

Transaction

1. What is database transaction? Discuss ACID properties of database transaction.

Answer:

A Transaction is a logical unit of work. It is the set of operations to perform unit of work. Transaction access data using read and write operations. Transaction which successfully completes its execution is said to have been committed, otherwise the transaction is aborted or rollback.

To maintain consistency in a database, before and after the transaction, certain properties are followed. These are called ACID Properties. ACID stands for Atomicity, Consistency, Isolation & Durability.

Atomicity: The entire transaction takes place at once or doesn't happen at all. It involves following two operations.

1. Abort: If a transaction aborts, changes made to the database are not visible.

2. Commit: If a transaction commits, changes made are visible.

Atomicity is also known as the "All or nothing rule".

Consistency: The database must be consistent before and after the transaction. It refers to the correctness of a database.

example:

The total amount before and after the transaction must be maintained.

Total before T occurs $= 500 + 200 = 700$

Total after T occurs $= 400 + 300 = 700$

Therefore, the data is consistent. Inconsistency occurs in case T_1 completes but T_2 fails.

As a result, T is complete.

Isolation: Multiple transactions occur independently without interference. (This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved if these were executed serially in some order.)

This property ensures that when multiple transactions run at the same time, the final outcome will be the same as if they ran one by one, in some order.

example:

Let $X = 50000$, $Y = 500$

consider two transaction T_1 & T_2

T_1	T_2
Read(X)	Read(X)
$X = X + 100$	Read(Y)
Write(X)	$Z = X + Y$
Read(Y)	Write(Z)
$Y = Y - 50$	
Write(Y)	

suppose, T_1 has been executed till Read(Y) and then T_2 starts. As a result T_2 reads the correct value of X but the incorrect value of Y & sum computed by,

$$T_2: (X+Y) = (50000 + 500) = 50,500$$

~~and etc~~ is thus not consistent with the sum at end of the transaction.

$$T_1: (X+Y) = (50000 + 450) = 50450$$

This results in database inconsistency, due to a loss of 50 units.

Each transaction must be isolated from other transactions until it is completed. The isolation prevents dirty reads, non-repeatable reads & phantom reads.

~~Durability: The changes of a successful transaction occurs even if the system failure occurs. The effects of the transaction, thus, ~~never~~ never lost.~~

Durability: It ensures that once a transaction is committed, its changes are permanent and will survive any subsequent system failures. The transaction's changes are saved to the database permanently, and even if the system crashes, the changes remain intact and can be recovered.

Overall, ACID properties provide a framework for ensuring data consistency, integrity & reliability in dbms.

2. What are the different states of Transaction? Explain briefly with necessary diagram.

Answer:

A transaction log is a file maintained by the recovery management component to record all the activities of the transaction. After the ~~commit~~ commit is done transaction log file is removed.

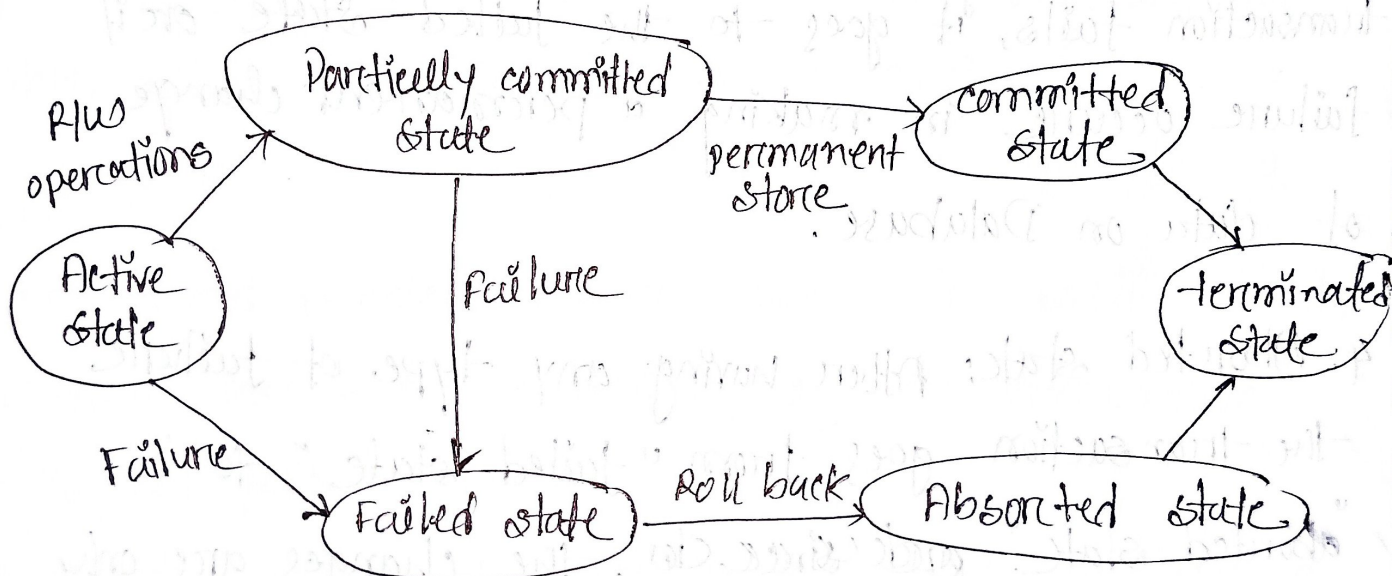


Fig: Transaction states in DBMS

1. Active state: when the instructions of the transaction are running then the transaction is in active state. If all the read/write operations are performed without any error then it goes to the particelly committed state. if ~~the~~ any instruction fails it goes to the failed state.

2. Partially committed: - After completion of all the read/write operation the changes are made in main memory or local buffer. If the changes are made permanent on the database then the state will change to "committed state" and in case of failure it will go to the "failure state".

3. Failed state: when any instruction of the transaction fails, it goes to the "failed state" or if failure occurs in making a permanent change of data on Database.

4. Aborted state: After having any type of failure the transaction goes from "failed state" to "aborted state" ~~and since~~, the changes are only made to local buffer or main memory and hence these changes are deleted or rolled-back.

5. Committed state: It is the state when the changes are made permanent on the Database and the transaction is complete, and therefore ~~the~~ terminated in the "terminated state".

6. Terminated state: If there isn't any roll-back on the transaction comes from the "committed state", then the system is consistent and ready for new transaction and old transaction is terminated.

These state are responsible for successfully completion of transaction.

3. Explain dirty read problem in transaction. What are the possible solutions to prevent this problem?

Answer:

The dirty read problem occurs when a transaction reads data that has been updated by another transaction that is not yet committed.

This situation can occur when a transaction modifies a data item and then fails to commit the changes due to a system failure, network error, or other issue.

Example:	T ₁	T ₂
	R(A) W(A) : : : Failure	R(A) // Dirty read W(A) commit

To prevent this problem the possible solution is given belows:

1. Ignore these type of scenarios.
 2. Don't read any value from local buffer.
 3. Read value directly from memory if needed.
 4. If read from local buffer before committing wait for previous transaction.
 5. Read value after committing.
 6. If read then ensure there is no possibility to failure.
 7. by making the schedule serializable.
4. Explain the Phantom read and last update problem in transaction with example.

Answer:

Phantom read problem:

The phantom read problem occurs when a transaction reads a variable once but when it tries to read that same variable again, an ~~error~~ error occurs saying that the variable does not exist.

Example:

T_1	T_2
$R(A)$	$R(A)$
Delete(A)	$R(A)$ / missing

In the above example, once transaction T_2 reads variable A, transaction T_1 deletes the variable A without transaction T_2 knowledge. Thus, when transaction T_2 tries to read A, it is not able to do it.

Lost update Problem:

In the lost update problem, an update done to a data item by a transaction is lost as it is overwritten by the update done by another transaction.

Example:

A lost update occurs when two different transactions are trying to update the same column on the same row within a database at the same time.

One transaction updates the column, but another

transaction that starts shortly after doesn't see this update & overwrites it. This causes the first update to be lost, as it is overwritten by the ~~so~~ 2nd transaction.

Example:

Time	T ₁	T ₂
t ₁	R(A)	—
t ₂	A = A - 20	—
t ₃	—	Read(A)
t ₄	—	A = A + 100
t ₅	—	—
t ₆	write(A)	—
t ₇	—	write(A)

In the above example, ~~at~~ t₂ time transaction T₁ update variable A, t₄ time transaction T₂ update variable A, ~~therefore~~ ~~so~~ thus T₁ transaction lost ~~updates~~ update value of A.

Q5. Check whether the schedule is conflict serializable or not?

S: $R_2(A)$; $W_2(A)$; $R_3(C)$; $W_2(B)$; $W_3(A)$; $W_3(C)$; $R_1(A)$; $R_1(B)$; $W_1(A)$; $W_1(B)$

Answer Order of transaction is called schedule.

