INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Computer Science and Engineering

DBMS TUTORIAL QUESTION BANK

TUTORIAL QUESTION BANK

UNIT – I Conceptual Modeling		
PART – A (Short Answer Questions)		
Q.No	Questions	Level
1.	List the advantages of DBMS. • Improved data sharing. • Improved data security. • Better data integration. • Minimized data inconsistency. • Improved data access. • Improved decision making. • Increased end-user productivity. • Increased costs.	Understand
2.	List the database Applications. Some of the applications of DBMS are: Telecom: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Industry: Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture. Banking System: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems. Education sector: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner. Online shopping: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.	Remember
3.	Define instances and schemas of database. SCHEMA: Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema. INSTANCE: The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.	Remember
4.	Discuss Data Independence. Data independence is the type of data transparency that matters for a centralised DBMS. It refers to the immunity of user applications to changes made in the definition and organization of data. Physical data independence deals with hiding the details of the storage structure from user applications.	Understand

5.	How application programs access data base? Database is accessed for application programs for the following: ● Test database applications that you intend to store for use in a production ● environment. ● Write procedures with SPL and store them in a database. ● Use embedded SQL in programs. ● Work with user-defined and system-defined casts on extended data types. ● Embed SQL statements directly into C programs. ● Define new data types or extend the functionality of existing data types.	Remember
6.	Define (i) Database (ii) DBMS. DATABASE: A database is a collection of information that is organized so that it can be easily accessed, managed and updated. DBMS: A database-management system (DBMS) is a computer-software application that interacts with end-users, other applications, and the database itself to capture and analyze data. A general-purpose DBMSallows the definition, creation, querying, update, and administration of databases.	Remember
7.	What are main components of Database storage structure? Database storage structures: Database tables and indexes may be stored on disk in one of a number of forms, including ordered/unordered flat files, ISAM, heap files, hash buckets, or B+ trees. Each form has its own particular advantages and disadvantages. The most commonly used forms are B+ trees and ISAM. There are two types to store data in a database: Unordered storage typically stores the records in the order they are inserted. Such storage offers good insertion efficiency, but inefficient retrieval times . Typically these retrieval times are better, however, as most databases use indexes on the primary keys, resulting in retrieval times of or for keys that are the same as the database row offsets within the storage system Ordered storage typically stores the records in order and may have to rearrange or increase the file size when a new record is inserted, resulting in lower insertion efficiency. However, ordered storage provides more efficient retrieval as the records are pre-sorted, resulting in a complexity of O(log n)	Understand
8.	What are the main responsibilities of Tnsaction management component? A transaction is a logical unit of work that contains one or more SQL statements. A transaction is an atomic unit. The effects of all the SQL statements in a transaction can be either all committed (applied to the database) or all rolled back (undone from the database). A transaction begins with the first executable SQL statement. A transaction ends when it is committed or rolled back, either explicitly with a COMMIT or ROLLBACK statement or implicitly when a DDL statement is issued. To illustrate the concept of a transaction, consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction can consist of three separate operations: Decrement the savings account Increment the checking account Record the transaction in the transaction journal	Understand
9.	Outline main functions of Query Processor A query processor is one of the major components of a relational database or an electronic database in which data is stored in tables of rows and columns. It complements the storage engine, which writes and reads data to and from storage media. .	Remember
10.	Define (i) Entity (ii) Attribute ENTITY: An entity is an object that exists. It doesn't have to do anything; it just has to exist. In database administration, an entity can be a single thing, person, place, or object. Data can be stored about such entities. A design tool that allows database administrators to view the relationships between several entities is called the entity relationship diagram (ERD).	Remember

	ATTRIBUTE: An attribute defines the information about the entity that needs to be stored. If the entity is an employee, attributes could include name, employee ID, health plan enrollment, and work location. An entity will have zero or more attributes, and each of those attributes apply only to that entity. For example, the employee ID of 123456 belongs to that employee entity alone.	
11.	Define Relationship and Relationship set. RELATION: In relational database theory, a relation, as originally defined by E. F. Codd, is a set of tuples (d1, d2, ..., dn), where each element dj is a member of Dj, a data domain. Codd's original definition notwithstanding, and contrary to the usual definition in mathematics, there is no ordering to the elements of the tuples of a relation.[2][3] Instead, each element is termed an attribute value. RELATIONSHIP SET: A relationship set is a set of relationships of the same type. A relationship is an association between several entities.	Remember
12.	Discuss about Data Definition language. Data Definition Language (DDL) is a standard for commands that define the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP. 1. CREATE - to create objects in the database 2. ALTER - alters the structure of the database 3. DROP - delete objects from the database 4. TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed 5. COMMENT - add comments to the data dictionary 6. RENAME - rename an object	Understand
13.	Discuss about Data Manipulation language. A data manipulation language (DML) is a computer programming language used for adding (inserting), deleting, and modifying (updating) data in a database. A DML is often a sublanguage of a broader database language such as SQL, with the DML comprising some of the operators in the language. 1. SELECT - retrieve data from the a database 2. INSERT - insert data into a table UPDATE - updates existing data within a table 3. DELETE - deletes all records from a table, the space for the records remain 4. MERGE - UPSERT operation (insert or update) 5. CALL - call a PL/SQL or Java subprogram 6. EXPLAIN PLAN - explain access path to data 7. LOCK TABLE - control concurrency	Remember
14.	List responsibilities of a DBA. DATABASE ADMINISTRATOR: A database administrator (DBA) directs or performs all activities related to maintaining a successful database environment. RESPONSIBILITIES OF DBA: Database administrators (DBAs) use specialized software to store and organize data. The role may include capacity planning, installation, configuration, database design, migration, performance monitoring, security, troubleshooting, as well as backup and data recovery.	Remember
15.	Outline the History of Database Systems. A Database Management System allows a person to organize, store, and retrieve data from a computer. It is a way of communicating with a computer's "stored memory." In the very early years of computers, "punch cards" were used for input, output, and data storage. Punch cards offered a fast way to enter data, and to retrieve it. Herman Hollerith is given credit for adapting the punch cards used for weaving looms to act as the memory for a mechanical tabulating machine, in 1890. Databases (or DBs) have played a very important part in the recent evolution of computers.	Understand

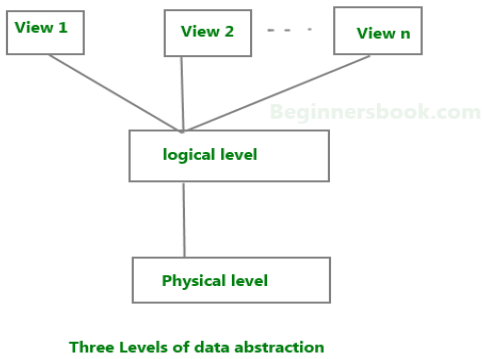
	The first computer programs were developed in the early 1950s, and focused almost completely on coding languages and algorithms. At the time, computers were basically giant calculators and data (names, phone numbers) was considered the leftovers of processing information. Computers were just starting to become commercially available, and when business people started using them for real-world purposes, this leftover data suddenly became important.	
16.	Discuss how can you change the data in the table. The UPDATE statement can change data values in single rows, groups of rows, or all the rows in a table or view. It can also be used to update rows in a remote server by using either a linked server name or the OPENROWSET, OPENDATASOURCE, and OPENQUERY functions, as long as the OLE DB provider used to access the remote server supports updates. An UPDATE statement referencing a table or view can change the data in only one base table at a time. The UPDATE statement has the following main clauses: (i) Set (ii) From (iii) Where	Understand
17.	List various types of attributes. Single valued Attributes : An attribute, that has a single value for a particular entity is known as single valued attributes. For example, age of a employee entity. Multi valued Attributes : An attributes that may have multiple values for the same entity is known as multi valued attributes. For example colors of a car entity. Compound Attribute/Composite Attribute : Attribute can be subdivided into two or more other Attribute. For Example, Name can be divided into First name, Middle name and Last name. Simple Attributes/Atomic Attributes : The attributes which cannot be divided into smaller subparts are called simple or atomic attributes. For example, age of employee entity Stored Attribute : An attribute, which cannot be derived from other attribute, is known as stored attribute. For example, BirthDate of employee. Derived Attribute : Attributes derived from other stored attribute. For example age from Date of Birth and Today's date. Complex Attributes : If an attribute fr an entity, is built using composite and multivalued attributes, then these attributes are called complex attributes. For example, a person can have more than one residence and each residence can have multiple phones, an addressphone for a person entity can be specified as – {Addressphone (phone {(Area Code, Phone Number)}, Address(Sector Address (Sector N umber,House Number), City, State, Pin))} Here {} are used to enclose multivalued attributes and () are used to enclose composite attributes with comma separating individual attributes	Remember
18.	Discuss How can you alter and destroy tables?. After you create a table, you might decide to change a column name, add a column, delete a column or change the table in some way. Changes to tables use the ALTER keyword. The following example adds a "CreateDate" to the Product table. ALTER TABLE Product add(goods varchar2(10)); [adding new attribute] We can destroy tables by using drop and delete commands as follows: Alter table product drop(goods) [deletes one complete column] Drop table product; [deletes complete table]	Remember
19.	Define a data model? List the types of data model used. A database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed. Individual database models are designed based on the rules and concepts of whichever broader data model the designers adopt. Most data models can be represented by an accompanying database diagram.	Understand

	Types of database models There are many kinds of data models. Some of the most common ones include: Hierarchical database model Relational model Network model Object-oriented database model Entity-relationship model Document model Entity-attribute-value model Star schema The object-relational model, which combines the two that make up its name	
20.	List the levels of data abstraction. Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction. We have three levels of abstraction: Physical level: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level. Logical level: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database. View level: Highest level of data abstraction. This level describes the user interaction with database system.	Understand
	PART – B (Long Answer Questions)	
1.	Compare and Contrast file Systems with database systems.	Understand

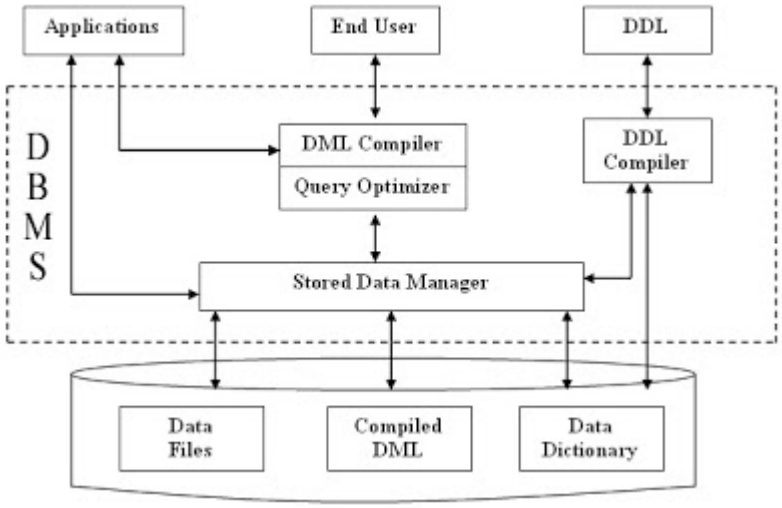
File Management System	Database Management System
File System is a general, easy-to-use system to store general files which require less security and constraints.	Database management system is used when security constraints are high.
Data Redundancy is more in file management system.	Data Redundancy is less in database management system.
Data Inconsistency is more in file system.	Data Inconsistency is less in database management system.

	Centralisation is hard to get when it comes to File Management System.	Centralisation is achieved in Database Management System.	
	User locates the physical address of the files to access data in File Management System.	In Database Management System, user is unaware of physical address where data is stored.	
	Security is low in File Management System.	Security is high in Database Management System.	
	File Management System stores unstructured data as isolated data files/entities.	Database Management System stores structured data which have well defined constraints and interrelation.	

These are the criteria which should taken into account when choosing a system to manage

2.	Define Data Abstraction and discuss levels of Abstraction. Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction. We have three levels of abstraction: Physical level: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level. Logical level: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database. View level: Highest level of data abstraction. This level describes the user interaction with database system.  Three Levels of data abstraction	Remember

3.	**Discuss about different types of Data models.** A Data Model is a logical structure of Database. It is a collection of concepts for describing data, reflects entities, attributes, relationship among data, constrains etc. A schema is a description of a particular collection of data, using the given data model. Relational model: The most common model, the relational model sorts data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity in question, such as price, zip code, or birth date. Together, the attributes in a relation are called a domain. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, when it's called a foreign key. Each row, also called a tuple, includes data about a specific instance of the entity in question, such as a particular employee. The model also accounts for the types of relationships between those tables, including one-toone, one-to-many, and many-to-many relationships. Here's an example: Hierarchical model The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root. Sibling records are sorted in a particular order. That order is used as the physical order for storing the database. This model is good for describing many real-world relationships. Network model The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records. Based on mathematical set theory, the model is constructed with sets of related records. Each set consists of one owner or parent record and one or more member or child records. A record can be a member or child in multiple sets, allowing this model to convey complex relationships. It was most popular in the 70s after it was formally defined by the Conference on Data Systems Languages (CODASYL). Object-oriented database model This model defines a database as a collection of objects, or reusable software elements, with associated features and methods. There are several kinds of object-oriented databases: A multimedia database incorporates media, such as images, that could not be stored in a relational database. A hypertext database allows any object to link to any other object. It's useful for organizing lots of disparate data, but it's not ideal for numerical analysis. The object-oriented database model is the best known post-relational database model, since it incorporates tables, but isn't limited to tables. Such models are also known as hybrid database models. Object-relational model This hybrid database model combines the simplicity of the relational model with some of the advanced functionality of the object-oriented database model. In essence, it allows designers to incorporate objects into the familiar table structure. Languages and call interfaces include SQL3, vendor languages, ODBC, JDBC, and proprietary call interfaces that are extensions of the languages and interfaces used by the relational model. Entity-relationship model This model captures the relationships between real-world entities much like the network model, but it isn't as directly tied to the physical structure of the database. Instead, it's often used for designing a database conceptually. Here, the people, places, and things about which data points are stored are referred to as entities, each of which has certain attributes that together make up their domain. The cardinality, or relationships between entities, are mapped as well.	Remember
4.	**Describe the Structure of DBMS.** Since DBMS is responsible to store huge amount of data and is capable of handling multiple requests from users simultaneously, it should be arranged properly. One can imagine a database as a brain! How is the structure of brain? Bit sophisticated and each part of the brain is responsible for some specific tasks. Similarly, Database is also designed. At very high level, a database is considered as shown in below diagram. Let us see them in detail below.	Understand



Applications: - It can be considered as a user friendly web page where the user enters the requests. Here he simply enters the details that he needs and presses buttons to get the data.
End User: - They are the real users of the database. They can be developers, designers, administrator or the actual users of the database.
DDL: - Data Definition Language (DDL) is a query fired to create database, schema, tables, mappings etc in the database. These are the commands used to create the objects like tables, indexes in the database for the first time. In other words, they create structure of the database.
DDL Compiler: - This part of database is responsible for processing the DDL commands. That means these compiler actually breaks down the command into machine understandable codes. It is also responsible for storing the metadata information like table name, space used by it, number of columns in it, mapping information etc.
DML Compiler: - When the user inserts, deletes, updates or retrieves the record from the database, he will be sending request which he understands by pressing some buttons. But for the database to work/understand the request, it should be broken down to object code. This is done by this compiler. One can imagine this as when a person is asked some question, how this is broken down into waves to reach the brain!
Query Optimizer: - When user fires some request, he is least bothered how it will be fired on the database. He is not all aware of database or its way of performance. But whatever be the request, it should be efficient enough to fetch, insert, update or delete the data from the database. The query optimizer decides the best way to execute the user request which is received from the DML compiler. It is similar to selecting the best nerve to carry the waves to brain!
Stored Data Manager: - This is also known as Database Control System. It is one the main central system of the database. It is responsible for various tasks.
Data Files: - It has the real data stored in it. It can be stored as magnetic tapes, magnetic disks or optical disks.
Compiled DML: - Some of the processed DML statements (insert, update, delete) are stored in it so that if there is similar requests, it will be re-used.
Data Dictionary: - It contains all the information about the database. As the name suggests, it is the dictionary of all the data items. It contains description of all the tables, view, materialized views, constraints, indexes, triggers etc.

