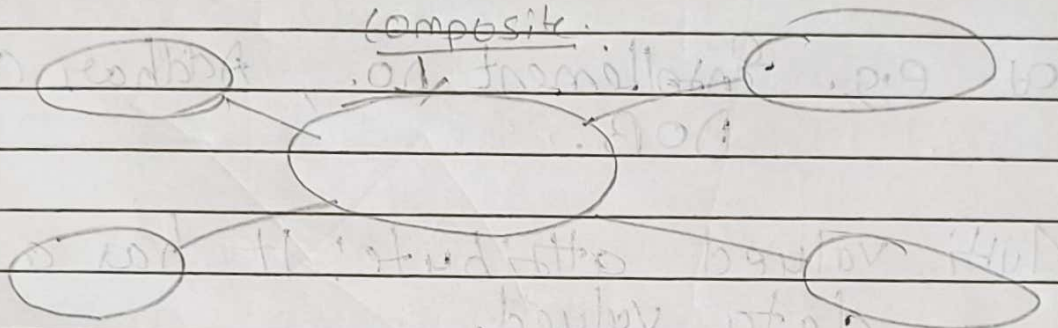
4. Descriptive attribute:

Note: Symbols : 1) Simple Attribute, Single valued Attribute, Stored attribute.

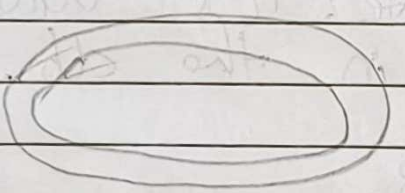


2) Composite Attribute.

composite



3) Multi-valued attribute.



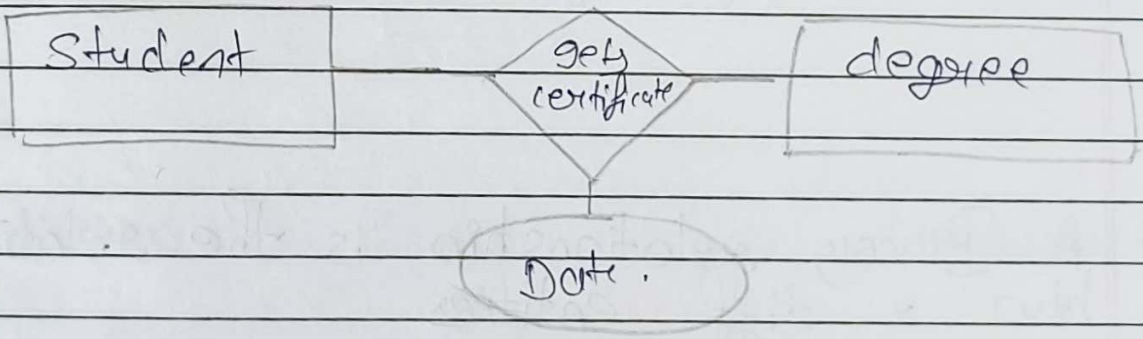
4) Derived attribute.

4. Descriptive attribute:

⇒ A relationship may also have attribute like an entity. These attributes are called descriptive attribute.

→ For eg: Student gets degree certificate on 14th Mar 2011. Student & degree are entities and gets certificate is relation and 14th Mar 2011 is attribute of relationship. gets certificate.

Note:



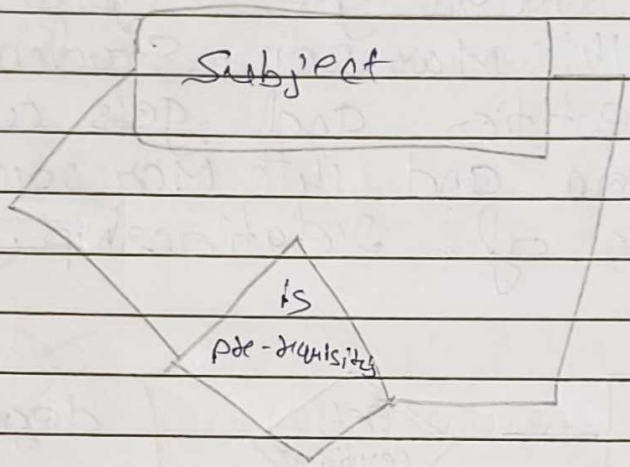
4. Degree of relationship:

⇒ The degree of relationship is the no. of entity types that participate in the relationship.