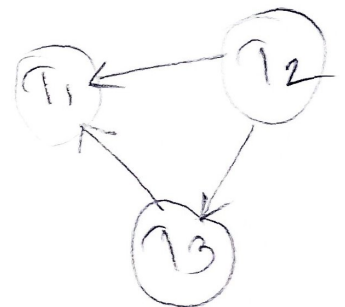
Given schedule:

S: $R_2(A)$; $W_2(A)$; $R_3(C)$; $W_2(B)$; $W_3(A)$; $W_3(C)$; $R_1(A)$; $R_1(B)$; $W_1(A)$; $W_1(B)$;

Identify Transactions & operations:

conflicts: $R-W$
 $W-R$
 $W-W$

Precedence graph:-



T_1	T_2	T_3
	$R_2(A)$	
	$W_2(A)$	
		$R_3(A)$
	$W_2(B)$	
		$W_3(A)$
		$W_3(C)$
$R_1(A)$		
$R_1(B)$		
$W_1(A)$		
$W_1(B)$		

$R_2(A) - W_3(A)$:-

$T_2 \rightarrow T_3$

$W_2(A) - R_3(A)$:-

$T_2 \rightarrow T_3$

$R_3(A) - W_1(A)$:-

$T_3 \rightarrow T_1$

$W_2(B) - R_1(B)$:-

$T_2 \rightarrow T_1$

E-R Diagrams

1. What is the basic difference between ER diagram and Schema diagram?

→ An Entity relationship diagram (ERD) is a visual representation of how entities relate to each other in ~~a system~~ the database.

1. ER diagram represent the erc model in a database, making them easy to convert.

2. ER diagram provide the purpose of real-world modelling of objects which makes them intently useful.

3. ER diagrams require no technical knowledge & no hardware support.

4. These diagram are very easy to understand & easy to create.

5. It gives a standard & solution for visualizing the data logically.

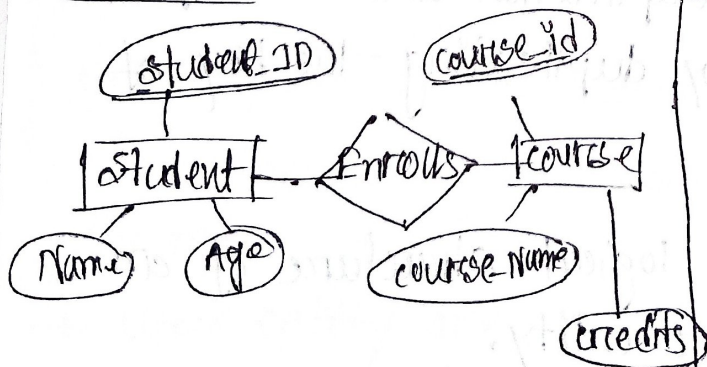
A schema diagram is a visual representation of a database's structure and organization.

Basic difference between er diagram and schema diagram.

ER Diagram

1. Represents the high-level conceptual design of a database.
2. Focuses on entities, their attributes & relationships between them.
3. Foreign keys are implied through relationship but not explicitly shown.
4. Abstract & independent of database implementation.
5. Used by designers to discuss database structure conceptually.

Examples



Schema Diagram

1. Represent the logical/physical structure of a database.
2. Focuses on tables, columns, datatype, & constraints.
3. Foreign keys are explicitly shown as attributes in tables.
4. Includes implementation-specific details.
5. Used by developers and DBAs for implementation & maintenance.

Example:

Student (Student ID, Name, Age)
Course (Course ID, Course Name, Credits)
Enrolls (Enroll ID, Student ID, Course ID)

2. We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Then why do we have weak entity sets?

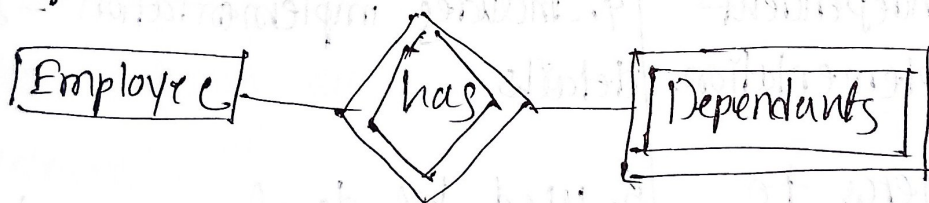
Ans: We can convert any weak entity set to a strong entity set by simply adding appropriate attributes.

weak entity: - a weak entity dependent on a strong entity to ensure its existence.

A weak entity is an entity in a relational database that cannot be uniquely identified by its attributes alone.

Weak entity are represented with double rectangular box in the ER diagram and the identifying relationships are represented with double diamond. Partial key attributes are represented with dotted lines.

example:



We have weak entity for several reasons:-

1. We want to avoid the data duplication and consequent possible inconsistencies caused by duplicating the key of the strong entity.
2. Weak entities reflect the logical structure of an entity being dependent on another entity.

3. Weak entities can be deleted automatically when their entity is deleted.

4. Weak entities can be stored physically with their strong entities.

3. Show with an example the association between a weak entity set and a strong entity set using E-R ~~data~~ diagram.

⇒ An entity is an object that exists and is distinguishable from other objects.

entity can be classified into two types:-

1. weak entity & 2. Strong entity

Strong entity:- A strong entity is an entity in a relational database that is not dependent on any other entity. It has a primary key or a table includes a primary key.

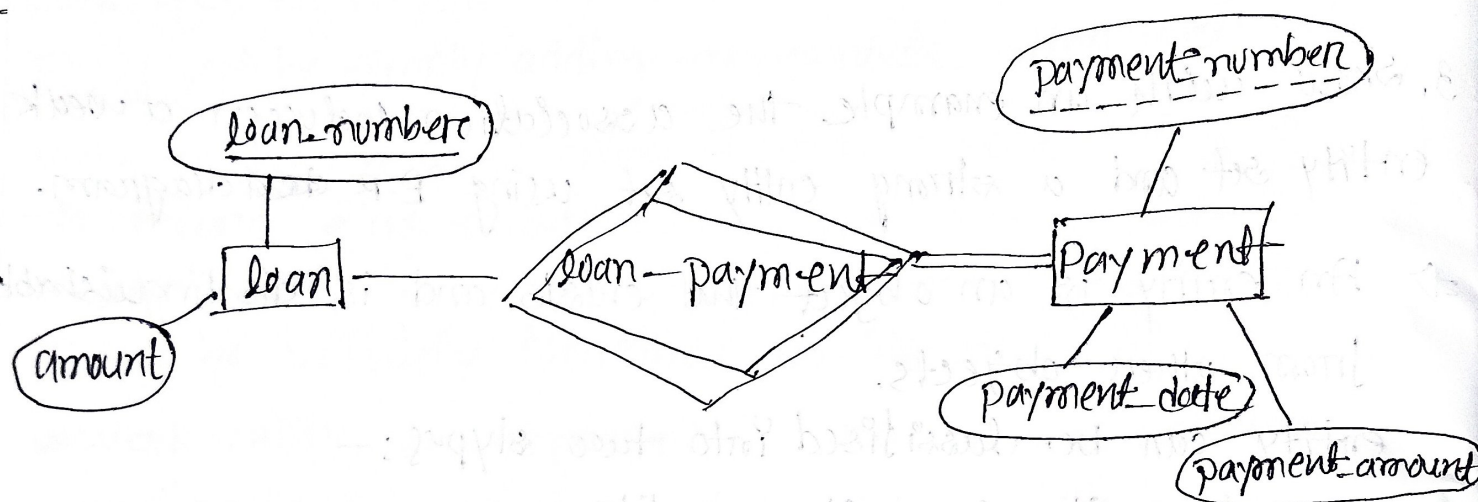
Weak entity:- A weak entity is an entity in a relational database that can't be uniquely identified by its attributes alone.

→ Strong entity ^{are} represented with solid rectangular, and the identifying relationships are represented with single diamond.

→ weak entity are represented with double rectangular and the identifying relationships are represented with double

diamond.

Now, the example of the association between weak entity set and a strong entity set using ER diagram is given below.



4. What is Entity Relationship Diagram? Explain the notational conventions used in ERD?

⇒ An Entity relationship diagram is a visual representation of the relationships among entities in a database.

It is a modeling technique used in database design to illustrate the structure of a database, and the relationships between entities.

ERDs use various symbols and notations to represent entities, attributes, relationships & constraints.

Key components of ERD:—

Entity:- An entity is an object that exists and is distinguishable from other object.

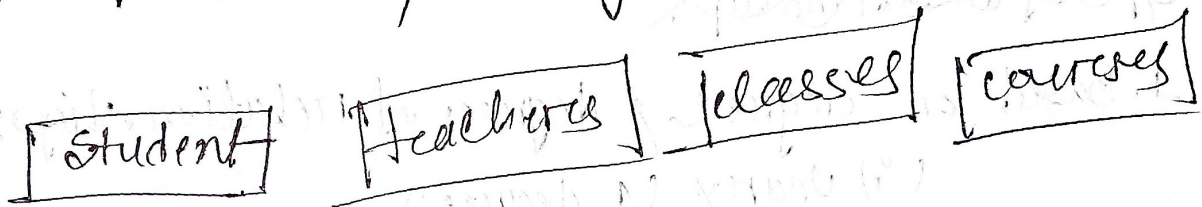
Entity can be a real world object, either animate or inanimate, that can be easily identifiable.

example:

① In a school database, students, teachers, classes and courses offered can be considered as entities.

Entity represented by a rectangular.

example:



Attribute: attributes are properties or characteristics of an entity. Attributes are used to describe the entity, attributes can help the database to be more structural and hierarchical.