5.	Discuss additional features of the ER-Models. Basic E-R model is good for many uses In this section, we discuss the extended E-R features of specialization, generalization, higherand lower-level entity sets, attribute inheritance, and aggregation. • Generalization: a “bottom up” approach – Taking similar entity-sets and unifying their common features – Start with specific entities, then create generalizations from them • Specialization: a “top down” approach – Creating general purpose entity-sets, then	Remember

	providing specializations of the general idea – Start with general notion, then refine it. Attribute Inheritance: A crucial property of the higher- and lower-level entities created by specialization and generalization is attribute inheritance. The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets. For example, customer and employee inherit the attributes of person. A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates. Aggregation: Aggregation is an abstraction through which relationships are treated as higher- level entities. Thus, for our example, we regard the relationship set works-on (relating the entity sets employee, branch, and job) as a higher-level entity set called works-on.	
6.	Discuss about the Concept Design with the ER Model. **Conceptual Database Design - Entity Relationship(ER) Modeling** **Database Design Techniques** 1. ER Modeling (Top down Approach) 2. Normalization (Bottom Up approach) **What is ER Modeling?** A graphical technique for understanding and organizing the data independent of the actual database implementation We need to be familiar with the following terms to go further. **Entity** Anything that has an independent existence and about which we collect data. It is also known as entity type. In ER modeling, notation for entity is given below.  **Entity instance** Entity instance is a particular member of the entity type. Example for entity instance : A particular employee **Regular Entity** An entity which has its own key attribute is a regular entity. Example for regular entity : Employee. **Weak entity** An entity which depends on other entity for its existence and doesn't have any key attribute of its own is a weak entity. Example for a weak entity : In a parent/child relationship, a parent is considered as a strong entity and the child is a weak entity. In ER modeling, notation for weak entity is given below.	Remember

Attributes

Properties/characteristics which describe entities are called attributes.
In ER modeling, notation for attribute is given below.

Domain of Attributes

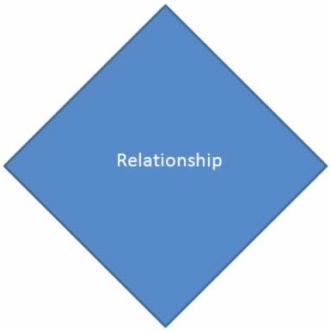
The set of possible values that an attribute can take is called the domain of the attribute. For example, the attribute day may take any value from the set {Monday, Tuesday ... Friday}. Hence this set can be termed as the domain of the attribute day.

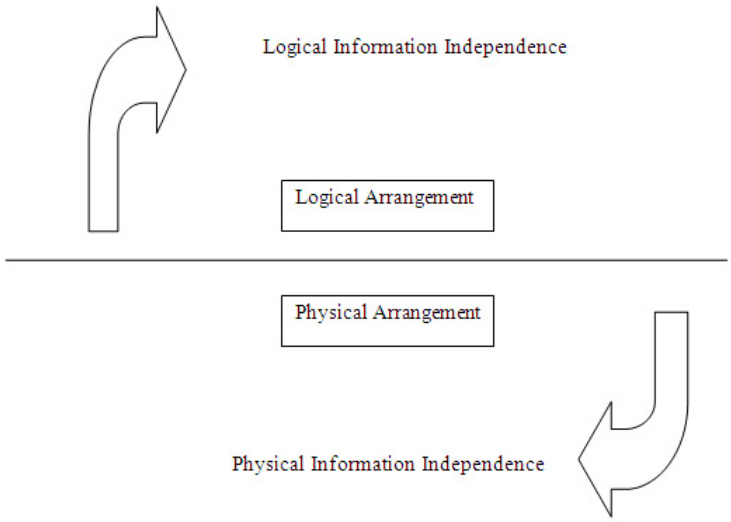
Key attribute

The attribute (or combination of attributes) which is unique for every entity instance is called key attribute.
E.g the employee_id of an employee, pan_card_number of a person etc.If the key attribute consists of two or more attributes in combination, it is called a composite key.
In ER modeling, notation for key attribute is given below.

Relationships

Associations between entities are called relationships
Example : An employee works for an organization. Here "works for" is a relation between the entities employee and organization.
In ER modeling, notation for relationship is given below.



7.	**Explain in detail Different types of Data Independence with examples.** Definition: The acquired skill to change a conceptual pattern by not altering the conceptual pattern of the next superior level is defined as the data independence. The conventional data processing does not provide data independence in application programs. So, any kind of changes in the information, layouts, or arrangements need the change in application programs also.  Fig 1: Conventional data processing without data independence Types of Data independence The data independency is classified as two types and they are as below: Logical Data Independence Physical Data Independence The diagrammatic representation of the logical and physical data independence is as a shown below:  Fig 3: Types of data independence Logical Data Independence Logical data independence points out that the conceptual pattern can be altered by non damaging the current external patterns or schemas. The external level and conceptual level has mapping in between them and it takes all the made alterations. It also protects and isolates application programs from actions like combination of dual records into a single record or separating a single record into two or more records. Physical Data Independence Physical data independence points out the physical storing patterns changes by non damaging conceptual structures or arrangements. The presence of internal level in the architecture of database and the operation of changes from the conceptual level to internal level achieves the physical data independence.	Understand
8.	**Explain different types of database users and write the functions of DBA.** TYPES OF DATABASE USERS: Database users are the one who really use and take the benefits of database. There will be different types of users depending on their need and way of accessing the database. 1. Application Programmers - They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program. 2. Sophisticated Users - They are database developers, who write SQL queries to	Understand

	select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL. 3. Specialized Users - These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement. 4. Stand-alone Users - These users will have stand –alone database for their personal use. These kinds of database will have readymade database packages which will have menus and graphical interfaces. 5. Native Users - these are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests. Database Administrator Coordinates all the activities of the database system Has a good understanding of the enterprise's information resources and needs. Database administrator's duties include: – Storage structure and access method definition Schema and physical organization modification Granting users authority to access the database Backing up data Monitoring performance and responding to changes Database tuning.	
9.	Explain about different types of integrity constraints. TYPES OF INTEGRITY CONSTRAINTS: Various types of integrity constraints are 1. Domain Integrity 2. Entity Integrity Constraint 3. Referential Integrity Constraint 4. Key Constraints 1. Domain constraints • Domain constraints can be defined as the definition of a valid set of values for an attribute. • The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain. **Example:** **ID** | **NAME** | **SEMENSTER** | **AGE** 1000 | Tom | 1st | 17 1001 | Johnson | 2nd | 24 1002 | Leonardo | 5th | 21 1003 | Kate | 3rd | 19 1004 | Morgan | 8th | A Not allowed. Because AGE is an integer attribute 2. Entity integrity constraints • The entity integrity constraint states that primary key value can't be null. • This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.	Remember

- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

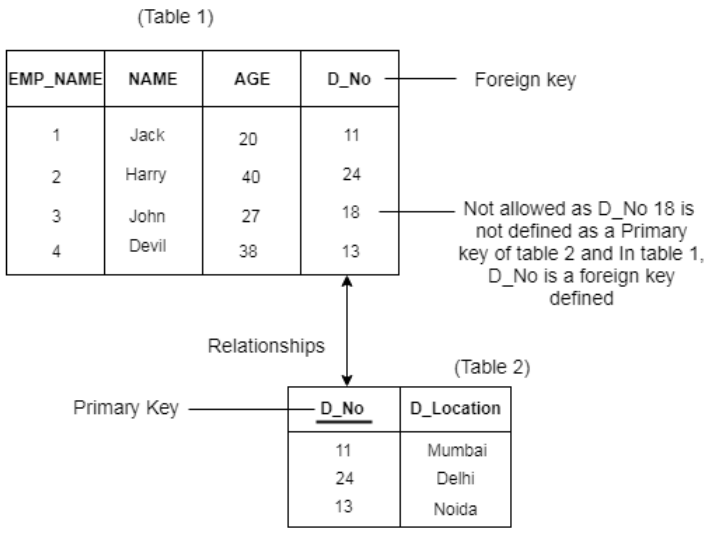
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:



4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

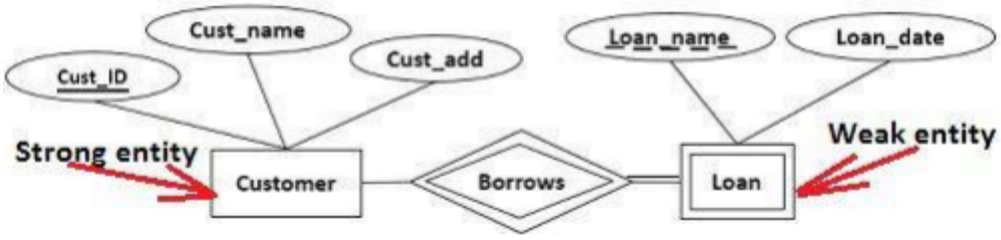
Not allowed. Because all row must be unique

10.	Discuss about Different keys used in database design with examples. Database supports the following types of keys. Super Key Candidate Key Primary Key Unique Key Foreign Key A Candidate key is an attribute or set of attributes that uniquely identifies a record. Among the set of candidate, one candidate key is chosen as Primary Key. So a table can have multiple candidate key but each table can have maximum one primary key. Example: Possible Candidate Keys in Branch_Info table. 1. Branch_Id 2. Branch_Name 3. Branch_Code Possible Candidate keys in Student_Information table. 1. Student_Id 2. College_Id 3. Rtu_Roll_No Primary Key A Primary key uniquely identifies each record in a table and must never be the same for the 2 records. Primary key is a set of one or more fields (columns) of a table that uniquely identify a record in database table. A table can have only one primary key and one candidate key can select as a primary key. The primary key should be chosen such that its attributes are never or rarely changed, for example, we can't select Student_Id field as a primary key because in some case Student_Id of student may be changed. Example: Primary Key in Branch_Info table: 1. Branch_Id Primary Key in Student_Information Table: 1. College_Id Unique Key: A unique key is a set of one or more attribute that can be used to uniquely identify the records in table. Unique key is similar to primary key but unique key field can contain a "Null" value but primary key doesn't allow "Null" value. Other difference is that primary key field contain a clustered index and unique field contain a non-clustered index. Example: Possible Unique Key in Branch_Info table. 1. Branch_Name Possible Unique Key in Student_Information table: 1. Rtu_Roll_No Super Key Super key is a set of on e or more than one keys that can be used to uniquely identify the record in table. A Super key for an entity is a set of one or more attributes whose combined value uniquely identifies the entity in the entity set. A super key is a combine form of Primary Key, Alternate key and Unique key and Primary Key, Unique Key and Alternate Key are subset of super key. A Super Key is simply a non-minimal Candidate Key, that is to say one with additional columns not strictly required to ensure uniqueness of the row. A super key can have a single column. Example: Super Keys in Branch_Info Table. 1. Branch_Id 2. Branch_Name 3. Branch_Code 4. { Branch_Id, Branch_Code } 5. { Branch_Name , Branch_Code } Super Keys in Student_Information Table: 1. Student_Id 2. College_Id 3. Rtu_Roll_No 4. { Student_Id, Student_Name} 5. { College_Id, Branch_Id }	Remember

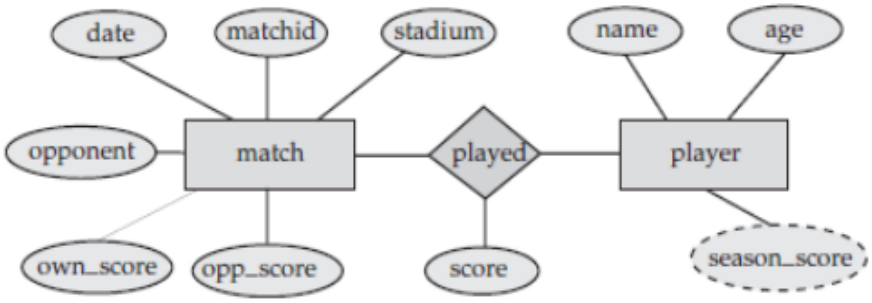
	6. { Rtu_Roll_No, Session }	
11.	Distinguish strong entity set with weak entity set? Draw an ER diagram to illustrate weak entity set.	Understand

BASIS FOR COMPARISON	STRONG ENTITY	WEAK ENTITY
Basic	The Strong entity has a primary key.	The weak entity has a partial discriminator key.
Depends	The Strong entity is independent of any other entity in a schema.	Weak entity depends on the strong entity for its existence.
Denoted	Strong entity is denoted by a single rectangle.	Weak entity is denoted with the double rectangle.
Relation	The relation between two strong entities is denoted by a single diamond simply called relationship.	The relationship between a weak and a strong entity is denoted by Identifying Relationship denoted with double diamond.
Participation	Strong entity may or may not have total participation in the relationship.	Weak entity always has total participation in the identifying relationship shown by double line.