→ The 3 most common relationship in E-R model's are unary, binary, and ternary.

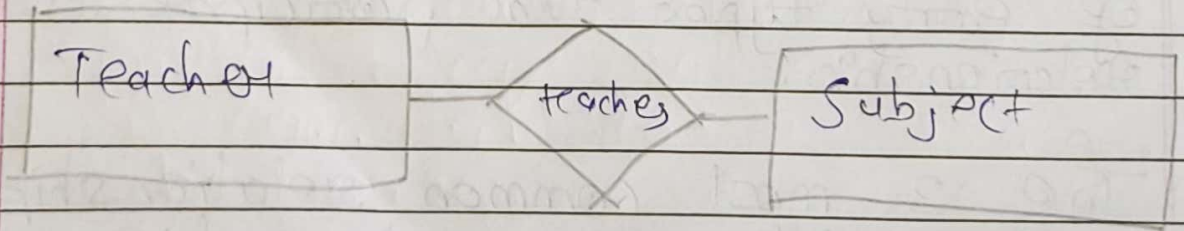
1) Unary relationship is when both participant entities in the relationship are the same entity.

→ For e.g.: Subjects may be pre-requisite for other's subject.



2) A Binary relationship is the relationship btw 2 diff. entities

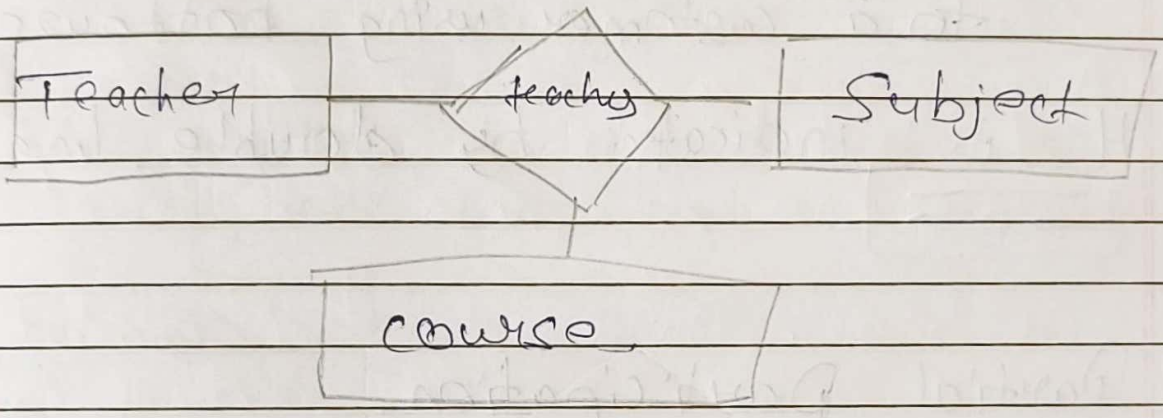
→ For e.g.: The university might may need to record which teacher thought which subjects.



Date
Page

A ternary relationship is the relationship btw 3 diff. entities.

For e.g.: The university might need to record which teacher taught which subject in which course.



Various participation constraints

It specifies the participation of an entity set in a relationship set.

There are 2 types of participation constraints
(i) Total participation
(ii) Partial participation

(i) Total participation: In total participation, every entity in the entity set participates in atleast one relationship in the relationship set

Date
Page

→ It specifies that every entity in superclass must be the member of same sub-class.

→ For e.g. Participation of loan in borrower relationship is total bcq every loan must be connected to a customer using borrower.

→ It is indicated by double line.