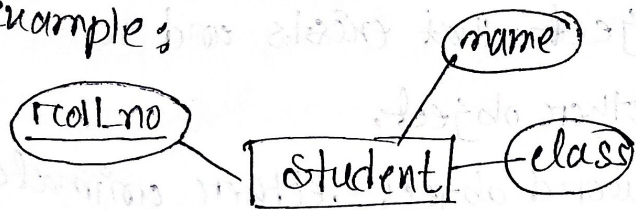
~~example~~ There are 8 types of attributes in dbms,

1. Simple attributes
2. Composite attributes
3. Single valued attributes
4. Multivalued attributes
5. Key attributes
6. Derived attributes
7. Stored attributes

8. Complex attributes

attributes ~~represent~~ are represented by an oval connected to its corresponding entity.

example:



Relationship: A relationship in a database is a logical association between two or more tables or entities. Relationships are represented by a diamond shape.

Types of relationships

1. Based on degree / degree of relationship:-

(i) unary (1 degree)

(ii) binary (2 degree)

(iii) Ternary (three or more degree)

2. Based on mapping cardinalities constraints:-

(i) one to one

(ii) one to many

(iii) Many to many

(iv) recursive relationship

Type of relationship:-

Primary Key:— A ~~primary~~ ^{dbms} key in a ~~database~~ is a column or set of ~~columns~~ columns that uniquely identifies each record in a table.

A primary key is a type of key that contains a column or set of columns in a table that uniquely identifies each row. primary key represent by underlined attribute or a separate notation.

Foreign key:— Foreign Keys are the columns of a table that points to the primary key of another table.

Foreign key build a relationships among tables & cross-reference between tables.

Represented with dashed lines connecting the foreign key to the referenced primary key.

Q. Explain the distinct between total & partial participation constraints.

⇒ Participation constraints in dbms refer to rules that determine the minimum and maximum participation of entities or relationships in a given relationship set.

There are two type of participation constraints in dbms such as:-

1. Total participation

2. partial participation.

The distinct between total & partial participation constraint is given below:-

Aspect	Total Participation	Partial Participation
Definition	Every entity in the entity set must participate in the relationship.	Some entities in the entity set may not participate in the relationship.
	Participation is mandatory for all entities.	Participation is optional for some entities.
	Represented by a double line in the ER diagram	Represented by a single line in the ER diagram.

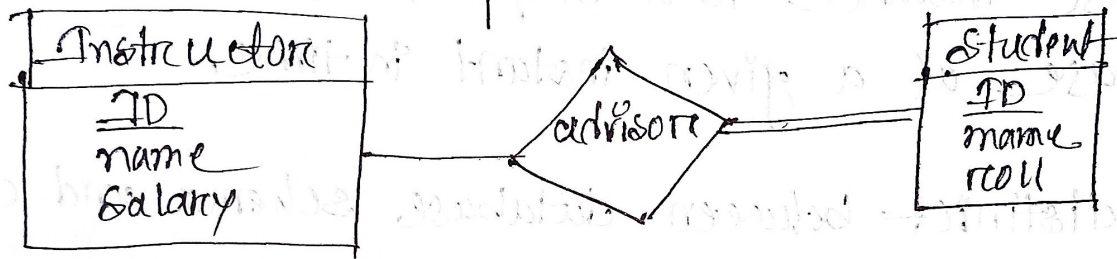
All entities are involved
in at least one relation-
-ship,

Some entities are not involved
in any relationship.

example: every student
must have an associated
instructor,

example: participation of instructor
in advisor is partial.

Example:



participation in advisor relationship set: -

(i) partial instructor

(ii) total student.

Q. Explain the distinct between database schema & instance.

→ Database schema is the logical structure of the database. A schema is a template that dictates how data items in a database will be stored, arranged and accessed.

There are three type of schema :-

(i) Logical schema: It describes the database designed at a logical level.

(ii) Physical

(iii) View

(ii) Physical schema: It describes the database designed at the physical level.

(iii) View schema: It defines the design of the database at the view level.

Database instance is a snapshot of the data in the database at a given instant in time.

The distinct between database schema and database instance is given below:-

Schema	Instance
It is the overall description of the database	It is the collection of information stored in a database at a particular moment.
The schema is same for the whole database.	Data in instance can be changed using addition, deletion & updation.
Does not change frequently	changes frequently

Define basic structure of the database how the data will be stored in the database.	It is the set of Information stored at a particular time.
Require Significant effort and planning to change.	Easily altered by performing create, read, write, update delete operations.
Table structures, relationships constraints	Data entries, records in tables.

Steps of Drawing ERD:-

1. Identify the Entities required.
2. Identify the attributes and Primary key for each Entity.
3. Identify the relationship.
4. Identify the cardinality, ratio and participation.
5. Draw the Diagram.

Human Resource management - (3.1)

Step-01: ~~Entity~~ Identify Entities

1. Department
2. Employees
3. managers
4. Projects

16-17] 3(a) Library:-

Step 01: Identify entities

1. Library
2. Books
3. borrowers
4. authors

Step 02: Identify attributes and primary key.

~~Book (ISBN number,~~

~~Library (library id, library~~

~~Library (library code, name, type);~~

~~Book (ISBN number, book title, author id, available copy);~~

~~Library (library code, name, type, address);~~

~~Authors (author id, author name, city).~~

~~Books (book id, book title, author id, available copy, library)~~

~~Borrowers (borrower id, name, address, phone numbers)~~

Library (library code, name, type, address);

Authors (author id, author name, city);

Books (ISBN number, book title, author id, cypis, library code)

Borrowers (borrower id, name, phone number, book id);

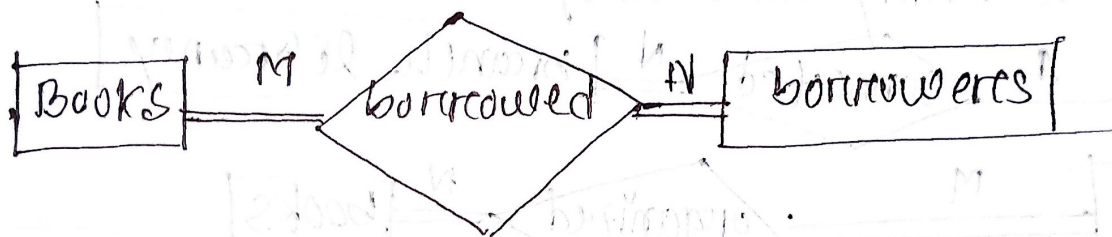
~~branch library (type, Name, address).~~

Step 03: Identify relationship

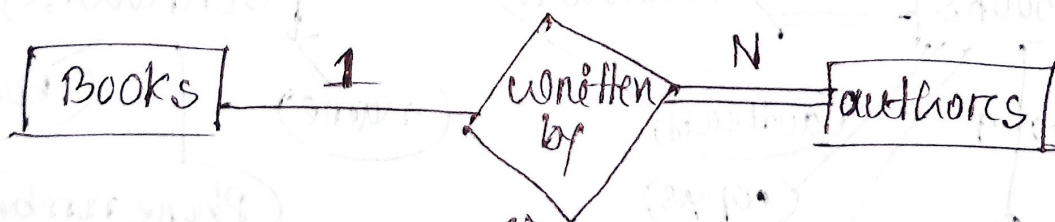
1. Books ~~have~~ borrowed borrowers.
2. Books ~~are~~ written ^{by} authors.
3. Books available in libraries.
4. Library connected branch library.
5. Library organised by books

Step 04: Identify the cardinality Ratio & Participation.

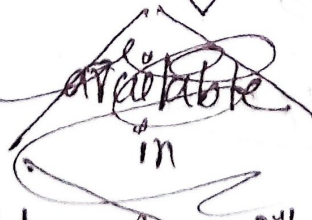
1. Books borrowed borrowers.