ER diagram to illustrate Weak entity:



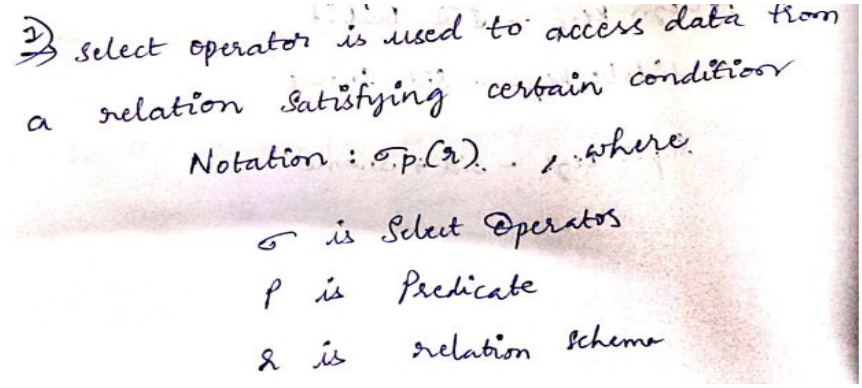
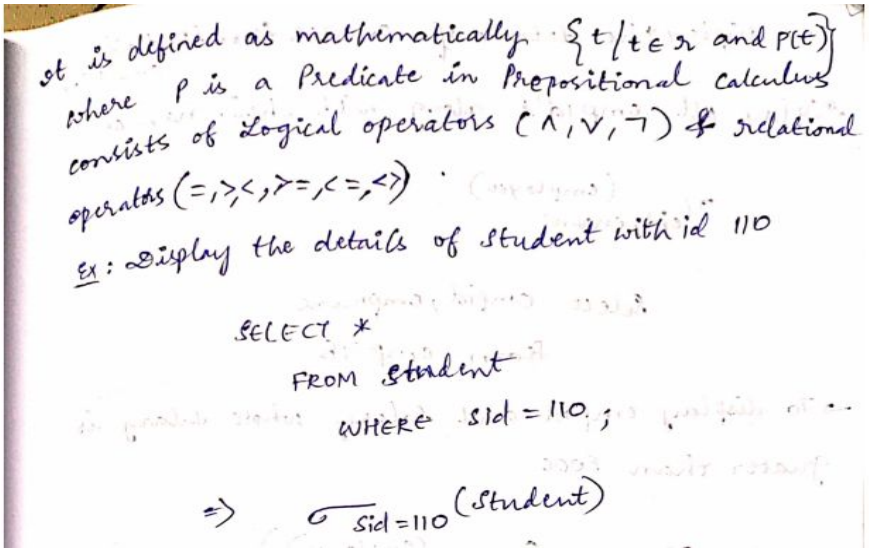
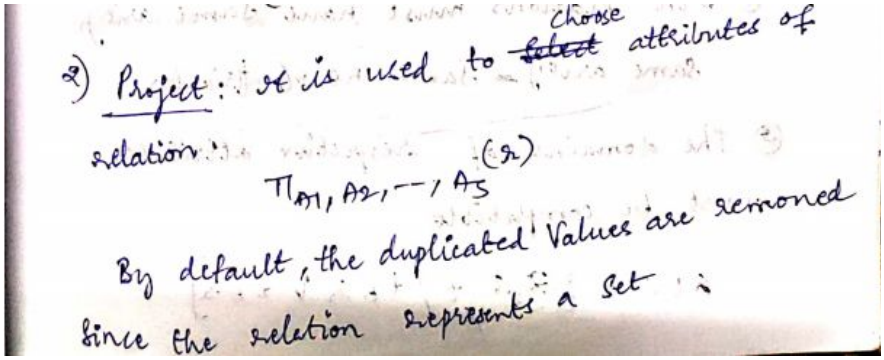
12.	Differentiate relation schema and relational instance. Define the terms arity and degree of s relation? What are domain constraints. Relation Schema : Relation schema defines the design and structure of the relation like it consists of the relation name, set of attributes/field names/column names. every attribute would have an associated domain. For example : Student is a schema : Student { name : String, rollnumber : String, contactnumber : Integer, yearofadmission : Integer, course : String} Relation Instance : Relation instance is the snapshot of the relation at a particular time. It consists of the data which is present inside the relation at a particular instance of time. Duplicate row entries are not allowed in the relation instance. Though in SQL, duplicate rows are allowed in the table. ARITY: In computer science, the arity of a function or operation is the number of arguments or operands that the function takes. The arity of a relation (or predicate) is the dimension of the domain in the corresponding Cartesian product. (A function of arity n thus has arity n+1 considered as a relation.) The term springs from words like unary, binary, ternary, etc. Unary functions or predicates may be also called "monadic"; similarly, binary functions may be called "dyadic". DOMAIN CONSTRAINTS: Domain constraints are user defined data type and we can define them like this: Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT) Example: For example I want to create a table “student_info” with “stu_id” field having value greater than 100, I can create a domain and table like this: *create domain id_value int* *constraint id_test* *check(value > 100);* *create table student_info (* *stu_id id_value PRIMARY KEY,* *stu_name varchar(30),* *stu_age int* *);*	Understand	CACS005.06
13.	List and explain the design issues of entity relationship. 1. Use of Entity set vs. Attributes In the real world situations, sometimes it is difficult to select the property as an attribute or an entity set 2. Use of Entity sets vs. Relationship sets Sometimes, an entity set can be better expressed in relationship set. Thus, it is not always clear whether an object is best expressed by an entity set or a relationship set 3. Binary vs. n-ary relationship sets Relationships in databases are often binary. Some relationships that appear to be non-binary could actually be better represented by several binary relationships 4. Placement of Relationship Attributes The cardinality ratio of a relationship can affect the placement of relationship attributes: • One-to-Many: Attributes of 1:M relationship set can be repositioned to only the entity set on the many side of the relationship	Remember	CACS005.04

	• One-to-One: The relationship attribute can be associated with either one of the participating entities • Many-to-Many: Here, the relationship attributes can not be represented to the entity sets; rather they will be represented by the entity set to be created for the relationship set		
14.	Develop ER-Diagram for a hospital with a set of patients and a set of medical doctors. Associated with each patient a log of the various tests and examinations conducted. ##########################	Remember	CACS005.05
PART – C (Problem Solving and Critical Thinking Questions)			
1	Design an E-R diagram for keeping track of the exploits of your favorite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modeled as derived attributes.  	Remember	CACS005.05
2	Let E1 and E2 be two entities in an E/R diagram with simple single-valued attributes. R1 and R2 are two relationships between E1 and E2, where R1 is one-to-many and R2 is many-to-many. R1 and R2 do not have any attributes of theirown. Calculate the minimum number of tables required to represent this situation in the relational model. Explanation: The answer is B, i.e minimum 3 tables. Strong entities E1 and E2 are represented as separate tables. In addition to that many-to-many relationships(R2) must be converted as seperate table by having primary keys of E1 and E2 as foreign keys. One-to-many relationship (R1) must be transferred to 'many' side table(i.e. E2) by having primary key of one side(E1) as foreign key(this way we need not to make a seperate table for R1). Let relation schema be E1(a1,a2) and E2(b1,b2). Relation E1(a1 is the key) a1 a2 ------- 1 3 2 4 3 4 Relation E2(b1 is the key, a1 is the foreign key, hence R1(one-many) relationship set satisfy here) b1 b2 a1 ----------- 7 4 2 8 7 2 9 7 3 Relation R2 ({a1, b1} combined is the key here , representing many-many relationship R2) a1 b1	Understand	CACS005.05

	-------- 1 7 1 8 2 9 3 9 Hence we will have minimum of 3 tables.		
3	Analyze and find whether modifications made at conceptual level makes application programs written by users at viewlevel to be modified in a database. Analyze your anser with illustration.	Remember	CACS005.05
4	We can convert any weak entity set to strong entity set by simply adding appropriate attributes. Analyze why, then, do we have weak entity sets? ● We want to avoid the data duplication and consequent possible inconsistencies ● caused by duplicating the key of the strong entity. ● Weak entities reflect the logical structure of an entity being dependent on ● another entity. ● Weak entities can be deleted automatically when their strong entity is deleted. ● Weak entities can be stored physically with their strong entities.	Understand	CACS005.04
5	What are the responsibilities of a DBA? If we assume that the DBA is never interested in running his or her own queries, does the DBA still need to understand query optimization? Why? Designing the logical and physical schemas, as well as widely-used portions of the external schema. Security and authorization. Data availability and recovery from failures. Database tuning: The DBA is responsible for evolving the database, in particular the conceptual and physical schemas, to ensure adequate performance as user requirements change. A DBA needs to understand query optimization even if s/he is not interested in running his or her own queries because some of these responsibilities (database design and tuning) are related to query optimization. Unless the DBA understands the performance needs of widely used queries, and how the DBMS will optimize and execute these queries, good design and tuning decisions cannot be made.	Remember	CACS005.02
6	Describe the structure of a DBMS. If your operating system is upgraded to support some new functions on OS files (e.g., the ability to force some sequence of bytes to disk), which layer(s) of the DBMS would you have to rewrite to take advantage of these new functions. The architecture of a relational DBMS typically consists of a layer that manages space on disk, a layer that manages available main memory and brings disk pages into memory as needed, a layer that supports the abstractions of files and index structures, a layer that implements relational operators, and a layer that parses and optimizes queries and produces an execution plan in terms of relational operators. In addition, there is support for concurrency control and recovery, which interacts with the buffer management and access method layers. The disk space management layer has to be rewritten to take advantage of the new functions on OS files. It is likely that the buffer management layer will also be affected.	Remember	CACS005.03

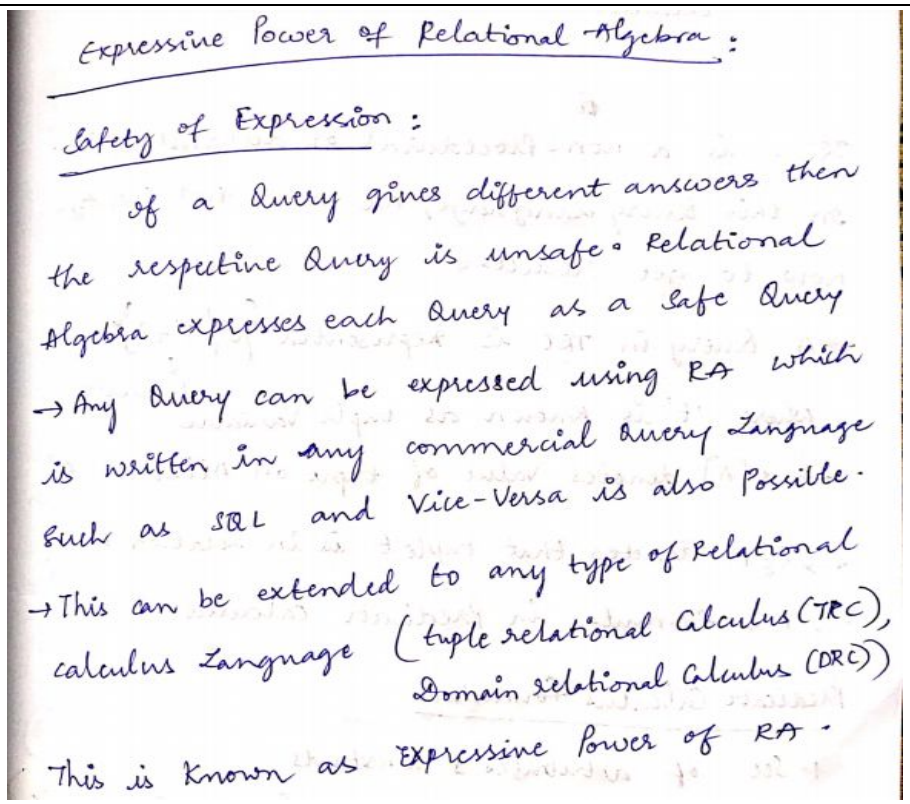
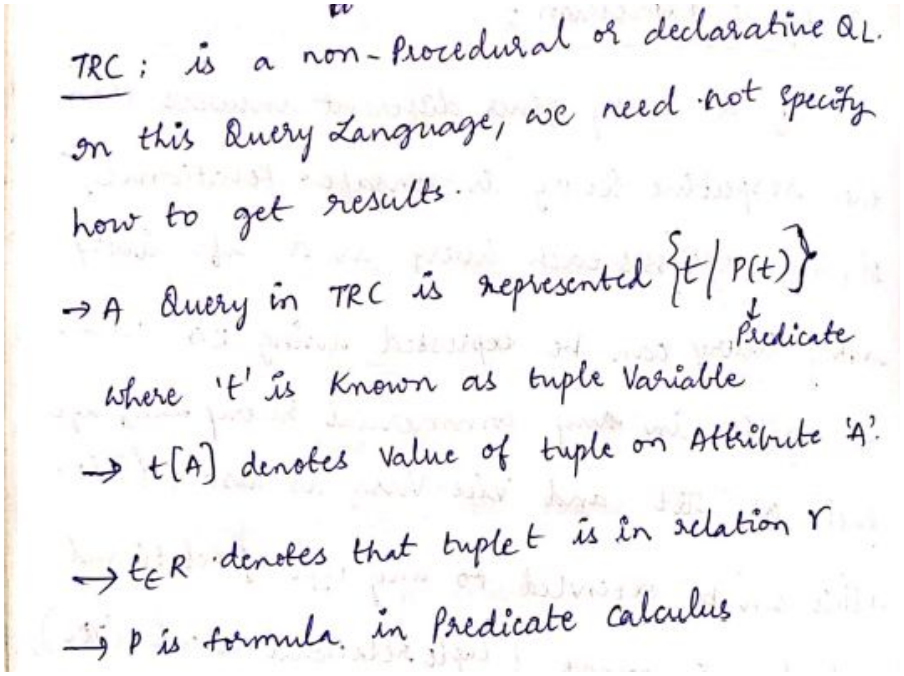
7	Why relational model became more popular comparing with other record based models? 1.A transaction is any one execution of a user program in a DBMS. This is the basic unit of change in a DBMS. 2. A DBMS is typically shared among many users. Transactions from these users can be interleaved to improve the execution time of users' queries. By interleaving queries, users do not have to wait for other user's transactions to complete fully before their own transaction begins. Without interleaving, if user A begins a transaction that will take 10 seconds to complete, and user B wants to begin a transaction, user B would have to wait an additional 10 seconds for user A's transaction to complete before the database would begin processing user B's request.	Understand	CACS005.03
8	Describe the process to convert ER model into relational schema. A user must guarantee that his or her transaction does not corrupt data or insert nonsense in the database. For example, in a banking database, a user must guarantee that a cash withdraw transaction accurately models the amount a person removes from his or her account. A database application would be worthless if a person removed 20 dollars from an ATM but the transaction set their balance to zero! A DBMS must guarantee that transactions are executed fully and independently of other transactions. An essential property of a DBMS is that a transaction should execute atomically, or as if it is the only transaction running. Also, transactions will either complete fully, or will be aborted and the database returned to it's initial state. This ensures that the database remains consistent.	Remember	CACS005.05
9	Discuss the disadvantages of file processing system, and explain how these disadvantages are avoided in DBMS? Disadvantage of file processing system: The file processing system has the following major disadvantages: Data redundancy and inconsistency. Integrity Problems. Security Problems Difficulty in accessing data. Data isolation. a) Data redundancy and inconsistency: Data redundancy means duplication of data and inconsistency means that the duplicated values are different. b) Integrity problems: Data integrity means that the data values in the database should be accurate in the sense that the value must satisfy some rules. c) Security Problem: Data security means prevention of data accession by unauthorized users. d) Difficulty in accessing data: Difficulty in accessing data arises whenever there is no application program for a specific task. e) Data isolation: This problem arises due to the scattering of data in various files with various formats. Due to the above disadvantages of the earlier data processing system, the necessity for an effective data processing system arises.	Understand	CACS005.02