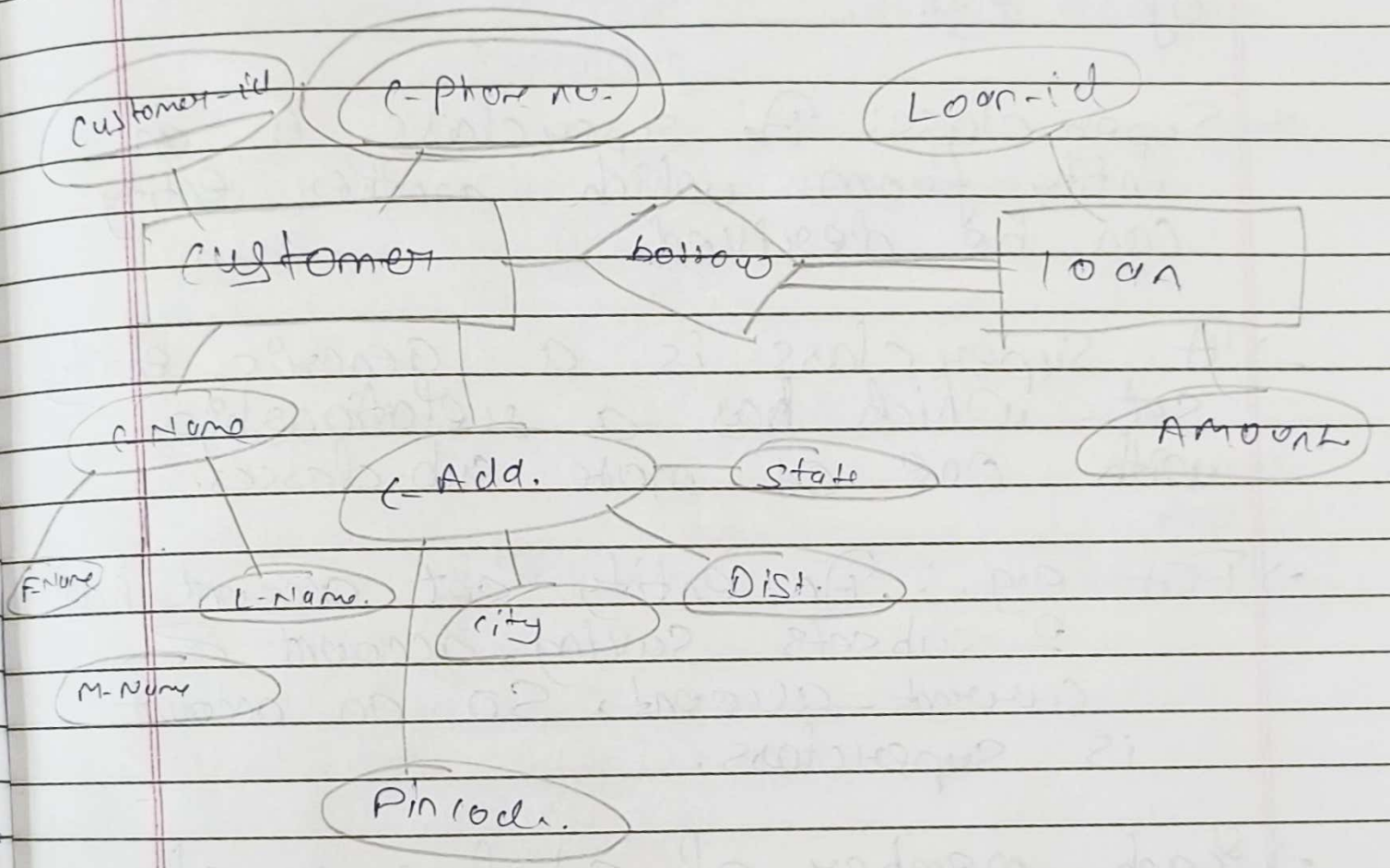
(ii) Partial Participation.

→ In partial participation, some entities may not participate in any relationship in the relationship set.

→ It specifies that even an entity may not belong to any sub-class.

→ For E.g. Participation of customer in borrower relationship is partial bcq not every customer is connected to a loan using borrower.

→ It is indicated by single line.


Customer-id
C-Phone no.
Loan-id
Customer
borrow
loan
C-Name
C-Add.
State
Amount
F-Name
L-Name.
City
Dist.
M-Name
Pincode.

Superclass and subclass in the E-R diagram. with the help of e.g.

* Superclass: A superclass is an entity from which another entity can be derived.

-> A superclass is a generic entity set which has a relationship with one or more sub-classes.