2. Books written ~~and~~ by authors.



3. ~~Books~~



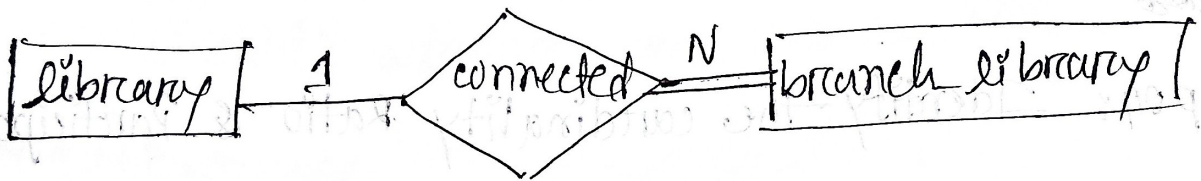
3. Books available in libraries



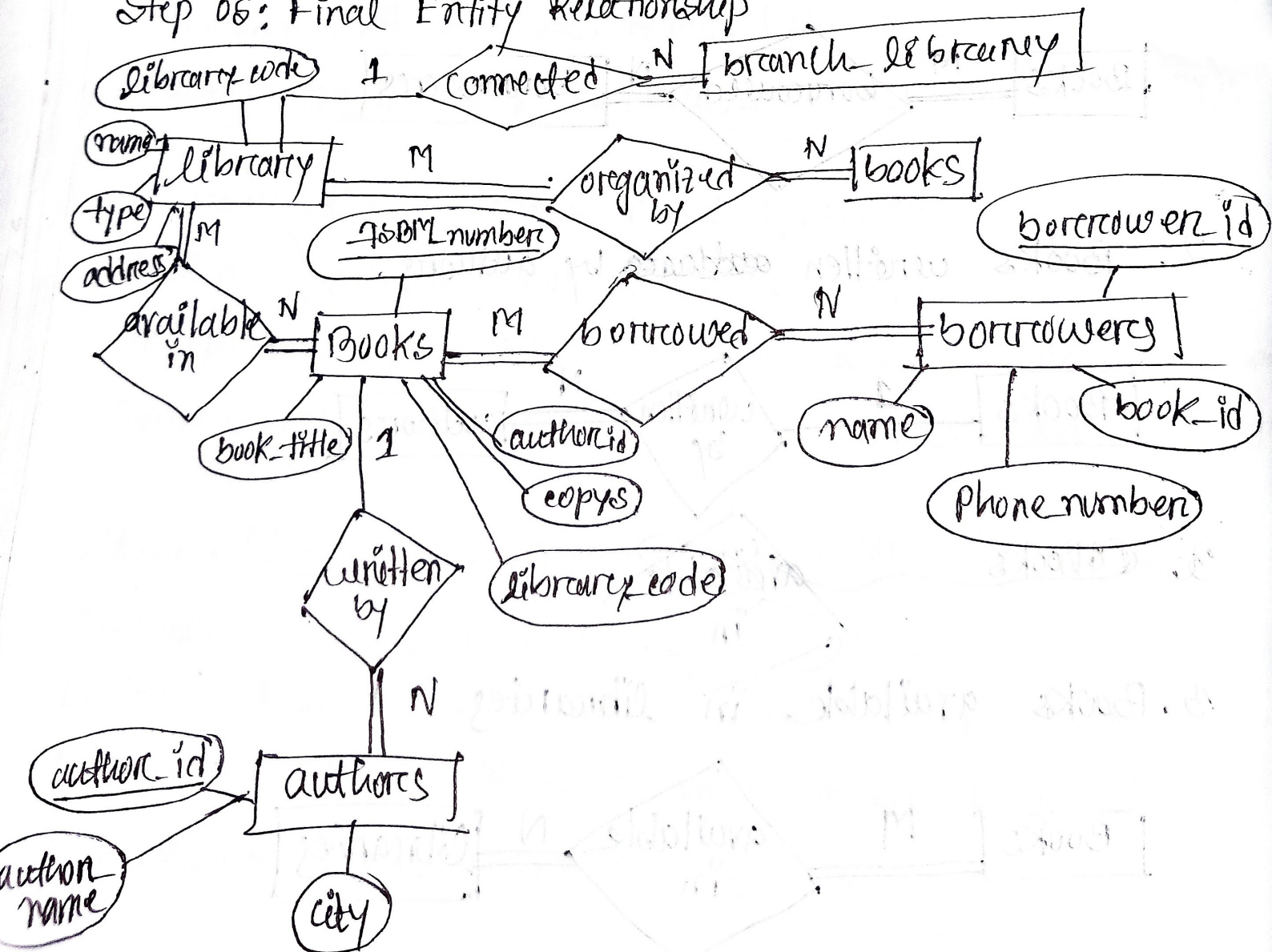
4. Library organized by books



5. library connected branch library



Step 06: Final Entity Relationship



3(b) Entities: (i) Lot
(ii) production units
(iii) Raw Materials

Schema:

(i) Lot (LotNumber, CreateDate, CostOf Materials)

(ii) ProductionUnits (serial-#, exactWeight, ProductType, qualityTest?, ProductDesc)

(iii) RawMaterials (MaterialID, Type, UnitCost)

(iv) Includes (LotNumber, serial-#)

(v) created from (LotNumber, MaterialID, Units)

Referential Integrity constraints:-

Primary Key: Each table has a primary key to uniquely identify each row.

(i) Lot: LotNumber (Primary Key)

(ii) ProductionUnits: serial-#

(iii) RawMaterials: MaterialID

Foreign Key: Foreign key are the column of the table that points to the primary key of another table.

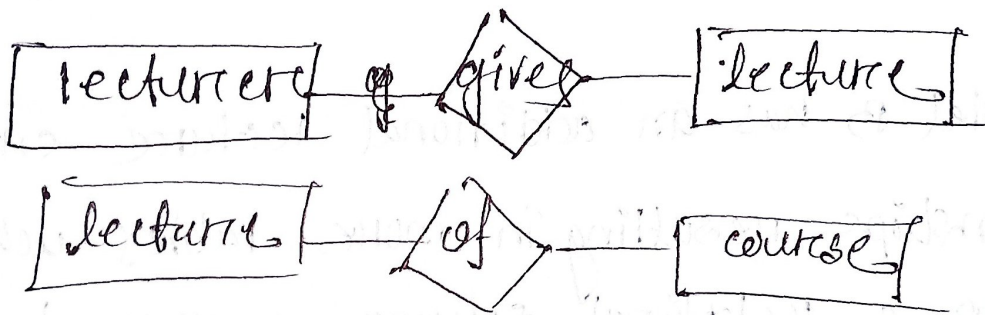
- (i) Includes, LotNumber references Lot.LotNumber.
- (ii) Includes, Serial# references ProductionUnits.Serial#.
- (iii) Created Form : LotNumber references Lot.LotNumber.
- (iv) Created Form, MaterialID references RawMaterials.MaterialID.

2017-18

4 (a) Models A:- lecturer, course



Model B:- lecturer, lecture, course



i). correct.

Explain:

In Model B, since there is a distinct lecture entity, a lecturer can give multiple lectures for a course.

In Model A, the teaches relationship can potentially allow multiple instance of the same lecturer teaching a course.

(ii) correct.

Explain: Model B explicitly includes the lecture entity allowing for individual lecture records with possible date and time attributes, making it suitable for tracking all past and present lectures.

(iii) correct.

Explain: In Model A lacks a separate lecture entity. So there is no direct way to associate specific date and times with individual lectures.

(iv) correct.

Explain: - Model B has an additional lecture entity and two relationships resulting in more tables when converted to a relational schema compared to model which only has two entities & one relationship.

4(b) | ~~Database~~ A company database system:—

Entities:—

(i) employee

(ii) department

(iii) child

Attributes:-

- (i) Employee (SSN, name, salary, phone)
- (ii) Department (dno, dname, budget)
- (iii) Child (name, SSN, age)

Relationship:-

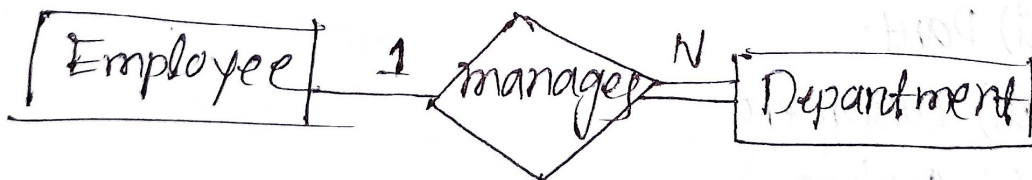
- (i) Employee works in Department
- (ii) Employee manages department
- (iii) Employee has child.