	Only at that time the concept of DBMS emerges for the rescue of a large number of organizations. Precautions to avoid the disadvantage of dbms: Provide data definition facilities Provide security of data Provide facilities for database recovery Provide concurrency control mechanism		
10	Design a relational database for a university registrar's office the office maintain data about each class, including the instructor, the number of sudents enrolled,and time and place of the class meatings. For each student - class pair, a grade is recorded.	Remember	CACS005.05

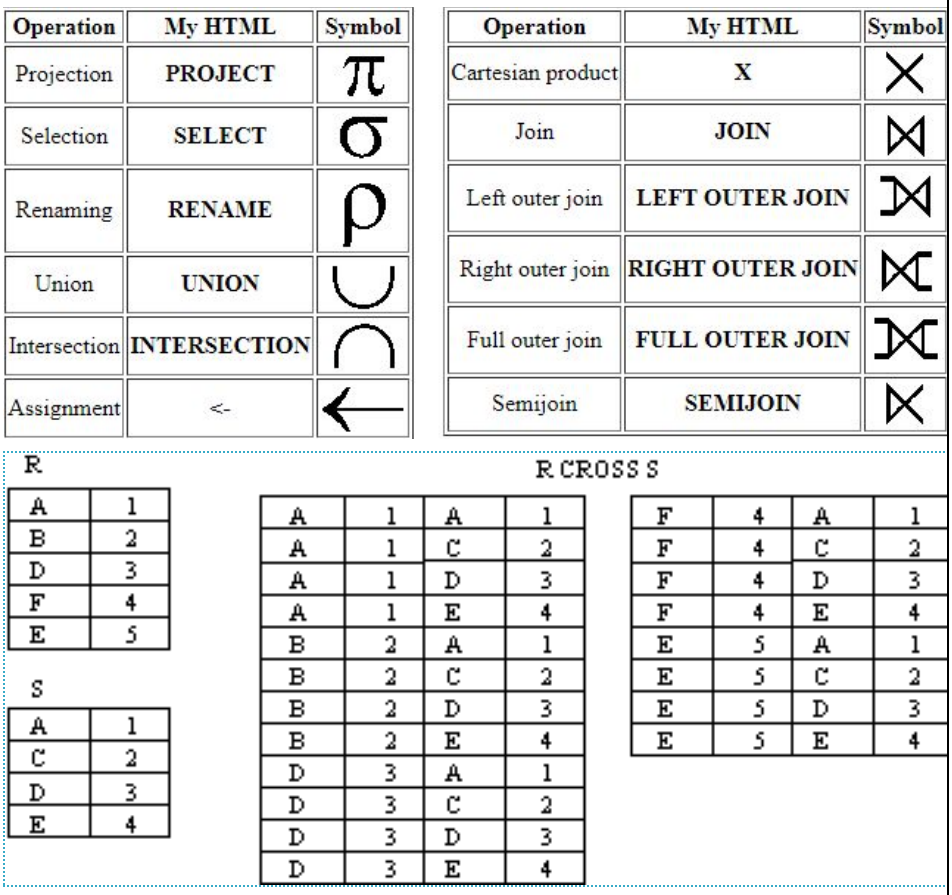
	UNIT – II **Relational Approach**		
	PART – A (Short Answer Questions)		
1. ans	Define relational database query. A relational database is a set of formally described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables. The standard user and application programming interface (API) of a relational database is the Structured Query Language (SQL).	Remember	CACS005.08
2. ans	State the purpose of SELECT operation in Relational algebra.  2) Select operator is used to access data from a relation satisfying certain condition Notation : $\sigma_p(r)$, where σ is Select Operator p is Predicate r is relation schema  It is defined as mathematically $\{t \mid t \in r \text{ and } P(t)\}$ where p is a Predicate in Propositional Calculus consists of Logical operators ($\wedge, \vee, \neg$) & relational operators ($=, >, <, >=, <=, <>$) Ex: Display the details of student with id 110 SELECT * FROM student WHERE sid = 110; => $\sigma_{sid=110}$(student)	Understand	CACS005.09
3 ans.	State the purpose of PROJECT operation in Relational algebra.  2) Project : It is used to ~~select~~ choose attributes of relation. $\Pi_{A1, A2, \ldots, An}(r)$ By default, the duplicated values are removed since the relation represents a set.	Understand	CACS005.09

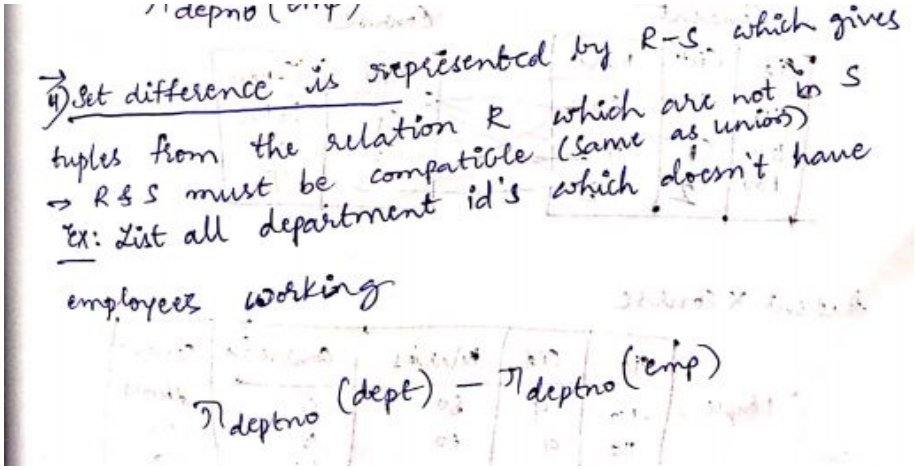
	→ To display empid and salary, whose salary is greater than 5000 $\Pi_{eid,sal}(\sigma_{sal>5000}^{(employee)})$		
4. a	Define a relational calculus. **Relational calculus** is a non procedural query language. It uses mathematical predicate **calculus**instead of algebra. It provides the description about the query to get the result where as **relational**algebra gives the method to get the result.	Understand	CACS005.10
5. a	Discuss the use of rename operation. 6) Rename: Rename allows us to give a new name to a relation and represented by 'ρ'. Ex: $\rho_x(E)$ then the relation E can also be reffered as X → Display the details of employee id, employee name and Job as Personal details $\rho_{p\_emp}(\Pi_{eid,ename,Job}(emp))$ $\sigma_{Job='Clerk' \vee Job='Manager'}^{(P\_emp)}$	Remember	CACS005.09
6. a	Illustrate division operation.	Remember	CACS005.09

	⑪ Division (r ÷ s) operator This operator is suitable for the queries including a phrase 'for all'. Let r, s be the relations on relational schema's R & S respectively. where $R(A_1, A_2, \ldots, A_n)$ with $R(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ $S(B_1, B_2, \ldots, B_m)$ Then, the relation schema of r ÷ s is $R-S = (A_1, A_2, \ldots, A_n)$ and the relation $r \div s = \{t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s \text{ and } tu \in R\}$ → Let $q = r \div s$ then 'q' is the largest relation satisfying the constraint $q \times s \subseteq r$ This operation can also be expressed by using basic relational operators. $r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ This operator re-orders the attributes of R This gives tuples t in $\Pi_{R-S}(r)$ such that for some $u \in s$, $tu \notin r$ Query) Find the customers who have an account in all branches. $\Pi_{c\text{-name}, B\text{-name}}(\text{Account} \bowtie \text{depositor}) \div \Pi_{B\text{-name}}(\text{Branch})$		
7. a	Discuss about expressive power of algebra and calculus.	Understand	CACS005.10

			
8 a.	Define a tuple relational calculus. 	Remember	CACS005.10
9. a	Illustrate union operation and intersection operation.	Understand	CACS005.09

	3) Union : $(R \cup S)$ This operator selects tuples from the both/two relations satisfying the following conditions ① both relations must have same arity same arity = same no. of attributes ② The domains of respective attributes must be compatible $R \cup S = \{t / t \in R \vee t \in S\}$ Ex: display the details of employee id who are drawing 50,000 salary & employees who belongs to department 20 Select empid from employee WHERE salary = 50,000 AND deptno = 20 relational Algebra: $\Pi_{eid}(\sigma_{sal=50000 \wedge deptno=20}(emp))$ ① Set intersection $(R \cap S)$ which results common tuples from both R & S satisfying compatibility condition. $R \cap S = R - (R - S)$ $R \cap S = \{t / t \in R \wedge t \in S\}$		
10 a.	Illustrate cross-product operation. **CARTESIAN PRODUCT** The Cartesian Product is also an operator which works on two sets. It is sometimes called the CROSS PRODUCT or CROSS JOIN. It combines the tuples of one relation with all the tuples of the other relation. **CARTESIAN PRODUCT example**	Remember	CACS005.09

			
11. a	List set operators in relational algebra. • Set operators.	Understand	CACS005.09
12. a	List aggregate functions used in Relational Algebra. There are five aggregate functions that are included with most relational database systems. These operations are Sum Count Average Maximum and Minimum.	Remember	CACS005.09
13. a	List out types of joins. # Types of JOIN Following are the types of JOIN that we can use in SQL: • Inner • Outer • Left	Remember	CACS005.09

	• Right		
14. a	Illustrate set difference operation.  	Understand	CACS005.09
15. a	Define a domain relational calculus. A **tuple relational calculus** is a non procedural query language which specifies to select the **tuples** in a relation. It can select the **tuples**with range of values or **tuples** for certain attribute values etc. The resulting relation can have one or more **tuples**.	Understand	CACS005.10
	PART – B (Long Answer Questions)		
1 a	Illustrate different set operations in Relational algebra with an example. The fundamental operations of relational algebra are as follows – • Select • Project • Union • Set different • Cartesian product • Rename We will discuss all these operations in the following sections. # Select Operation (σ) It selects tuples that satisfy the given predicate from a relation.	Understand	CACS005.09

Notation – $\sigma_p(r)$

Where σ stands for selection predicate and r stands for relation. *p* is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like – =, ≠, ≥, < , >, ≤.

For example –

$\sigma_{subject = "database"}(Books)$

Output – Selects tuples from books where subject is 'database'.

$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output – Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output – Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation – $\prod_{A1, A2, An}(r)$

Where A_1, A_2 , A_n are attribute names of relation r.

Duplicate rows are automatically eliminated, as relation is a set.

For example –

$\prod_{subject, author}(Books)$

Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as –

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$$

Notation – r U s

Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold –

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$$\prod_{author} (Books) \cup \prod_{author} (Articles)$$

Output – Projects the names of the authors who have either written a book or an article or both.

Set Difference (–)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation – r – s

Finds all the tuples that are present in r but not in s.

$$\prod_{author} (Books) - \prod_{author} (Articles)$$

Output – Provides the name of authors who have written books but not articles.

Cartesian Product (X)

Combines information of two different relations into one.

Notation – r X s

Where r and s are relations and their output will be defined as –

r X s = { q t | q ∈ r and t ∈ s}

$\sigma_{author = 'tutorialspoint'}(Books\ X\ Articles)$

Output – Yields a relation, which shows all the books and articles written by tutorialspoint.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho *ρ*.

Notation – $\rho_x(E)$

Where the result of expression E is saved with name of x.

Additional operations are –

- Set intersection
- Assignment
- Natural join

2 a	Define Join. Explain different types of joins in relational algebra.	Remember	CACS005.09

JOINS : to get details from two or more tables satisfying certain conditions

Self Join : a join operation is used with a table itself.

→ joining a table with itself is Self Join

eid	ename	Mgrid	sal
110	John	120	5000
115	James	120	6000
120	Mary	130	7000
125	Raja	130	4000
130	Akbar	130	7500

→ List all eid, ename along with their manager id & manager name

```
SELECT emp.eid As "employee ID",
emp.ename AS "employee name"
emp.mgrid As "manager Id"
manage.ename As "manager name"
FROM emp, emp manager
where emp.mgrid = Manager.eid;
```

Inner Join : Joining two tables satisfying the condition in the WHERE clause (The condition may be equality condition known as "Equi-Join" or other condition known as "Theta Join"

θ - Join

→ Inner Join always lists the tuples satisfying the Join condition only.

Ex: List all emp names along with the departments in which they are working along with the details as deptno, deptname

emp table

eid	e-name	Mgrid	Sal
110	John	120	5000
115	James	120	6000
120	Mary	130	7000
125	Raja	130	4000
130	Akbar	130	75000

dept table

deptno	deptno	Accounts
10	10	Accounts
20	20	HR
30	30	Products
10	40	Logistics
20	50	Marketing

```
SELECT empname, dept.deptno, dname
FROM emp, dept
WHERE emp.deptno = dept.deptno;
```

emp

empid	ename	deptno
110	X	10
120	Y	
130	Z	20
140	A	20

dept

deptno	d-name	loc
10	CSE	I^st
20	ECE	II^nd
30	EEE	III^rd

Equi Join :

```
SELECT emp.eid, emp.ename, emp.deptno,
                                dept.dname
  FROM emp, dept
  WHERE emp.deptno = dept.deptno
```

Outer Join : It is used to retrieve unmatched data in relevant columns also to be listed.

three types : ① Left-outer Join : It displays left table entries for which there is no match also

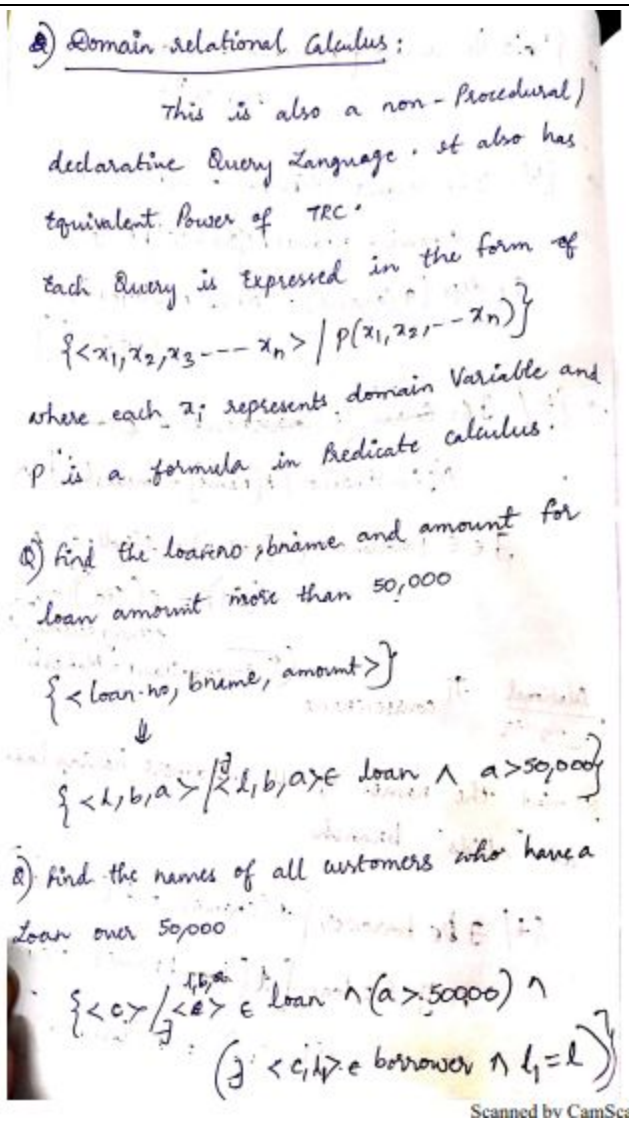
② right-outer Join : displays right table column values for which there is no match in the left table

③ Full-outer Join : displays the column values which doesn't have any match from the two tables along with matching attributes.