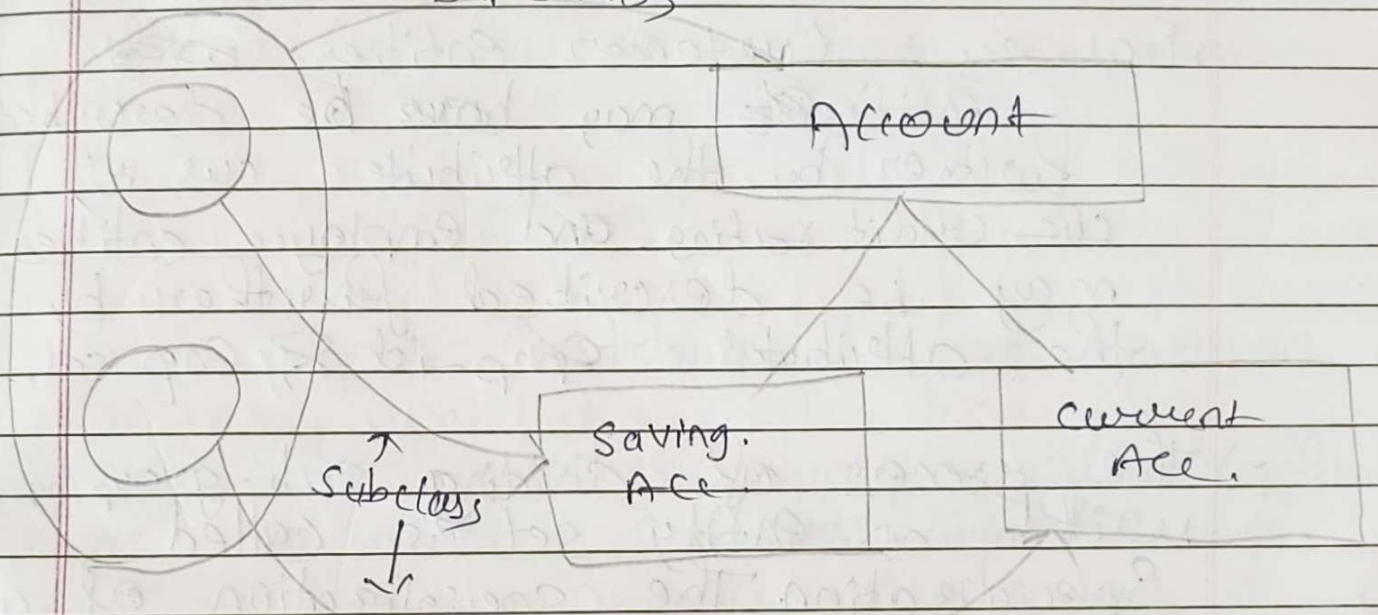
-> For eg.: An entity set account has 2 subsets saving_account & current_account. So an account is superclass.

-> Each member of subclass is also a member of superclass. So any saving account or current account is a member of entity set account.

* Sub-class: A subclass is an entity that is derived from ~~super~~ another entity.

-> A class is a subset of entities in an entity set which has attributes diff. from those in other subset.

→ T-Od e.g.: Entities of entity set account are grouped into 2 sub classes saving account & current account.



Generalization & Specialization in the E-R diagram with e.g.

* Specialization: A top down design process that creates sub-classes based on some diff characteristics of the entities in the super-class.

→ An entity set may include ~~sub grouping~~ sub-grouping of entities that are diff. in some way from other entity in the entity set.

Date
Page

-> For e.g.: A person in a bank may be further classified as a customer or employee.

-> For e.g.: Customer entities may be described further by the attribute cus_id, credit_rating. and employee entities may be described further by the attributes emp_id & emp_sal..

-> The process of dividing sub-grouping within an entity set is called Specialization. The specialization of a person allowed us to differentiate among the person acc. to their are employee or customers.

-> Now again, employers may be further classified as a officer, teller, or secretary

-> Each of these employee type is discribed by a set of attributes that includes all attributes of entity set employee + additional attributes.

-> For e.g.: officer entities may be described further by the attribute office no., teller entities by the attribute station_no., & hours per w