Cardinality ratio and Participation:-

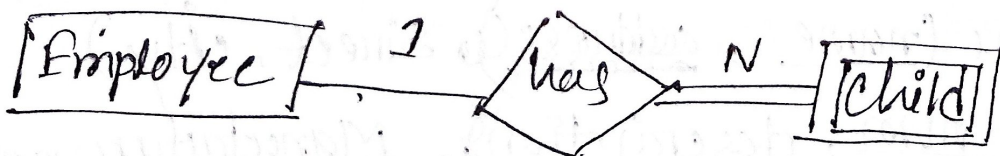
- (i) Employee works in Department



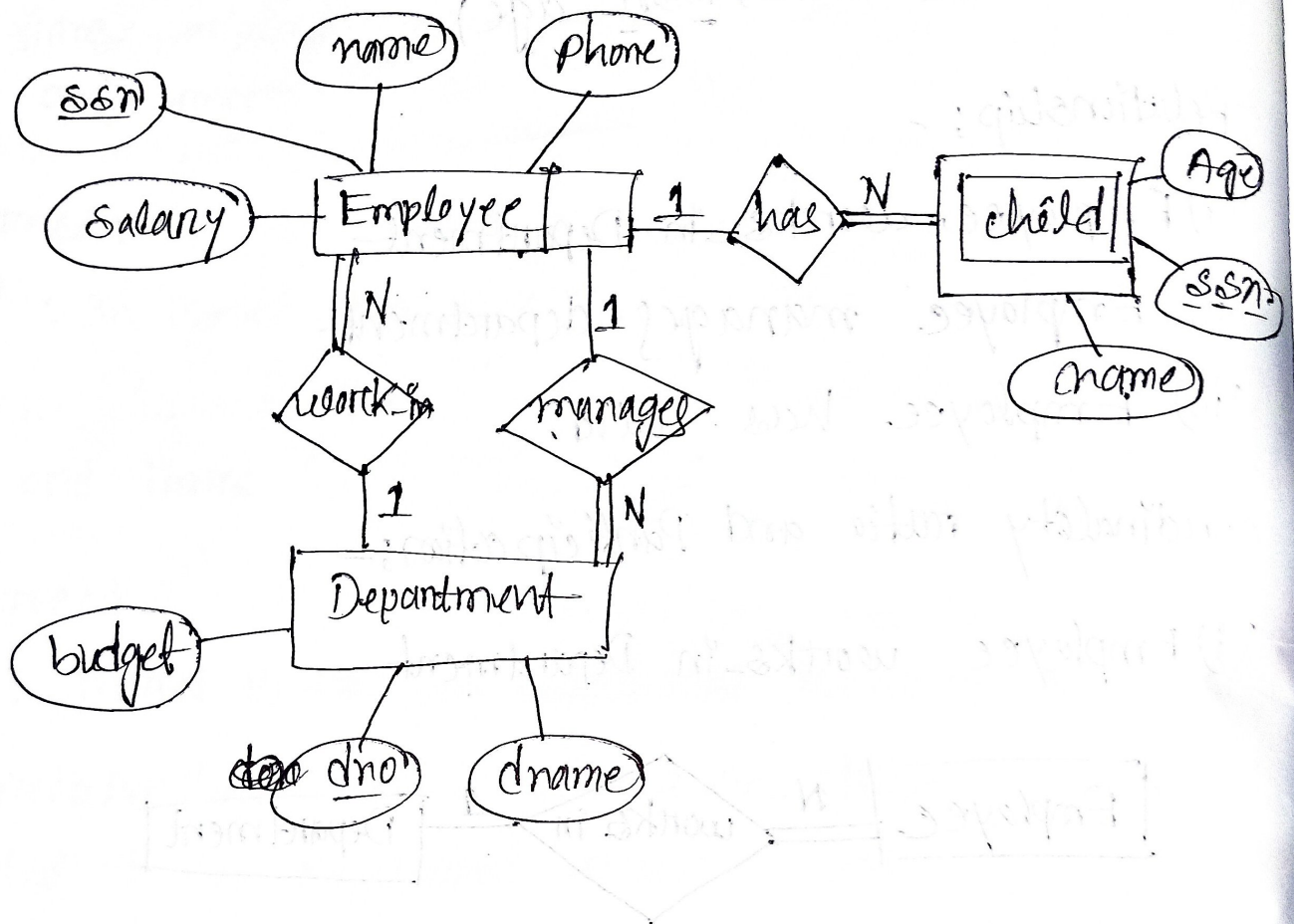
- (ii) Employee manages department



- (iii) Employee has child.



E-R diagram:



3 b/c)

- Entities:—
- (i) Manufacturere
 - (ii) Part
 - (iii) customer
 - (iv) Order

relational schema:

- (i) Manufacturere (name, ~~address~~, street, city)
- (ii) part (part-num, description, Manufactur_name)

- (iii) Customer (ID, name)
- (iv) Order (order_num, customer_ID)
- (v) made by (name, part_num)
- (vi) contains (quantity, part_num,
- (v) contains (order_num, part_num, quantity)

Attributes:

- (i) Manufacturer (name, address (street & city))
- (ii) Part (Part_num; Description)
- (iii) Customer (ID, name)
- (iv) Order (order_num).

Primary keys:-

- (i) Manufacturer : name
- (ii) Part : part_num
- (iii) Customer : ID
- (iv) Order : order_num
- (v) constraints : composite key (order_num, Part_num).

Foreign keys:-

- (i) Manufacturer name in part reference Manufacturer (Name).
- (ii) Customer ID in Order reference customer (ID).

- (iii) Order_num in contains references Order (Order_num)
(iv) Part_num in contains references Part (Part_num).

[16-17]

4(a) A composite attribute is an attribute that can be broken down into smaller sub-parts, each with independent meaning.

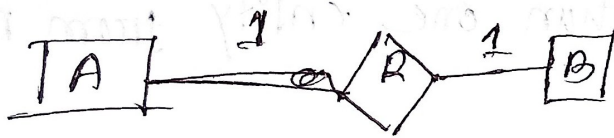
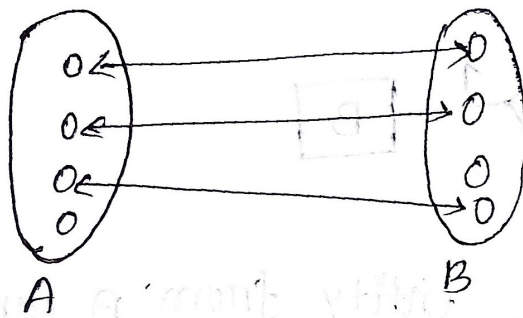
Here, createdate could be a composite attribute because it may contain parts such as day, month and year.

4(b) Data redundancy occurs when the same piece of information is stored in multiple place, leading to inconsistency.

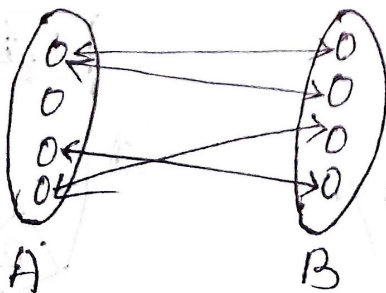
In this, ER diagram a redundancy issue might exist between RawMaterials and ProductionUnits. Specifically, attributes like unit of measurement could be duplicated across these entities if they are being managed in both places.

3) Q1) Matching cardinalities describe the type of association between two entities

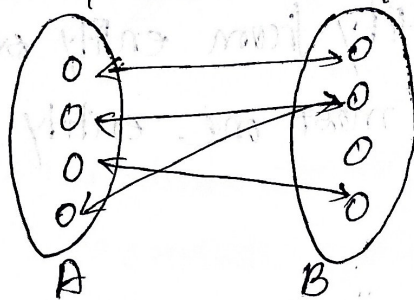
i) One to One: One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



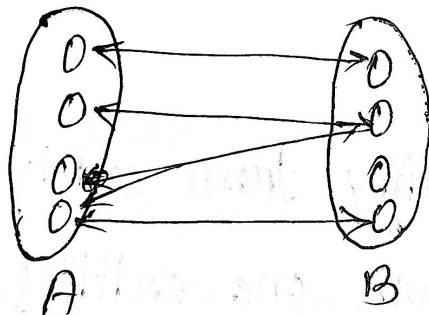
ii) One to many: One entity from entity set A can be associated with more than one entities of entity set B. However, an entity from entity set B, can be associated with at most one entity.



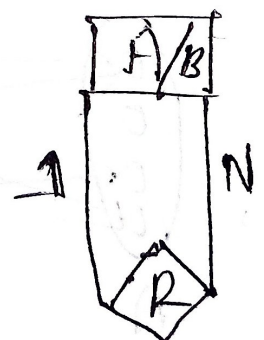
(iii) Many to One: More than one entity from entity set A can be associated with at most one entity set B. However an entity from entity set B can be associated with more than one entity from entity set A.



(iv) Many to Many: One entity from A can be associated with more than one entity from B and vice versa.



(v) Recursive relationship: relationship within an entity.



Keys

A key is a attributes or set of attributes that uniquely identifies any record from the table.

Super key: A set of attributes that collectively identifies an entity is an entity set.

candidate key: A minimal super key is called candidate key.

A Entity ~~can~~ set may have more than one candidate key.

primary key: A primary key is one of the candidate key chosen by the database designer to uniquely identify the entity set.

Foreign key: Foreign key are the column of ~~the~~ a table that point to the primary key of another table.

• They act as a cross-reference between tables.

• Build a relation ship among tables.

Primary	candidate	composite
1. There can be only one primary key in any table or relation	1. There can be multiple / more than one candidate key in a table / relation.	1. Two or more attributes (columns) that together can uniquely identify a tuple in a table table and work as PK
2. The primary key is unique & not null.	2. Where as a candidate unique & alternate keys can be null.	2. combination of columns.
3. Only one candidate key can be primary key	3. Each candidate key can qualify as primary key	3. creating & more than one primary key is jointly known as composite key
4. Student (ID, name, address) primary key: ID	4. Example: - Student (ID, name, email) candidate key: ID, email, (minimal super key are)	4. Example: - Student (ID, email, name) composite key: - ID, email

❏ The number of attributes in a table depends on the database management system being used. For example:

1. MySQL supports up to 4096 columns.

2. PostgreSQL supports an unlimited columns, limited only by system memory.

3. Other DBMS have their own limitations.

Index

Indexing is a basically a technique that helps in reducing the search time of a database query or it helps faster accessing the database.

There is a three type of indexing.

1. Primary indexing
2. Secondary indexing
3. Clustering indexing

Primary Indexing:-

- If the index is created on the basis of the primary key of the table, then it is known as primary indexing.

These primary key are unique to each record and contain 1:1 relation between the records.