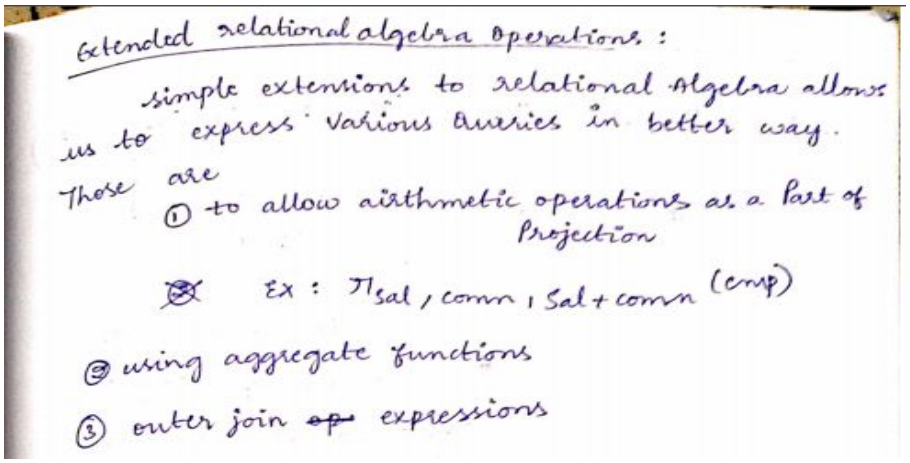
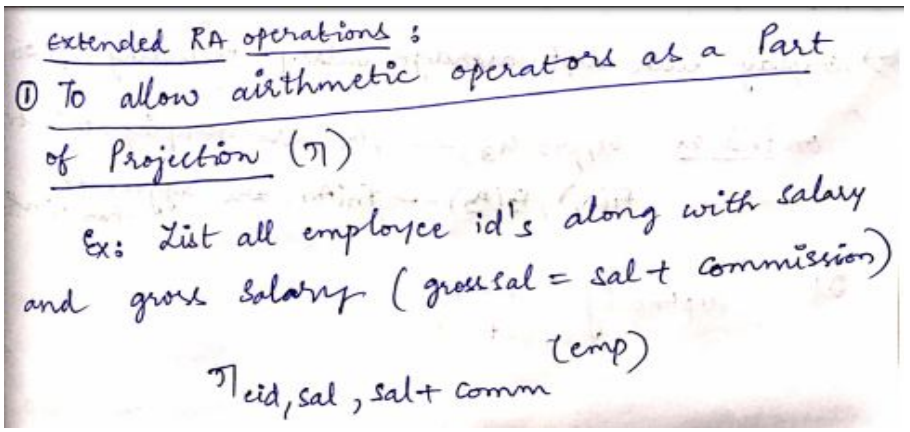
Ex for Left outer Join :

```
SELECT emp.eid, emp.ename, dept.deptno, dept.dname
  FROM emp, dept
  WHERE emp.deptno = dept.deptno(+)
```

	Output: 110 X 10 CSE 120 Y 130 Z 20 ECE 140 A 20 ECE Right - outer Join : SELECT emp.eid, emp.ename, dept.deptno, dept.dname FROM emp, dept WHERE emp.deptno(+) = dept.deptno output : 110 X 10 CSE 130 Z 20 ECE 140 A 20 ECE 30 EEE Full - outer Join : WHERE emp.deptno (+) = dept.deptno(+) output : 110 X 10 CSE 120 Y 130 Z 20 ECE 140 A 20 ECE 30 EEE		
3 a	Discuss about Domain Relational calculus in detail.	Remember	CACS005.10

	 a) Domain relational Calculus: This is also a non-Procedural / declarative Query Language. It also has equivalent Power of TRC. Each Query is expressed in the form of $\{<x_1, x_2, x_3 \dots x_n> \mid P(x_1, x_2, \dots x_n)\}$ where each x_i represents domain Variable and P is a formula in Predicate calculus. Q) find the loanno, bname and amount for loan amount more than 50,000 $\{<\text{loan-no, bname, amount}>\}$ $\{<l, b, a> \mid <l, b, a> \in \text{loan} \wedge a > 50{,}000\}$ Q) find the names of all customers who have a loan over 50,000 $\{<c> \mid \exists l, b, a (<l, b, a> \in \text{loan} \wedge (a > 50000) \wedge (\exists <c, l> \in \text{borrower} \wedge l_1 = l))\}$ Scanned by CamSca		
4 a	Discuss the difference between Relational Algebra and Relational Calculus.	Remember	CACS005.10

BASIS FOR COMPARISON	RELATIONAL ALGEBRA	RELATIONAL CALCULUS
Basic	Relational Algebra is a Procedural language.	Relational Claculus is Declarative language.
States	Relational Algebra states how to obtain the result.	Relational Calculus states what result we have to obtain.
Order	Relational Algebra describes the order in which operations have to be performed.	Relational Calculus does not specify the order of operations.
Domain	Relational Algebra is not domain dependent.	Relation Claculus can be domain dependent.
Related	It is close to a programming language.	It is close to the natural language.

5 a	Illustrate Extended relational operations with examples.  	Understand	CACS005.09

③ using aggregate Functions :

applied to collection of Values

Sum, min, max, Avg, Count

→ This function is represented by $\mathcal{G}$

Ex: No. of employees in your table

$\mathcal{G}_{count}(emp)$

Print min and avg salaries of emp

$\mathcal{G}_{min(sal),Avg(sal)}(emp)$

→ we can use aggregate functions along grouping.

General Syntax of aggregate operator is

$G_1, G_2, ---, G_n\ \mathcal{G}_{F_1(A_1),F_2($

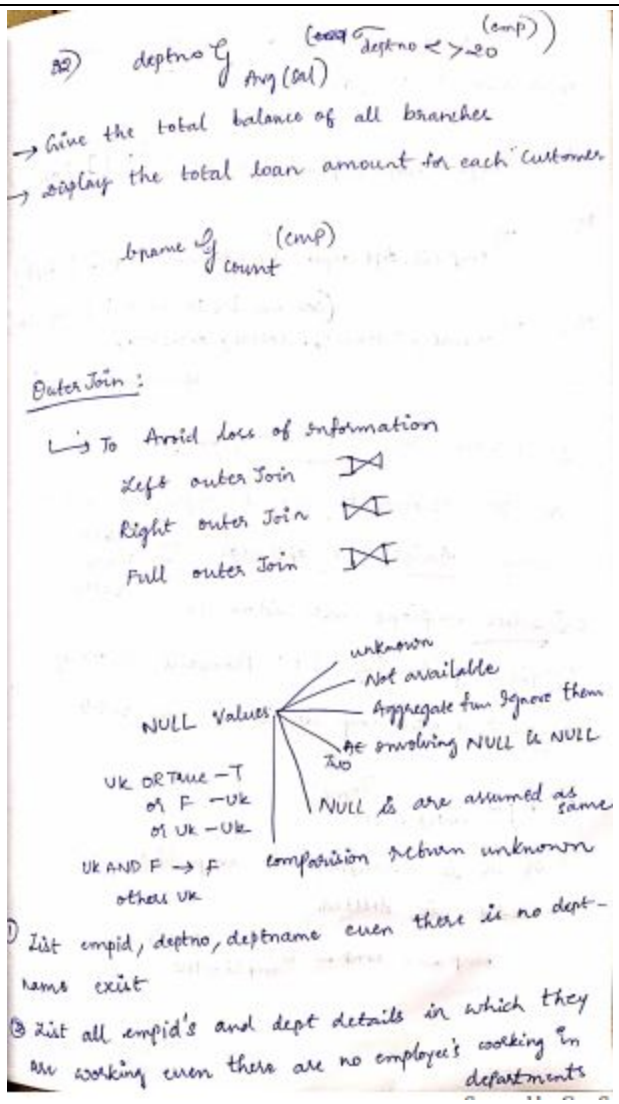
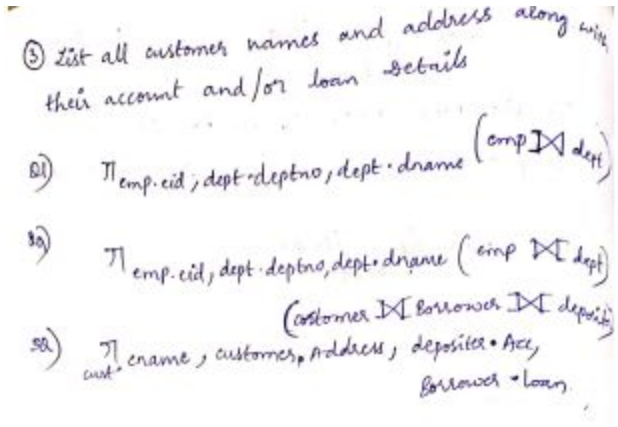
→ List no. of employees in each department

→ display each dept average salary excludin

In this Ex: $G_1, G_2, G_3, --, G_n$ are grouping

$F_1(A_1), F_2(A_2) --- F_n(A_n)$ are aggre

Q1) deptno $\mathcal{G}_{count}(emp)$

	 		
6 a	Discuss about Complex integrity constraints in SQL.	Remember	CACS005.06

Constraints

Integrity Constraints: These are the rules enforced on data of a Table to ensure accuracy and reliability of data in the database. Whenever modifications are done on data of a Table it will check the constraints are satisfied before and after modifications.

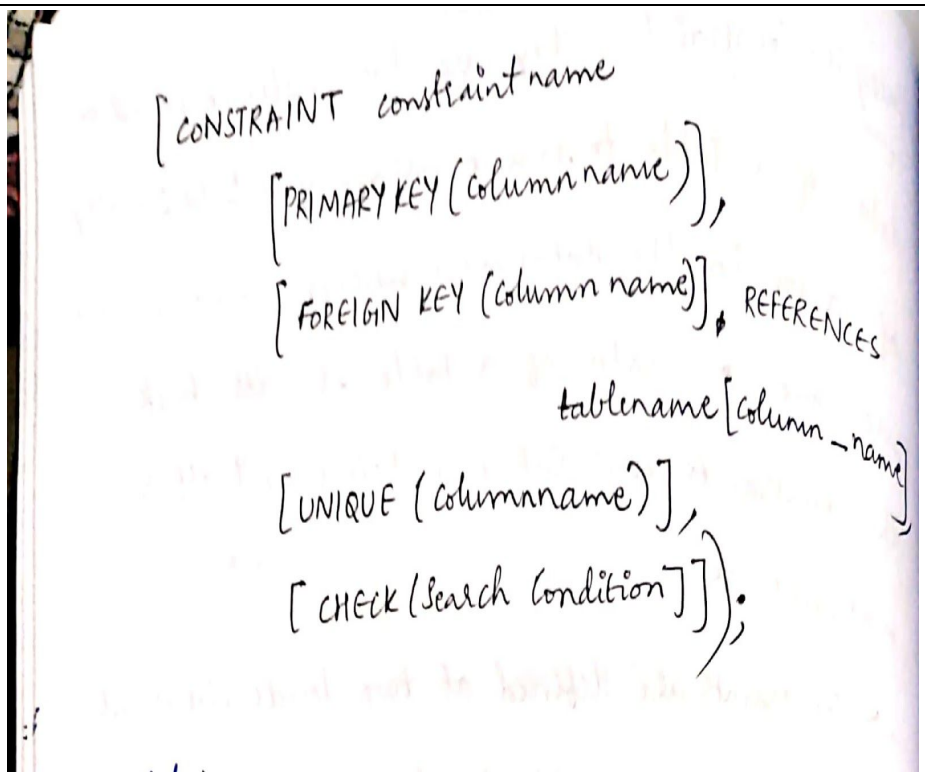
→ Constraints are defined at two levels either at

Column level or **Table level**

→ Column level constraints are defined along with column defination without specifying the colname. (except for check constraint).

→ Table level constraints are specified at the end of the table defination. These may refer to one or more columns. These constraints should specify column names also

```
CREATE TABLE Tablename
(columnname datatype [default value]
   [CONSTRAINT constraint name
      { PRIMARY KEY
        UNIQUE
        CHECK (search condition)
        NOTNULL
```



7 a	Discuss structure of query in TRC and DRC with example.	Understand	CACS005.05

Tuple Relational Calculus (TRC)

Filtering variable ranges over tuples

Notation – {T | Condition}

- Returns all tuples T that satisfies a condition.

For example –

```
{ T.name |  Author(T) AND T.article = 'database' }
```

Output – Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential (∃) and Universal Quantifiers (∀).

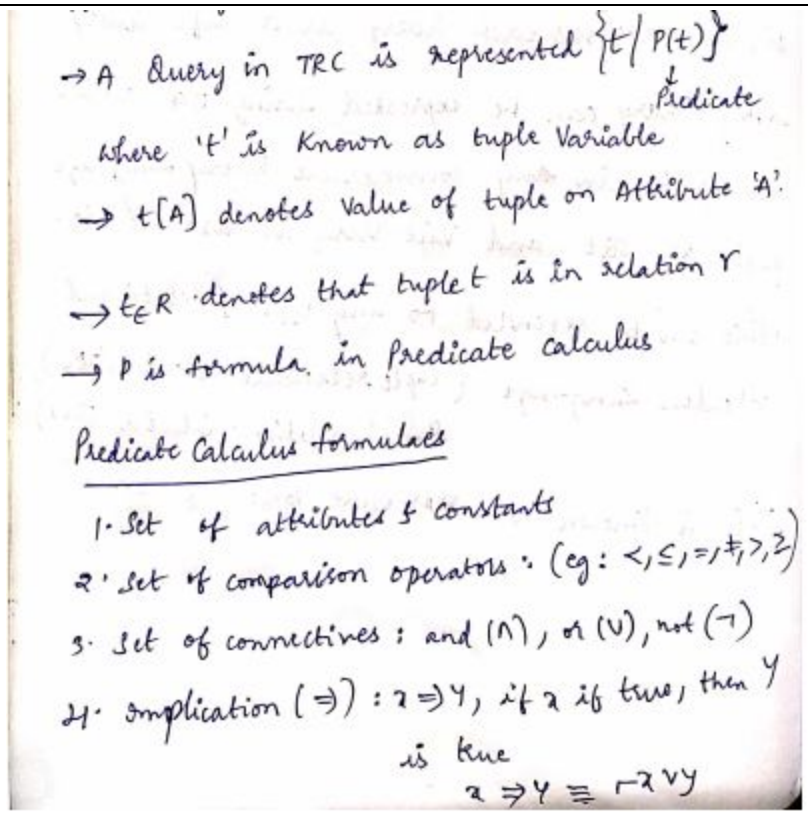
For example –

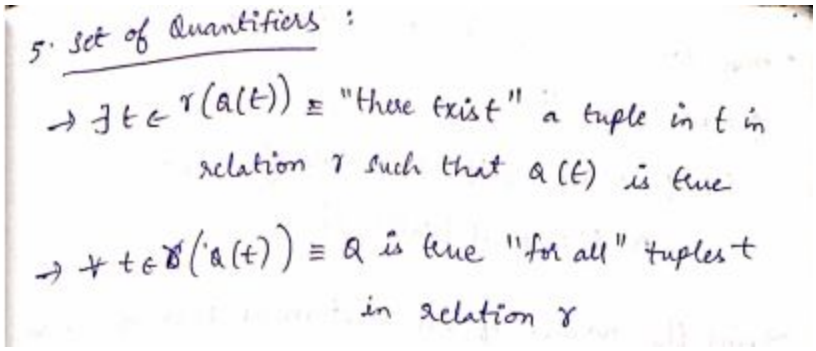
```
{ R| ∃T   ∈ Authors(T.article='database' AND R.name=T.name)}
```

Output – The above query will yield the same result as the previous one.