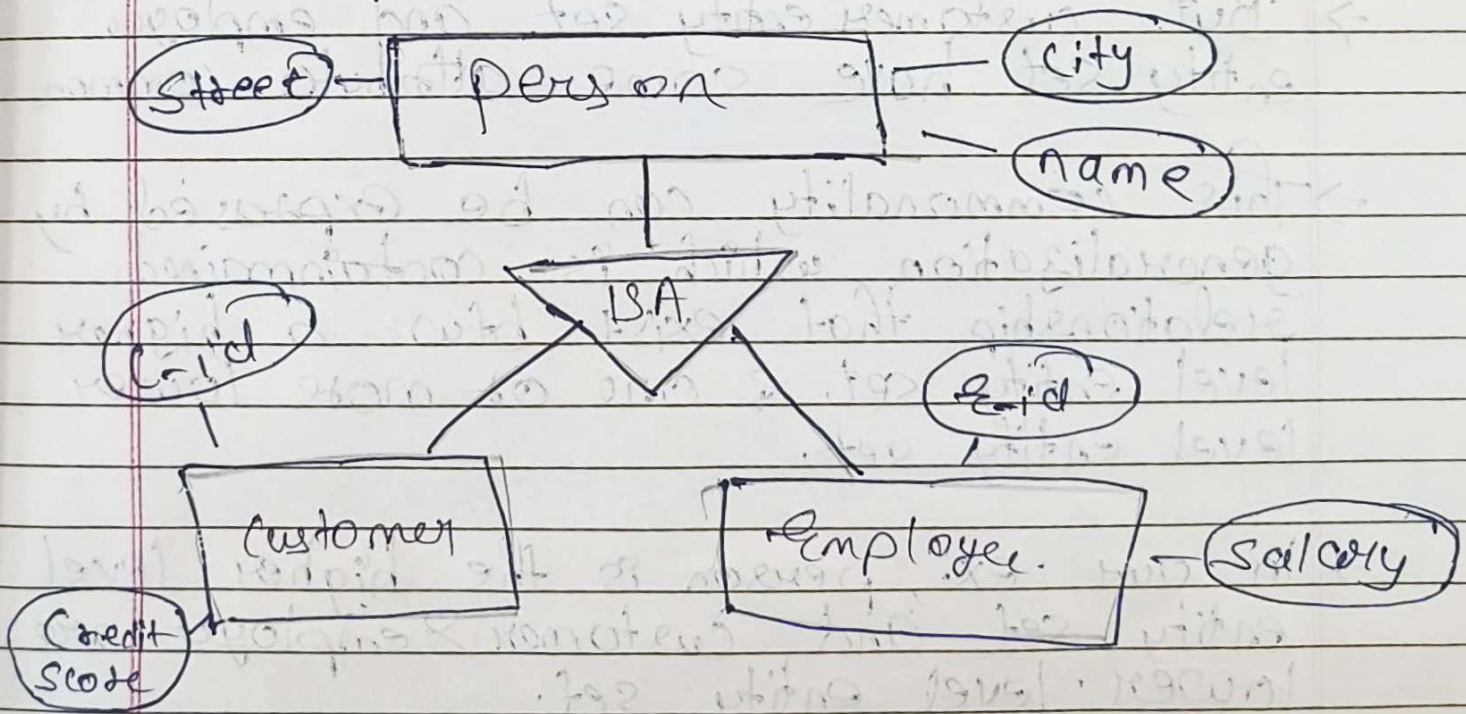
and secretary entities by the att. office_location.

-> In terms of E-R diagram, specialization is denoted by a triangle component labeled "ISA"

-> The label "ISA" stands for "is a"

-> For eg.: Customer "IS A" person.

-> The "ISA" relationship is also referred as superclass - subclass relationship.



* Generalization.

-> A bottom up design process that combines no. of entity sets that have same features into a higher level entity set.

→ The design process proceed in a bottom up manner in which multiple entity sets are synthesized into a higher level entity set on the basis of common features

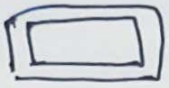
→ The db designer may have to first identify a customer entity set with the attributes - name, street, city and customer-id and an employee entity sets with the attributes - name, street, city; employee-id & salary.

→ But customer entity set and employee entity set have some attributes common.

→ This commonality can be expressed by generalization which is containmained relationship that exist btw a higher level entity set & one or more lower level entity set.

→ In our ex., person is the higher level entity set and customer & employee are lower level entity set.

→ Higher level entity set is called super-class & lower level entity set is called sub-class, so the person entity set is the super class

 - weak entity set.

Date
Page

of two sub-classes customer & employee

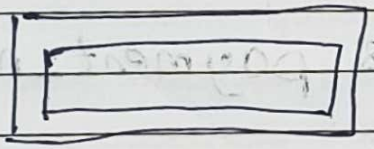
→ Differences in the two approaches may be characterized by their starting point and overall growth.

* Weak entity set with the help of e.g.

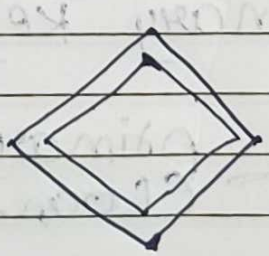
→ An entity set that does not have a primary key is called weak entity set.