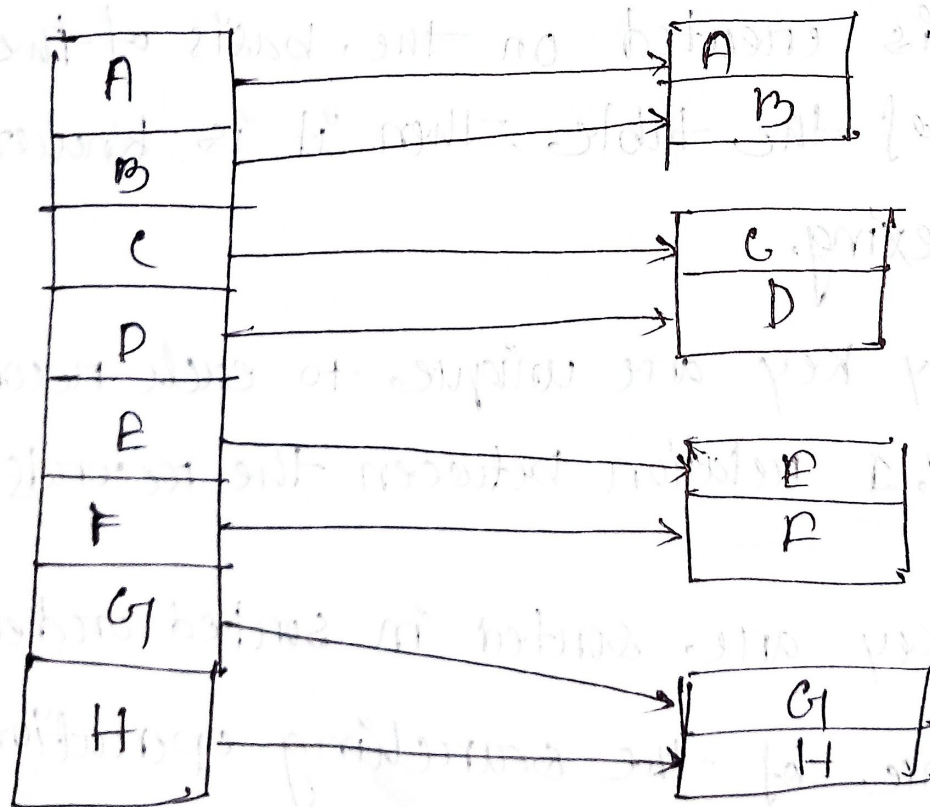
- As primary key are sorted in sorted order, the performance of the searching operation is quite efficient.

There are two type of primary indexing.

1. Dense index
2. Sparse index

Dense index:

- This dense index contains an index record for every search key value in the data file.
- It makes searching faster.
- In this, the number of records in the index table is same as the number of records in the main table.
- It needs more space to store index record itself.

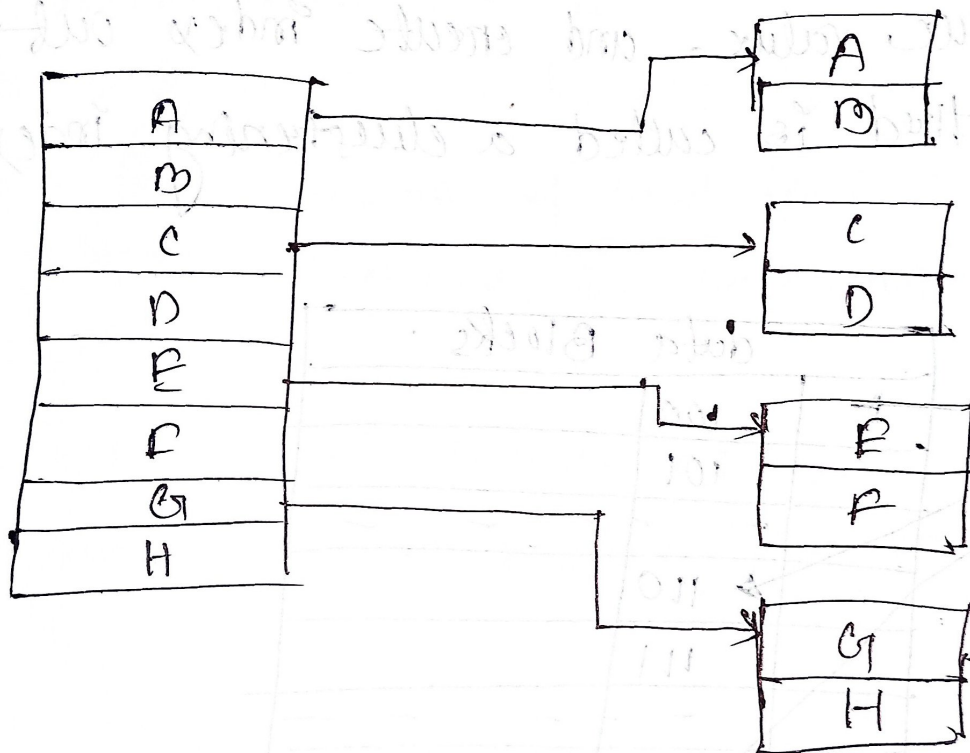


Data File

Index record.

Sparse index

- In the data file, index record appears only for a few items. Each item points to a block.
- In this, instead of pointing to each record in the main table, the index points to the records in the main table in a gap.

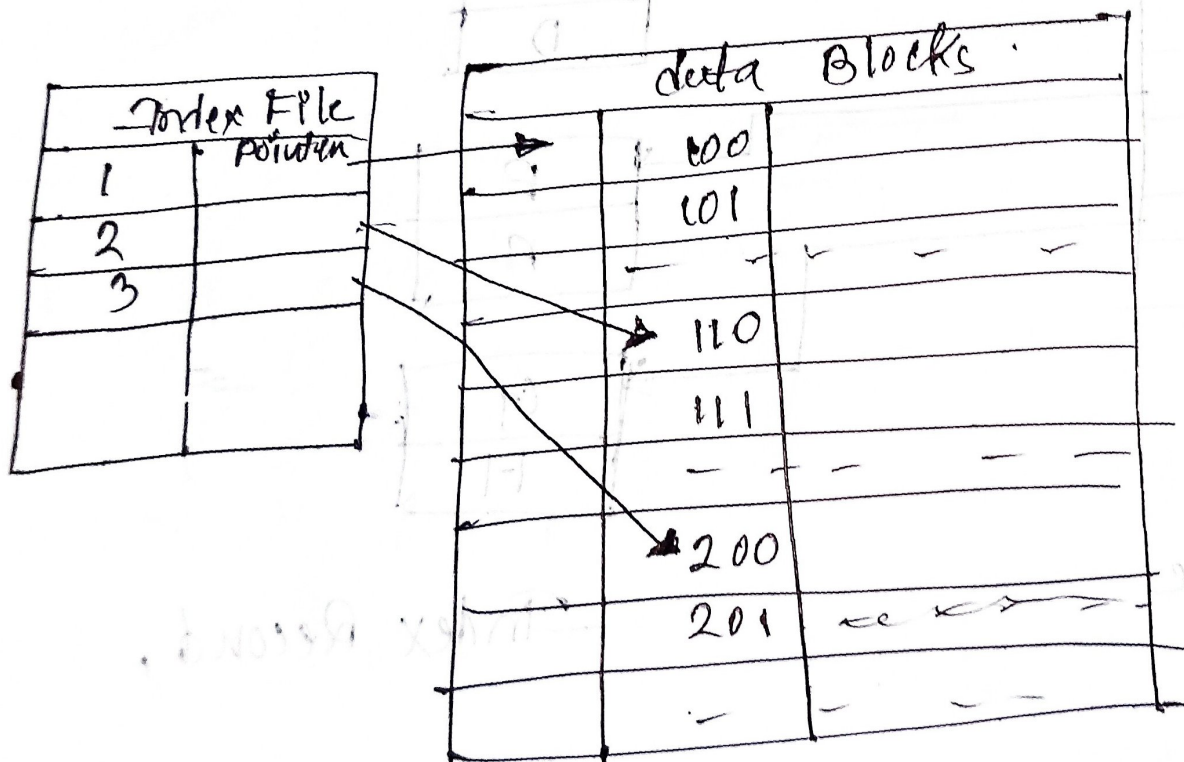


Data File

Index Record,

clustering index:-

A clustering index can be defined as an ordered data file. Sometimes the index is created on non-primary key columns which may not be unique for each record. In this case to identify the record faster we will group two or more columns to identify unique values and create index out of them. This method is called a clustering index.