Domain Relational Calculus (DRC)

	In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above). Notation – $\{ a_1, a_2, a_3, ..., a_n \mid P (a_1, a_2, a_3, ... ,a_n)\}$ Where a1, a2 are attributes and P stands for formulae built by inner attributes. For example – `{< article, page, subject > | ∈ TutorialsPoint ∧ subject = 'database'}` Output – Yields Article, Page, and Subject from the relation TutorialsPoint, where subject is database. Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators. The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.		
8 a	a.Define a query in Tuple relational Calculus. ans.	Remember	CACS005.10





b.Write a query in TRC to find the names of sailors who have reserved both a red and green boat?

ans.

ρ(T empboats2,(σcolor='red'Boats) ∩ (σcolor='green'Boats))
πsname(T empboats2 ⋪⋫ Reserves ⋪⋫ Sailors)

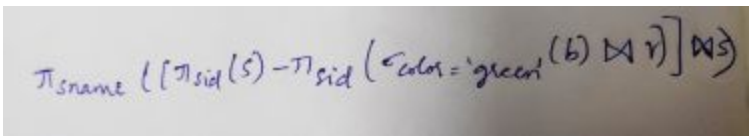
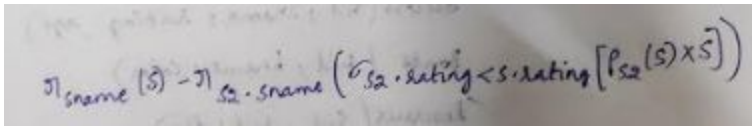
c.Write a query in TRC to find the names of sailors who have reserved all boats?

ans.

	8c) Names of Sailors who have reserved all boats $\{P \mid \exists S \in Sailors \forall B \in Boats$ $(\exists R \in Reserves (S.sid = R.sid \wedge R.bid = B.bid$ $\wedge P.sname = S.sname))\}$ Sailors (sid, Sname, rating, age) Boats (bid, bname, color) Reserves (sid, bid, day)		
9	a. Define query in Domain Relational Calculus. ans. 9a) Domain relational Calculus: This is also a non-Procedural/ declarative Query Language. It also has equivalent Power of TRC. Each Query is expressed in the form of $\{<x_1, x_2, x_3 --- x_n> \mid P(x_1, x_2, --- x_n)\}$ where each x_i represents domain Variable and P is a formula in Predicate calculus. b. Write a query in DRC to find the names of sailors who have reserved a red boat? ans. 9b) $\{<N> \mid \exists I, R, A (<I, N, R, A> \in Sailors \wedge$ $\exists <I, Bi, D> \in Reserves \wedge \exists <Bi, BN, 'red'> \in$ $Boats)\}$ c.Write a query in DRC,to find the names of sailors who have not reserved a red boat? ans.	Remember	CACS005.10

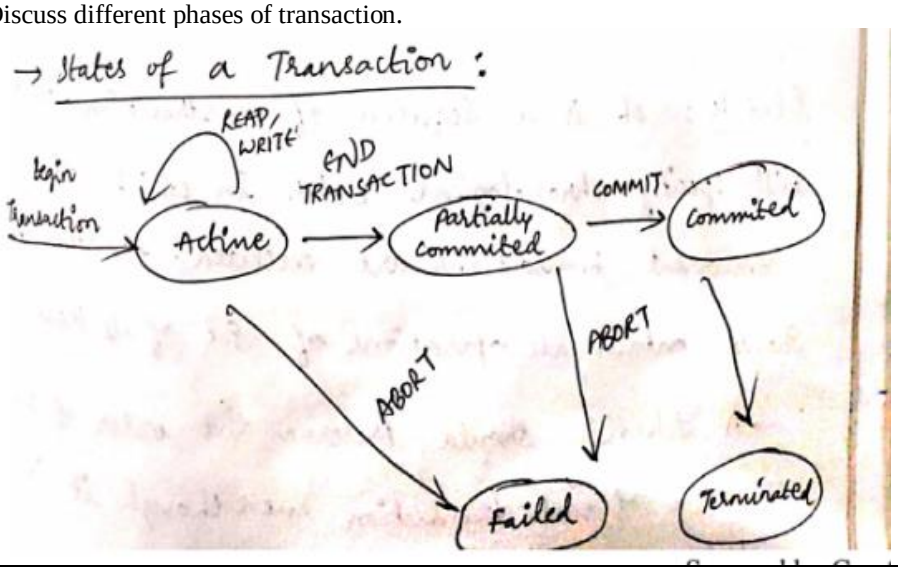
	9c) $\{\langle N\rangle \mid \neg(\exists I,R,A)(\langle I,N,R,A\rangle \in Sailors \wedge \exists \langle I,Bi,D\rangle \in Reserves \wedge \exists \langle Bi,BN,'red'\rangle \in Boats)\}$		
10	a. Explain Relational calculus. ans. **Relational Algebra** Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations. The fundamental operations of relational algebra are as follows – • Select • Project • Union • Set different • Cartesian product • Rename b.Write a TRC query to find the names of sailors who have reserved boat 103? ans. 10) b) $\{P \mid \exists S \in Sailors \exists R \in Reserves (R.sid = S.sid \wedge R.bid = 103 \wedge P.sname = S.sname)\}$ c.Write a DRC query to find the names of sailors who have reserved boat 103? ans.	Remember	CACS005.10

	10c) $\{\langle N \rangle \mid \exists I,T,A\ (I,N,T,A \in Sailors \wedge \exists Ir,Br,D(\langle Ir,Br,D \rangle \in Reserves \wedge Ir = I \wedge Br = 103))\}$		
11	Let R=(ABC) and S=(DEF) let r(R) and s(S) both relations on schema R and S. Give an expression in the Tuple relational calculus that is equivalent to each of the following. i) $\sigma B=19(r)$ ans. $\{t \mid t \in r \wedge t[B] = 19\}$ ii) $\prod A,F,(\sigma C=D(r \times s))$ ans. $\{t \mid \exists\ p \in r\ \exists\ q \in s\ (t[A] = p[A] \wedge t[F] = q[F] \wedge p[C] = q[D]\}$ iii) $r \cap s$ ans. $\{t \mid t \in r \wedge t \in s\}$	Remember	CACS005.10
12	Consider the following schema instructor (ID, name, dept_name), teaches (ID, course_id, sec_id, semester, year), section (course_id, sec_id, semester, year), student (ID, name, dept_name), takes (ID, course_id, sec_id, semester, year, grade) Write the following query in RA,TRC and DRC a) Find the names of the instructors not teaching any course.	Remember	CACS005.10
	PART – C (Problem Solving and Critical Thinking Questions)		
1	Given the Students relation as shown below *StudentID* | *StudentName* | *StudentEmail* | *StudentAge* | *CPI* 2345 | Shankar | shankar@math | X | 9.4 1287 | Swati | swati@ee | 19 | 9.5 7853 | Shankar | shankar@cse | 19 | 9.4 9876 | Swati | swati@mech | 18 | 9.3 8765 | Ganesh | ganesh@civil | 19 | 8.7 For (Student Name, Student Age) to be the key for this instance, analyze and find the value of X not be equal to student age.	Remember	CACS005.07
ans.	For (Student Name, Student Age) to be a key for given instance of STUDENTS relation, the pair value should not get repeated in any two tuples p and q (uniqueness is forced by the definition of key) Tuple | Student Name | Student Age P | Shankar | X → should not be 19		

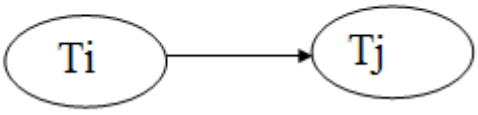
2 ans.	Given the relations *employee(name,salary,deptno)* *department (deptno, deptname, address)* Solve which query cannot be expressed using the basic relational algebra operations. **The sum of all employees' salaries** **Explanation:** The six basic operators of relational algebra are the selection(σ), the projection(π), the Cartesian product (x) (also called the cross product or cross join), the set union (U), the set difference (-), and the rename (p). These six operators are fundamental in the sense that none of them can be omitted without losing expressive power. Many other operators have been defined in terms of these six. Among the most important are set intersection, division, and the natural join, but aggregation is not possible with these basic relational algebra operations. So, we cannot run sum of all employees' salaries with the six operations.	Remember	CACS005.09
3	Write Query in relational algebra to find second highest salary of Employee from Employee relation.	Understand	CACS005.09
4	Consider the following schema given. The primary keys are underlined. Sailors(sailor-id, sailor-name, sailor-rating, sailor-age) Boats(boat-id, boat-name, boat-color) Reserves(sailor-id, boat-id, day) Write queries in Relational Algebra. i. Find the names of sailors who have reserved boat number 120 ans. πsname(σbid = 120 (Reserves ∞ Sailors)) ii. Find the names of sailors who have reserved a green boat ans. πsname((σcolor='green' Boats) ◁▷ Reserves ◁▷ Sailors) iii. Find the names of sailors who have not reserved a green boat ans.  iv. Find the names of sailors with the highest rating ans. 	Remember	CACS005.09

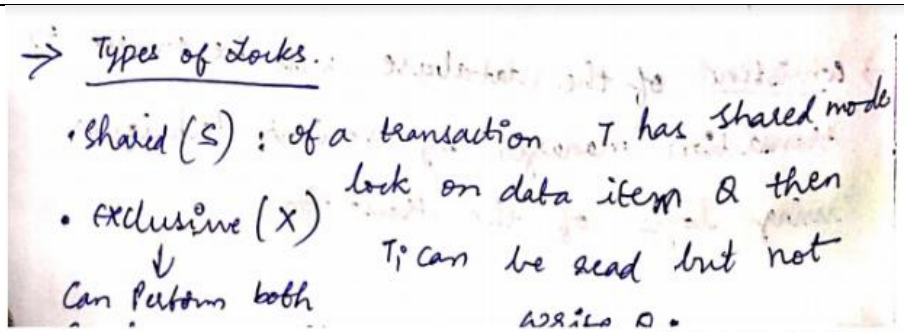
5	Consider the following database. Employee (employee-name, street, city) Works (employee-name, company-name, salary)	Understa nd	CACS005. 10

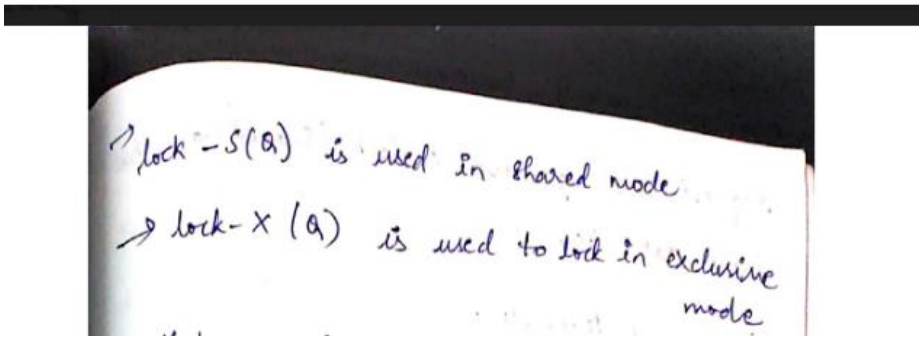
	Company (company-name, city) Manager (employee-name, manager-name) Give an expression in the relational algebra, the tuple relational calculus, and the domain relational calculus, for the following query. Find the names of all employees who work for estate bank.		
ans.	relational algebra - Πperson-name (σcompany-name = "estate" (works)) TRC -. {p| ∃ w ∈ works(p.person_name = w.person_name ∧ w.company_name = éstate'')} DRC - {<pn>| ∃ v(<pn, 'estate' , v> ∈ works)}		
6	Write the RA expression for the following Queries. Sailor Schema (sailor id, Sailorname, Rating.Age) Reserves (Sailor id, Boat id, Day) Boat Schema (Boat id, Boatname.color) i. Find the names of sailors who have reserved boat name 103; ans. πsname(σbid = 103 (Reserves ∞ Sailors)) ii. Find the sailor id of sailors who have reserved a red boat; ans .πsid((σcolor=′red′ Boats) ⧐ Reserves ⧐ Sailors) iii. Find the colors of boats reserved by the sailor rubber. ans. iv. Find the names of sailors who have reserved a red boat. ans. πsname((σcolor=′red′ Boats) ⧐ Reserves ⧐ Sailors)	Understan d	CACS005. 09
7	For the following relational database, give the expressions in RA. student(stuno, stuname, major,level,age) Class(Classname, meets at, Room, fid) Faculty(fid,fname,deptid) i. Find the names of all uniors (level = JR) Who are enrolled in a class taught by I.Teach. ii. Find the age of the oldest student who is either a history major or is enrolled in a course taught by I.Tech? iii. Find the names of all classes that either meet in room R128 or have five or more students enrolled? iv. Find the names of faculty members whom the combined enrollment of the course that they is less than 5? v. Print the level and the average age of students for that level, for each level?	Remembe r	CACS005. 09

	UNIT – IV **Transaction Management**		
	PART – A (Short Answer Questions)		
1 ans	Define a Transaction. List the properties of transaction. A transaction is a unit of Program execution that access and possibly updates various data items. Transaction Properties : To Preserve Integrity of data in database System a transaction has to Satisfy the following Properties always. These Properties are known as ACID Properties Scanned by CamS Stands for Atomicity, consistency, Isolation, Durability	Remember	CACS005.18
2	Discuss different phases of transaction. → States of a Transaction : 	Remember	CACS005.18

	① Active: When a transaction Processing either read or write statements ② Partially commited: When a transaction completed but not yet commited completely ③ commited: Succesfully completed transaction ④ terminated: Removing the transaction from the queue. ⑤ failed: Because of failures (software/hardware) transaction will stop abruptly		
3	Discuss recoverable schedules. Recovarable Schedules: To maintain the consistency of the database in a Schedule of a Transaction T_j reads a data item which is written by T_i then the commit operation of T_j must be appear before commit operation of T_i Scanned by CamSc Then the Schedule is called recovarable Schedule.	Remember	CACS005.19
4 ans	Discuss cascade less schedules A **cascadeless schedule** is one where, for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti , the commit operation of Ti appears before the read operation of Tj .	Understand	CACS005.19
5 ans	Define Two Phase Commit protocol. The **two phase commit protocol** is a distributed algorithm which lets all sites in a distributed system agree to **commit** a **transaction**. The **protocol**results in either all nodes **committing** the**transaction** or aborting, even in the case of site failures and message losses	Remember	CACS005.19
6 ans	Demonstrate the implementation of Isolation.	Remember	CACS005.18