→ The existence of a weak entity set depends on the existence of a strong entity set.

→ A weak entity set is indicated by double rectangle.



→ Weak entity relationship set is indicated by double diamond.



→ The discriminator (partial key) of a weak entity set is a set of attributes that differentiate btw all the entities

of a weak entity set.

→ The primary key of weak entity set is created by combining the primary of the strong entity set on which weak entity set is existence dependent and weak entity set's discriminator.

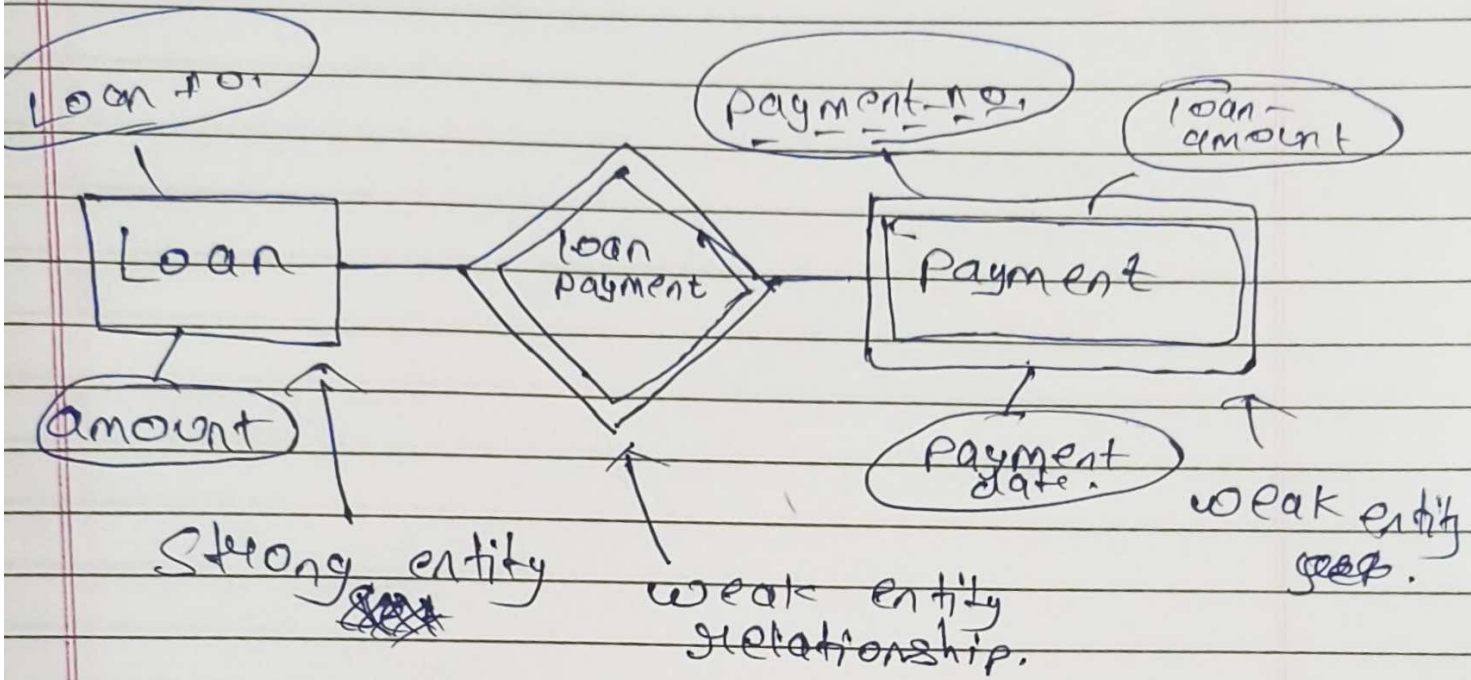
→ we underlined the discriminator attribute of a weak entity set with a dashed line.

→ For eg.: In below fig. there are 2 entities loan and payment in which loan is a strong entity and payment is a weak entity set

→ Payment entity has payment_no. which is discriminator.

→ Loan entity has Loan_no. as a primary key.

→ So primary key for payment is (Loan_no., payment_no.)


Loan no.
payment no.
loan amount
Loan
loan payment
payment
amount
payment date.
Strong entity
weak entity relationship.
weak entity