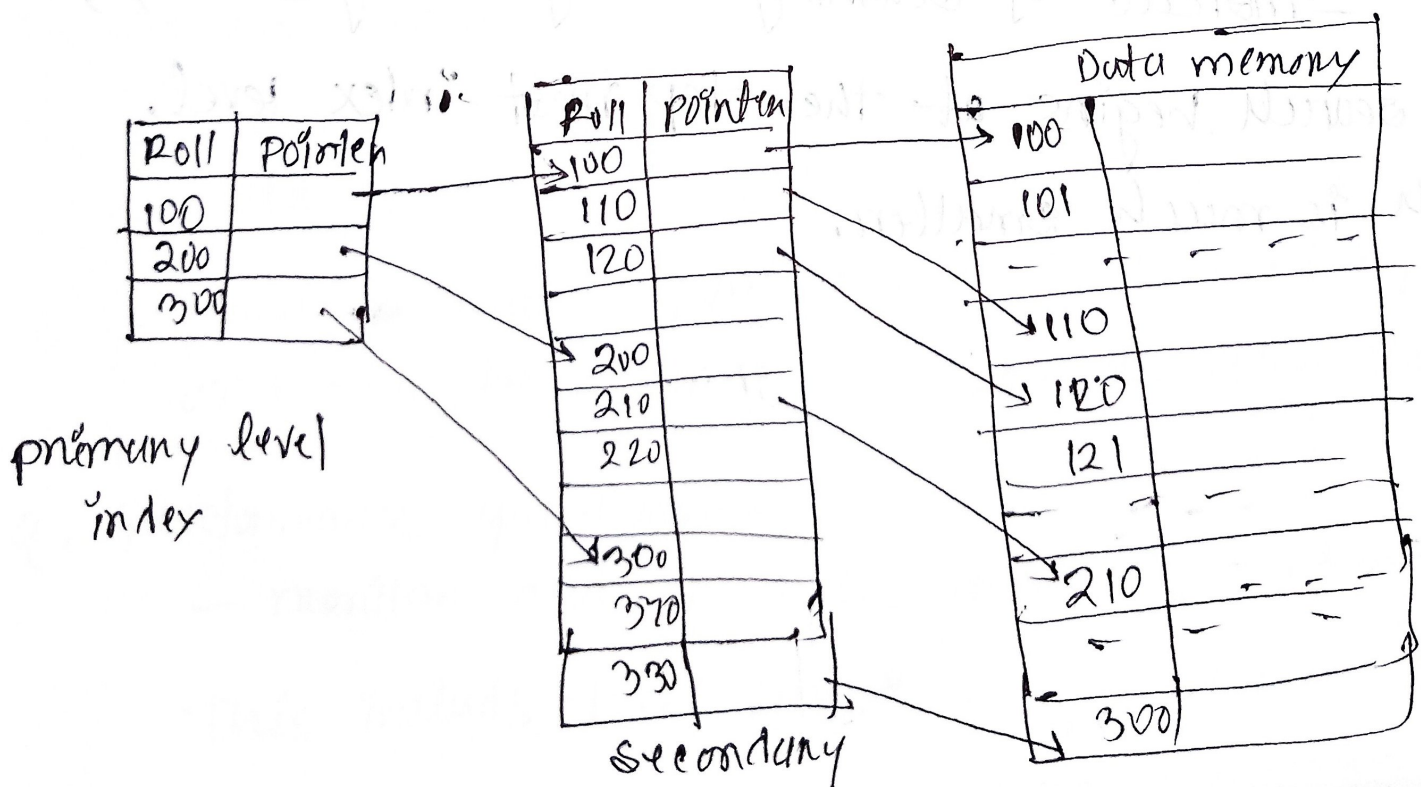
Secondary Index

In secondary indexing to reduce the size of mapping, a two-level indexing is used, with a large initial range and smaller subdivisions.

In this method, the huge range of records column is selected initially so that the mapping size of the first level becomes small.

Then each range is further divided into smaller range. The mapping of first level is stored in the primary memory, so that address fetch is faster.

The mapping of the second level and actual data are stored in the secondary memory (Hard disk)



A dense index is preferable when -

1. The dataset is small or ~~more~~ manageable in size.
2. Frequent point queries are expected.
3. The ~~ind~~ indexed column has unique values.
4. Data updates are infrequent.
5. Sorted data cannot be guaranteed.

Multilevel indexing improves the efficiency of searching an index in the following ways:-

1. Reduces search space:-

Instead of scanning a single large index, the search begins at the top most index level, which is much smaller.

Q. What is DBA ? Five main function of DBA,

⇒ A database administrator (DBA) is an individual or person responsible for controlling, maintaining, coordinating and operating a database management system.

The five main functions of a database administrator (DBA) are given below:—

1. Backup and recovery: DBAs are responsible for ensuring that data is backed up securely and can be restored in case of data loss.
2. Database design: DBAs design, create and manage database based on the needs of end users.
3. Database security and safety:
 - ensure that database system comply with data security regulations.
 - ensure that important data is backed up and can be restored in case of system failure.
4. Performance optimization:—
 - monitor and optimize database performance.

This includes tasks like indexing, optimize SQL queries.

D. Implementing data policies, procedures, standards, and resolving data conflicts.

G. Maintaining corporate data dictionaries

④

Generalization

Specialization

Generalization works in bottom-up approach.

Specialization works in top-down approach.

In generalization, the size of schema get decrease.

In specialization, the size of schema get increase.

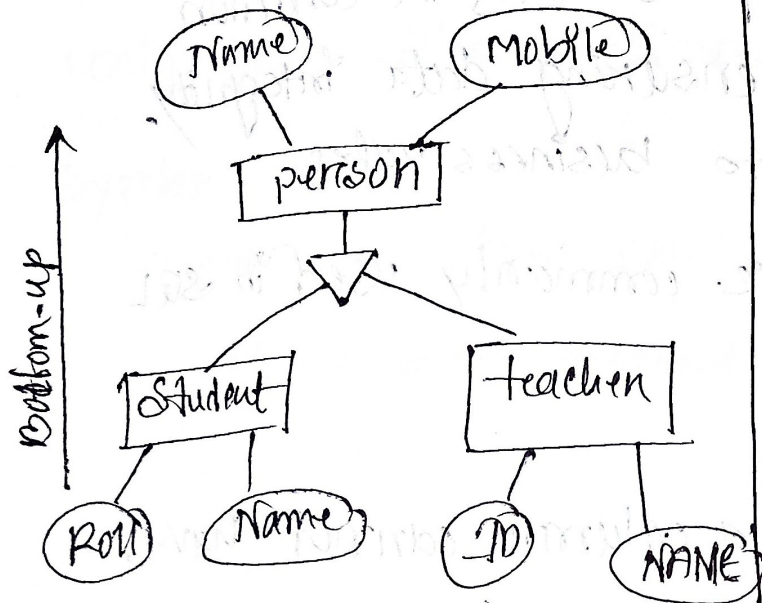
Generalization is normally applied to group of entities.

Specialization is normally applied to a single entity.

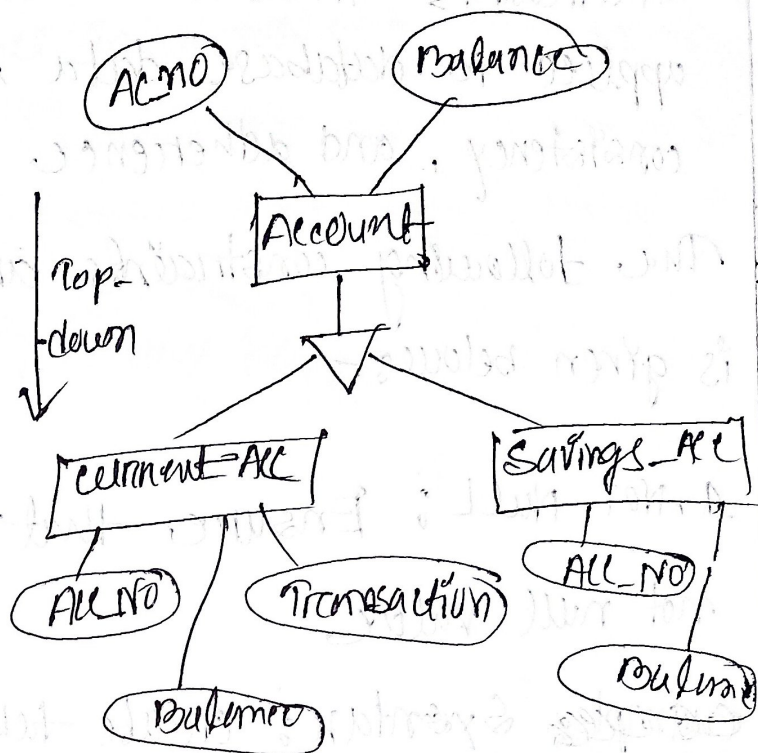
Generalization can be defined process of creating grouping from various entity sets.

Specialization can be defined process of creating sub grouping within in entity set.

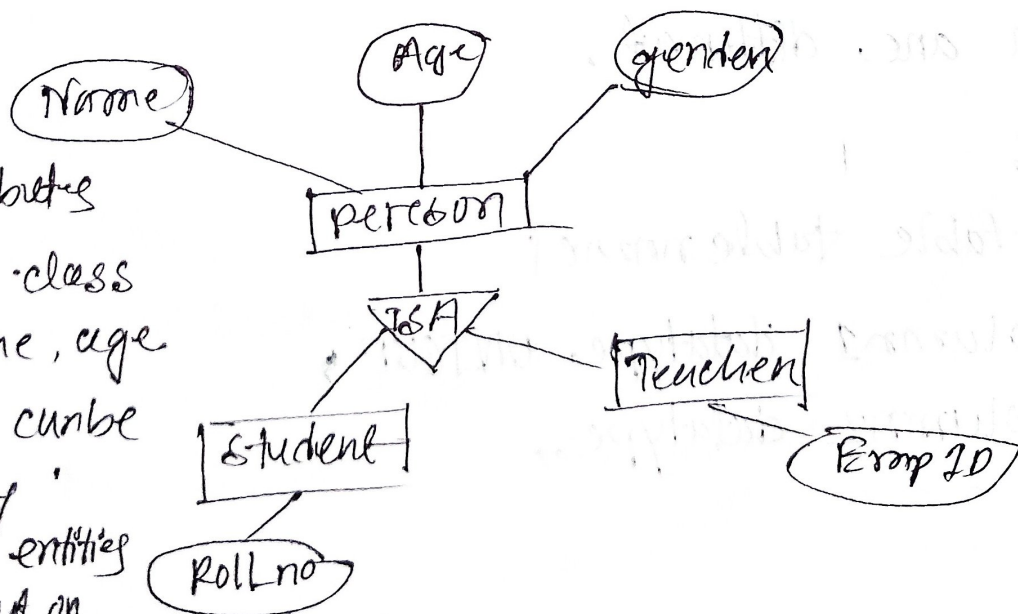
examples



Examples



Inheritance: ~~From~~ Inheritance is an important feature of generalization and specialization. It allows lower-level entities to inherit the attributes of higher-level entities.