	→ Isolation : All the multiple transactions are executing concurrently. Each transaction is unaware of other concurrently executing transactions i.e intermediate results are hidden from other transactions when they are executing concurrently.		
7 ans	Discuss the Procedure to test Serializability. **Testing of Serializability** Serialization Graph is used to test the Serializability of a schedule. Assume a schedule S. For S, we construct a graph known as precedence graph. This graph has a pair G = (V, E), where V consists a set of vertices, and E consists a set of edges. The set of vertices is used to contain all the transactions participating in the schedule. The set of edges is used to contain all edges Ti ->Tj for which one of the three conditions holds: 1. Create a node Ti → Tj if Ti executes write (Q) before Tj executes read (Q). 2. Create a node Ti → Tj if Ti executes read (Q) before Tj executes write (Q). 3. Create a node Ti → Tj if Ti executes write (Q) before Tj executes write (Q). **Precedence graph for Schedule S**  • If a precedence graph contains a single edge Ti → Tj, then all the instructions of Ti are executed before the first instruction of Tj is executed. • If a precedence graph for schedule S contains a cycle, then S is non-serializable. If the precedence graph has no cycle, then S is known as serializable.	Understand	CACS005.19
8 ans	List different types of locks and write about compatability among them.	Remember	CACS005.19





Compatibility of Locks

Suppose that there are A and B two different locking modes. If a transaction T1 requests a lock of mode on item Q on which transaction T2 currently hold a lock of mode B. If transaction can be granted lock, in spite of the presence of the mode B lock, then we say mode A is compatible with mode B. Such a function is shown in one matrix as shown below:

	S	X
S	true	false
X	false	false

Compatibility Graph

9 ans	Discuss about Failure Classification.	Remember	CACS005.19

Failure Classification

To find that where the problem has occurred, we generalize a failure into the following categories:

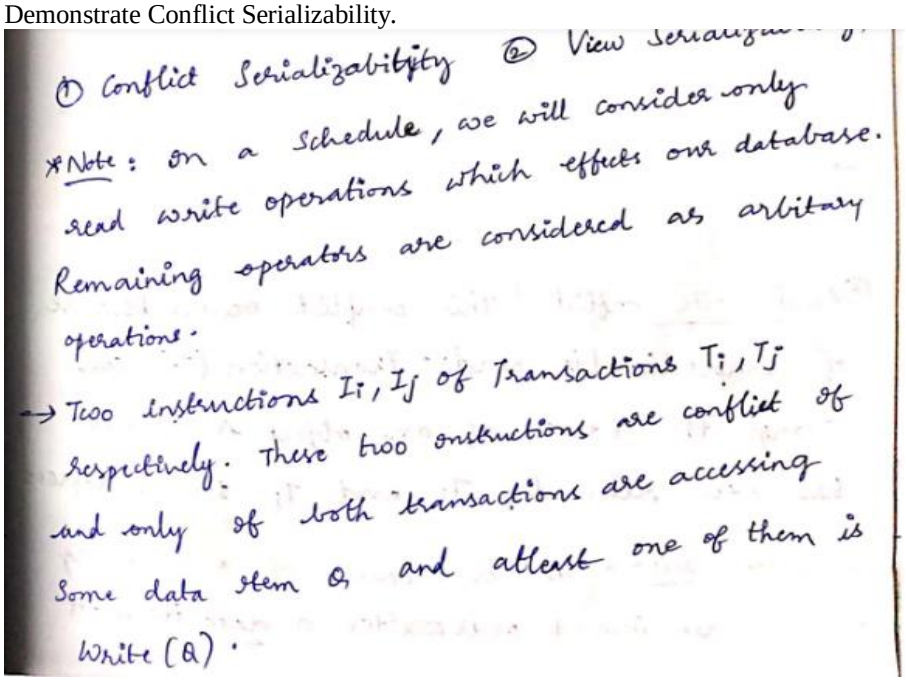
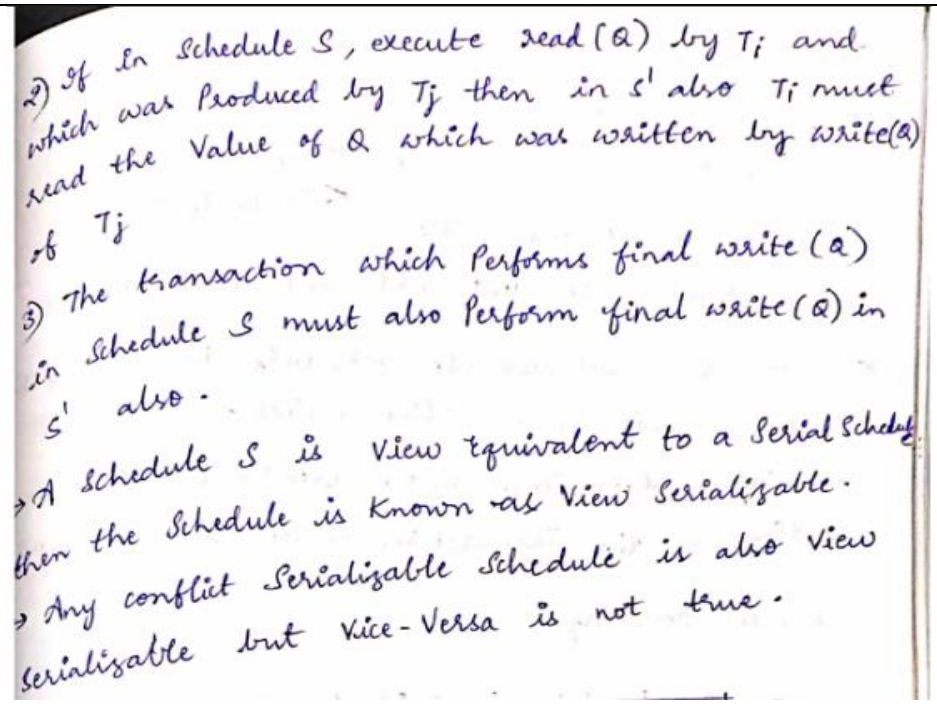
1. Transaction failure
2. System crash
3. Disk failure
4. 1. Transaction failure

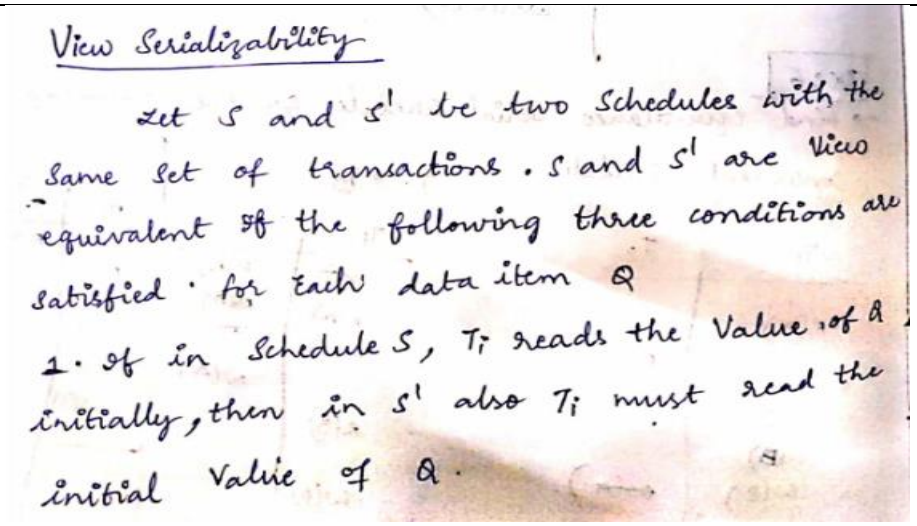
The transaction failure occurs when it fails to execute or when it reaches a point from where it can't go any further. If a few transaction or process is hurt, then this is called as transaction failure.

Reasons for a transaction failure could be -

- **Logical errors:** If a transaction cannot complete due

	to some code error or an internal error condition, then the logical error occurs. ○ **Syntax error:** It occurs where the DBMS itself terminates an active transaction because the database system is not able to execute it. **For example,** The system aborts an active transaction, in case of deadlock or resource unavailability. 5. 2. System Crash ○ System failure can occur due to power failure or other hardware or software failure. **Example:** Operating system error. **Fail-stop assumption:** In the system crash, non-volatile storage is assumed not to be corrupted. 6. 3. Disk Failure ○ It occurs where hard-disk drives or storage drives used to fail frequently. It was a common problem in the early days of technology evolution. ○ Disk failure occurs due to the formation of bad sectors, disk head crash, and unreachability to the disk or any other failure, which destroy all or part of disk storage.		
10 ans	Define a checkpoint. **Checkpoint** is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. **Checkpoint** declares a point before which the **DBMS** was in consistent state, and all the transactions were committed.	Understand	CACS005.20
11	Discuss the failures that can occur with loss of Non-volatile storage. The basic scheme is to **dump** the entire content of the database to stable storage periodically—say, once per day. For example, we may dump the database to one or more magnetic tapes. If a failure occurs that results in the loss of physical database blocks, the system uses the most recent dump in restoring the database to a previous consistent state. Once this restoration has been accomplished, the system uses the log to bring the database system to the most recent consistent state. More precisely, no transaction may be active during the dump procedure, and a procedure similar to checkpointing must take place: **1.** Output all log records currently residing in main memory onto stable storage.	Remember	CACS005.20

	2. Output all buffer blocks onto the disk. **3.** Copy the contents of the database to stable storage. **4.** Output a log record <dump> onto the stable storage		
12	Demonstrate Conflict Serializability.  ① Conflict Serializability ② View Serializa... *Note: on a Schedule, we will consider only read write operations which effects our database. Remaining operators are considered as arbitary operations. → Two instructions I_i, I_j of Transactions T_i, T_j respectively. These two instructions are conflict if and only if both transactions are accessing Some data item Q and atleast one of them is Write(Q).	Understand	CACS005.19
13	 2) If in Schedule S, execute read(Q) by T_i and which was Produced by T_j then in S' also T_i must read the Value of Q which was written by write(Q) of T_j 3) The transaction which Performs final write(Q) in Schedule S must also Perform final write(Q) in S' also. → A Schedule S is View tquivalent to a Serial Schedule then the Schedule is Known as View Serializable. → Any conflict Serializable Schedule is also View Serializable but Vice-Versa is not true. Discuss View Serializability.	Remember	CACS005.19


View Serializability

Let S and S' be two Schedules with the same set of transactions. S and S' are view equivalent if the following three conditions are satisfied. for each data item Q

1. if in Schedule S, T_i reads the Value of Q initially, then in S' also T_i must read the initial Value of Q.

PART – B (Long Answer Questions)

1 ans	Explain ACID properties and Illustrate them through examples.	Remember	CACS005.18

ACID Properties in DBMS

A transaction is a single logical unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.

In order to maintain consistency in a database, before and after transaction, certain properties are followed. These are called **ACID** properties.

Atomicity

By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves following two operations.

—**Abort**: If a transaction aborts, changes made to database are not visible.

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

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

Consider the following transaction **T** consisting of **T1** and **T2**: Transfer

of 100 from account **X** to account **Y**.

Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X − 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

If the transaction fails after completion of **T1** but before completion of **T2**.(say, after **write(X)** but before **write(Y)**), then amount has been deducted from **X** but not added to **Y**. This results in an inconsistent database state. Therefore, the transaction must be executed in entirety in order to ensure correctness of database state.

Consistency

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to correctness of a database. Referring to the example above,

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, database is **consistent**. Inconsistency occurs in case **T1** completes but **T2** fails. As a result T is incomplete.

Isolation

This property ensures that multiple transactions can occur concurrently without leading to inconsistency of database state. Transactions occur

independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

Let **X**= 500, **Y** = 500.

Consider two transactions **T** and **T".**

T	T"
Read (X) X: = X*100 Write (X) Read (Y) Y: = Y – 50 Write	Read (X) Read (Y) Z: = X + Y Write (Z)

Suppose **T** has been executed till **Read (Y)** and then **T''** starts. As a result , interleaving of operations takes place due to which **T''** reads correct value of **X** but incorrect value of **Y** and sum computed by

T'': (X+Y = 50, 000+500=50, 500)

is thus not consistent with the sum at end of transaction:

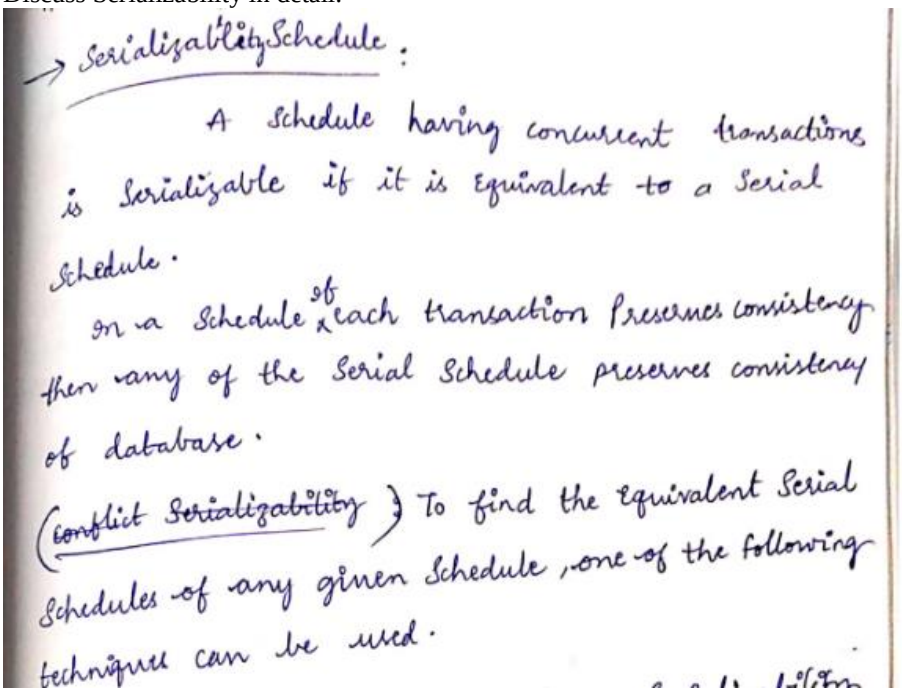
T: (X+Y = 50, 000 + 450 = 50, 450).

This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after a they have been made to the main memory.