For the attributes of a person class such as name, age and gender can be inherited by lower-level entities such as Student or Teacher.

constraints in DBMS encompass rules or condition applied to database data, ensuring data integrity, consistency, and adherence to business rules.

The following constraints are commonly used in SQL is given below:-

1. NOT NULL: Ensure that a column cannot have not null value

~~example~~ Syntax: create table table name (
column1 datatype NOT NULL,
column2 datatype,

);

2. Unique: Ensure that all the value in a column are different.

Syntax:

create table table name (
column1 datatype UNIQUE,
column2 datatype,

);

3. Primary Key: A combination of a NOT Null & unique, uniquely identifies each row in a table.

Syntax:

```
create table table-name (
```

```
column1 datatype primary key,
```

```
column2 datatype,
```

```
-----
```

```
);
```

4. Foreign Key: Column of a table, that points to the primary key of another table.

Syntax:

```
create table table-name1 (
```

```
column1 datatype primary key,
```

```
column2,
```

```
-----
```

```
);
```

```
create table table-name2 (
```

```
column1 datatype,
```

```
column2 datatype,
```

```
Foreign Key (column1) reference table-name  
(column2)
```

```
);
```


5. check: Ensures the values in a column satisfy a specific condition.

Syntax:

```
create table table-name (  
    column1 datatype,  
    column2 datatype,  
    check (condition)  
);
```

6. Default constraint: Sets a default value for a column if no value is specified.

Syntax:

```
create create table table-name (  
    column1 datatype default default_value,  
    column2 datatype,  
    -- --  
);
```

7. create index: used to create and retrieve data from the database

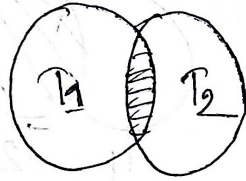
example:

very quickly

```
create index table-name  
on table-name (column1, column2, --);
```

type of join in SQLs

(i) Inner join: return records that have matching value in both tables



— Inner join

Syntax:

SELECT column_name(s)

FROM table_1

INNER JOIN table_2

ON table_1.column_name = table_2.column_name

(ii) Left join: return all records from the left table and matched records from right table

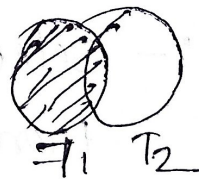
Syntax:

SELECT column_name(s)

FROM table_1

LEFT JOIN table_2

ON table_1.column_name = table_2.column_name



— Left join

Right join: returns all ~~set~~ records from the right table and matched value from the left table

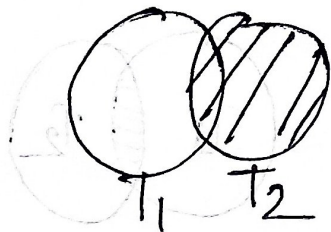
Syntax:

Select column name(s)

from table 1

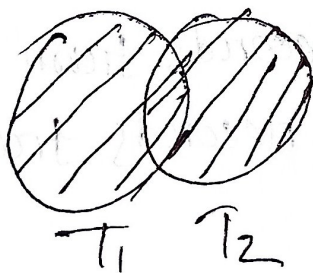
Right join table 2

ON table 1. column name = table 2. column 2;



Right join

Full join: returns all records when there is a matched ~~records from table~~ in either left or right table



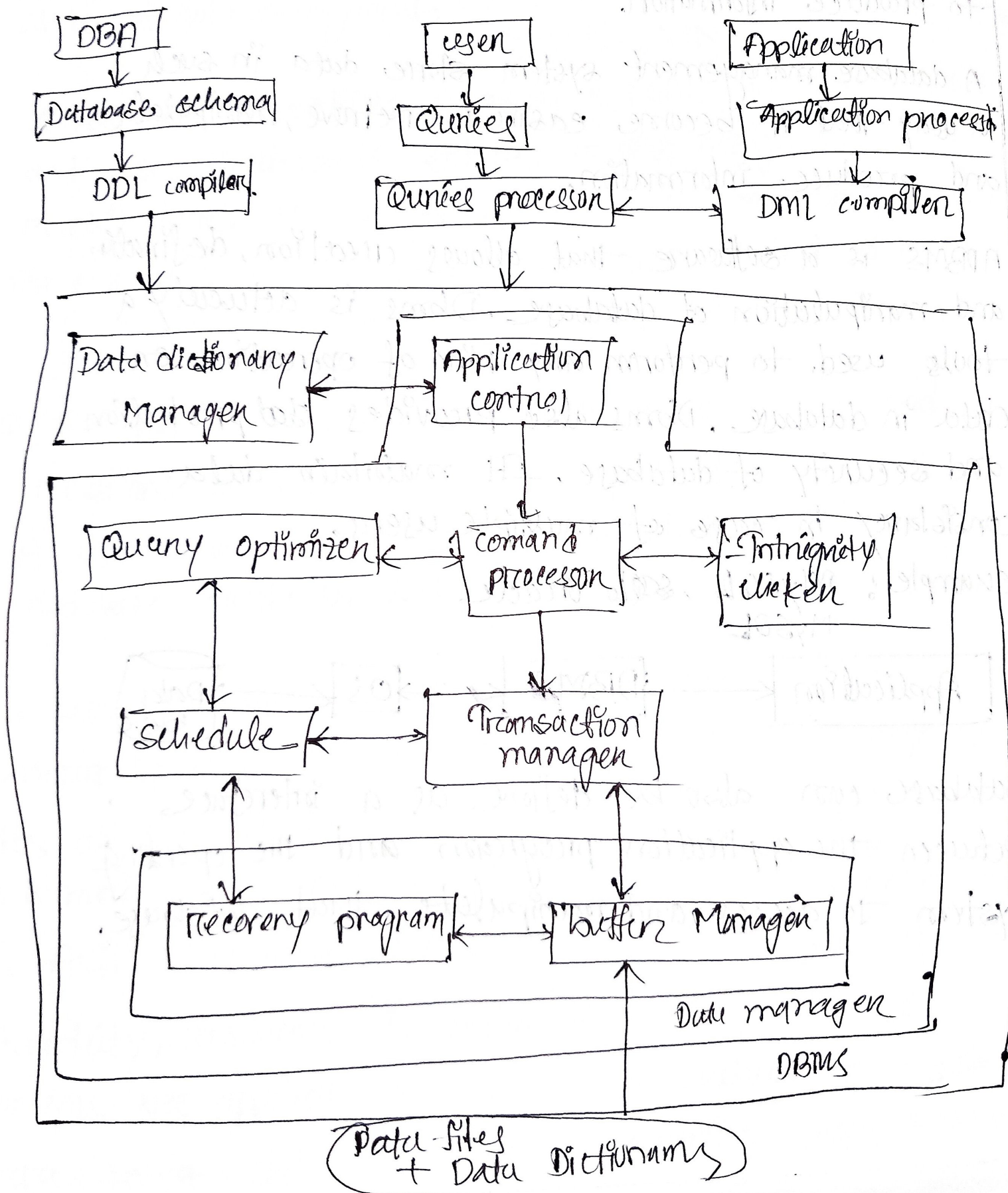
Syntax:- Select column name(s)

from table 1

Full outer join table 2

on table 1. column = table 2. column.

Draw a neat diagram of a typical components module of DBMS is given below:-

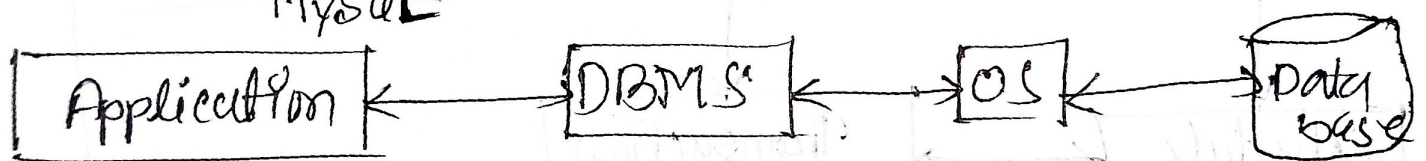


Database is a collection of related data. And data is a collection of facts and figures can be processed to produce information.

A database management system stores data in such a way that it becomes easier to retrieve, manipulate and produce information.

ADBMS is a software that allows creation, definition and manipulation of database. Dbms is actually a tool used to perform any kind of operation on data in database. Dbms also provides protection and security of database. It maintains data consistency in case of multiple users.

example: MySQL, Oracle
MySQL



Database can also be defined as an interface between the application program and the operating system to access and manipulate that database.

Traditionally, data was organized in file format. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management.

1. Real-world entity: A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behaviour and attributes too. For example: a school database may use student as an entity and their age as attributes.

2. Relation-based tables: DBMS allow entities and relations among them to form tables.

A user can understand the architecture of a database just by looking at the tables name.

3. Less redundancy: - DBMS follows the rules of normalization, which splits a relation when any of its attribute is having redundancy for values. Normalization is a mathematically rich and scientific process that reduce data redundancy.

4. ACID properties: DBMS follows the concepts of Atomicity, consistency, Isolation, durability. These concepts are applied on transaction, which manipulate data in a database. ACID properties help database

stay healthy in multi-transactional environments
and in case of failure;

Multiple views: DBMS offer multiple views for
different user.

consistency:— consistency is a state where
every relation in a database
remain consistent.

Query languages:— DBMS is equipped with query
language, which make it more
efficient of retrieve and manipulate
data.