Durability:

This property ensures that once the transaction has completed execution,

	the updates and modifications to the database are stored in and written to disk and they persist even if system failure occurs. These updates now become permanent and are stored in a non-volatile memory. The effects of the transaction, thus, are never lost. The **ACID** properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.		
2 ans	Discuss How do you implement Atomicity and Durability. **Implementation of Atomicity and Durability** The recovery-management element of a database system can support atomicity and durability by a range of plans. This plan, which is based on making copies of the database, called shadow copies, presumes that just one deal is active at a time. In the shadow-copy plan, a deal that desires to upgrade the database first develops a total copy of the database. If at any point the deal has actually to be terminated, the system simply erases the brand-new copy. If the deal finishes, it is dedicated as follows. After the operating system has actually composed all the pages to disk, the database system updates the guideline db-pointer to point to the brand-new copy of the database; the brand-new copy then ends up being the present copy of the database. **Atomicity and Durability** ● - Atomicity handle these failures: ● - User terminates deal (e.g., cancel button). ● - System terminates deal (e.g., deadlock). ● - Transaction terminates itself (e.g., unanticipated db state). ● - System crashes. ● Durability handle this kind of failure:. ● - Media failure.	Understand	CACS005.18
3 ans	Illustrate Concurrent execution of transaction with examples/ **Concurrent execution** of database **transactions** in a multi-user system means that any number of users can use the same database at the same time. **Concurrency**control is needed in order to avoid inconsistencies in the database. Concurrent execution of database transactions in a multi-user system means that any number of users can use the same database at the same time. Concurrency control is needed in order to avoid inconsistencies in the database. Here is an example of how this scenario can lead to an inconsistency: Assume we have two users, Alice and Bob, who both have access to	Remember	CACS005.19

	the same bank account. Alice wants to deposit $100 and Bob wants to withdraw $200. Assuming there is $500 in the account, here is how the execution *might* take place if they perform their actions at the same time: 1. Alice gets initial amount (x = $500) 2. Bob gets initial amount (x = $500) 3. Alice deposits $100 (x + 100) = $600 4. Bob withdraws $200 (x - 200) = $300 5. Alice saves the new balance ($600) 6. Bob saves the new balance ($300) New balance after both actions should be $400. Now the database is in an inconsistent state. Concurrency control can prevent inconsistencies by providing Alice with a temporary "lock" on the database until she is done with her action.		
4 ans	Discuss Serializability in detail.  → Serializability Schedule : A Schedule having concurrent transactions is Serializable if it is Equivalent to a Serial Schedule. In a Schedule of each transaction Preserves consistency then any of the Serial Schedule preserves consistency of database. (Conflict Serializability) To find the Equivalent Serial Schedules of any given Schedule, one of the following techniques can be used.	Remember	CACS005.19

techniques can

① Conflict Serializability ② View Serializability

*Note: on a schedule, we will consider only read write operations which effects our database. Remaining operators are considered as arbitary operations.

→ Two instructions I_i, I_j of Transactions T_i, T_j respectively. These two instructions are conflict if and only if both transactions are accessing some data item Q, and atleast one of them is Write(Q).

Note: dbms sometimes execute transactions which are not equivalent to any serial schedule. The reasons may be

① DBMS may use some other concurrence control mechanism which ensures the given schedule is serializable.

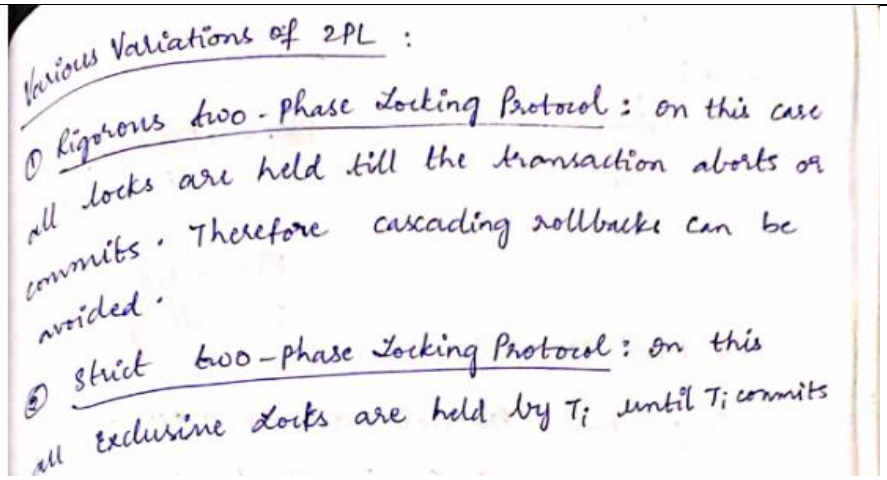
② SQL Provides an opportunity to users to instruct dbms to accept non serializable schedules

View Serializability

Let S and S' be two schedules with the same set of transactions. S and S' are view equivalent if the following three conditions are satisfied. for each data item Q

1. if in schedule S, T_i reads the value of Q initially, then in S' also T_i must read the initial value of Q.

	2) If in Schedule S, execute read(Q) by T_i and which was produced by T_j then in S' also T_i must read the value of Q which was written by write(Q) of T_j 3) The transaction which performs final write(Q) in Schedule S must also perform final write(Q) in S' also. → A Schedule S is View equivalent to a Serial Schedule then the Schedule is known as View Serializable. → Any conflict Serializable Schedule is also View Serializable but Vice-Versa is not true.		
5 ans	Discuss two phase locking protocol and strict two phase locking protocols. **Lock-based Protocols :** Lock-based Protocols use granting of locks on data items which are requested by transaction and that transactions cannot read or write a data item until it acquires appropriate lock on data item. These Protocols may solve Schedules which are not Serializable ① Two phase Locking Protocol (2PL) → Growing Phase (also Locking phase) → Shrinking phase (unlocking or releasing phase) → 2PL ensures assures Serializability Lock point is a point where transaction acquires its final Lock-based upon the Lock point Serializability will be assured. 1. 2PL doesn't ensure freedom from deadlocks Cascading rollbacks also possible	Understand	CACS005.19



6 ans	Describe Timestamp based locking protocols.
	
	Timestamp-based Protocols
	
	The timestamp-based algorithm uses a timestamp to serialize the execution of concurrent transactions. This protocol ensures that every conflicting read and write operations are executed in timestamp order. The protocol uses the **System Time or Logical Count as** a Timestamp.
	The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.
	Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created.
	Example:
	`Suppose there are there transactions T1, T2, and T3.`
	`T1 has entered the system at time 0010`
	`T2 has entered the system at 0020`
	`T3 has entered the system at 0030`
	`Priority will be given to transaction T1, then transaction T2 and lastly Transaction T3.`
	
	Advantages:
	
	● Schedules are serializable just like 2PL protocols
	● No waiting for the transaction, which eliminates the possibility of deadlocks!
	
	Disadvantages:
	Starvation is possible if the same transaction is restarted and continually aborted
		Remember	CACS005.19

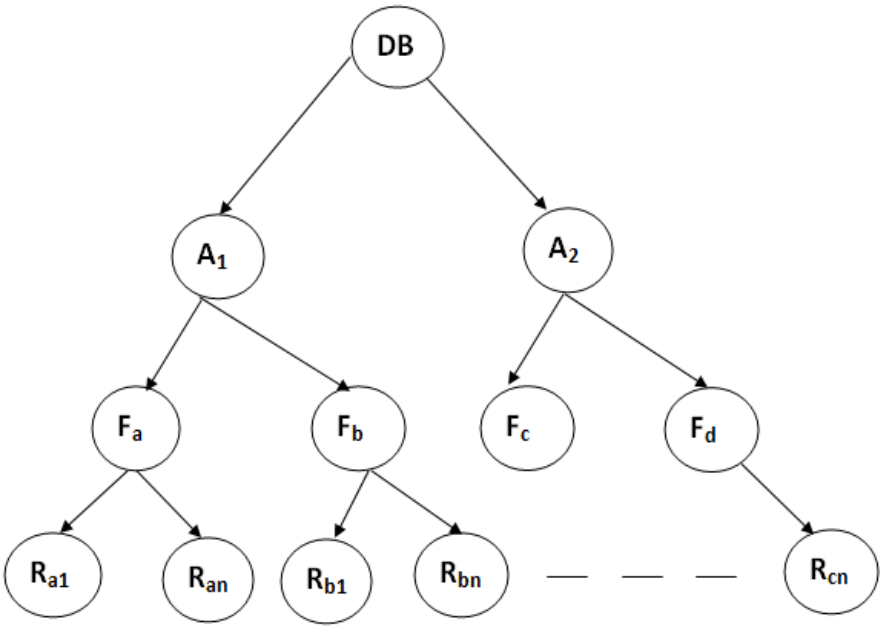
7 ans	Describe Validation-based locking protocols. # Validation Based Protocol Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases: 1. **Read phase:** In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database. 2. **Validation phase:** In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability. 3. **Write phase:** If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back. Here each phase has the following different timestamps: **Start(Ti):** It contains the time when Ti started its execution. **Validation (T_i):** It contains the time when Ti finishes its read phase and starts its validation phase. **Finish(Ti):** It contains the time when Ti finishes its write phase. • This protocol is used to determine the time stamp for the transaction for serialization using the time stamp of the validation phase, as it is the actual phase which determines if the transaction will commit or rollback. • Hence TS(T) = validation(T). • The serializability is determined during the validation process. It can't be decided in advance. • While executing the transaction, it ensures a greater degree of concurrency and also less number of conflicts. • Thus it contains transactions which have less number of rollbacks. •	Remember	CACS005.19
8 ans	Discuss in detail Multiple Granularity. **Granularity:** It is the size of data item allowed to lock.	Understand	CACS005.20

Multiple Granularity:

- It can be defined as hierarchically breaking up the database into blocks which can be locked.
- The Multiple Granularity protocol enhances concurrency and reduces lock overhead.
- It maintains the track of what to lock and how to lock.
- It makes easy to decide either to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.

For example: Consider a tree which has four levels of nodes.

- The first level or higher level shows the entire database.
- The second level represents a node of type area. The higher level database consists of exactly these areas.
- The area consists of children nodes which are known as files. No file can be present in more than one area.
- Finally, each file contains child nodes known as records. The file has exactly those records that are its child nodes. No records represent in more than one file.
- Hence, the levels of the tree starting from the top level are as follows:
    1. Database
    2. Area
    3. File
    4. Record

	 **Figure:** Multi Granularity tree Hierarchy		
9 ans	Explain in detail Storage structure. We have already described the storage system. In brief, the storage structure can be divided into two categories − • Volatile storage − As the name suggests, a volatile storage cannot survive system crashes. Volatile storage devices are placed very close to the CPU; normally they are embedded onto the chipset itself. For example, main memory and cache memory are examples of volatile storage. They are fast but can store only a small amount of information. • Non-volatile storage − These memories are made to survive system crashes. They are huge in data storage capacity, but slower in accessibility. Examples may include hard-disks, magnetic tapes, flash memory, and non-volatile (battery backed up) RAM.	Remember	CACS005.20
10	Discuss Deferred database modification and Immediate database modification.	Remember	CACS005.20

ans

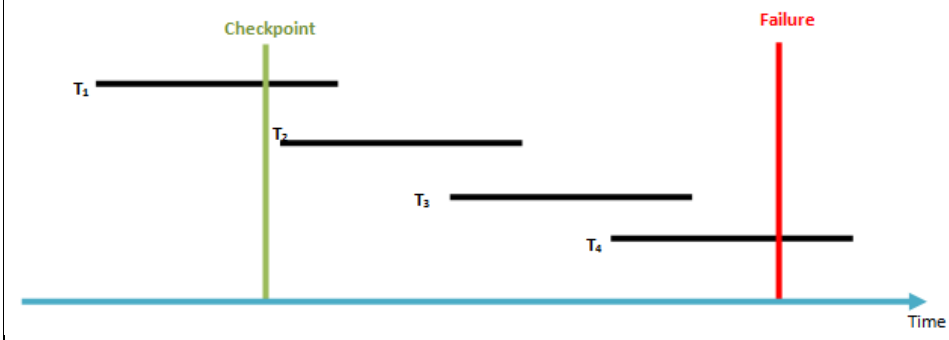
1. **Deferred Modification Technique:** If the transaction does not modify the database until it has partially committed, it is said to use deferred modification technique.
2. **Immediate Modification Technique:** If database modification occur while transaction is still active, it is said to use immediate modification technique.

SN	Deferred Modification	Immediate Modification
1	This scheme is easier to implement as fewer operations are needed. Only redo operation is required after a system failure.	Both redo and undo operations are required after a system failure.
2	No extra I/O operations are needed before commit time.	There could be extra I/O operation by operating system to flush out the block-buffer.
3	It does not require old value of data items on the log.	Both old and new value of data items are written on the log.
4	If a transaction requires to read a data item modified by other transaction, it cannot do it as change may not have gone to database.	Due to immediate modification, a transaction will get the modified value of a data item. This will mean a higher concurrency.
5	Write blocks are held much longer. These locks are held till the commit point. This will lead to a lower concurrency.	Write blocks can be released after modification and hence a higher concurrency.
6	For long transaction the memory needed for log and local variables could be very high.	It can manage with less memory space.

11 Discuss how do you recover from Concurrent transactions. | Remember | CACS005.20

Recovery with Concurrent Transaction

- Whenever more than one transaction is being executed, then the interleaved of logs occur. During recovery, it would become difficult for the recovery system to backtrack all logs and then start recovering.
- To ease this situation, 'checkpoint' concept is used by most DBMS.
- When more than one transaction are being executed in parallel, the logs are interleaved. At the time of recovery, it would become hard for the recovery system to backtrack all logs, and then start recovering. To ease this situation, most modern DBMS use the concept of 'checkpoints'.

	• Checkpoint • Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. As time passes, the log file may grow too big to be handled at all. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed. • Recovery • When a system with concurrent transactions crashes and recovers, it behaves in the following manner – • 		
12 ans	Explain Buffer Management. **BUFFER MANAGEMENT** The **buffer manager** is the software layer that is responsible for bringing pages from physical disk to main memory as needed. The buffer manages the available main memory by dividing the main memory into a collection of pages, which we called as **buffer pool.** The main memory pages in the buffer pool are called **frames.** • **Data must be in RAM for DBMS to operate on it!** • **Buffer manager hides the fact that not all data is in RAM.** The goal of the buffer manager is to ensure that the data requests made by programs are satisfied by copying data from secondary storage devices into buffer. Infact, if a program performs reading from existing buffers. similarly, if a program performs an output statement: it calls the buffer manager for	Understand	CACS005.20

	output operation - to satisfy the requests by writing to the buffers. Therefore, we can say that input and output operation occurs between the program and the buffer area only.		
13 ans	Explain different types of Advanced Recovery Techniques. ● Recovery algorithms are techniques to ensure transaction atomicity and durability despite failures. The recovery subsystem, using recovery algorithm, ensures atomicity by undoing the actions of transactions that do not commit and durability by making sure that all actions of committed transactions survive even if failures occur. The recovery procedure involves flushing, its logs stored in data buffers into disk ● Strategies that can be used when flushing occurs. ● In-place updating: - Writes the buffer back to the same original disk location – overwriting the old value on disk. ● Shadowing: - Writes the updated buffer at a different disk location. – Multiple versions of data items can be maintained. ● Two main approaches in recovery process a) Log-based recovery using WAL protocol. b) Shadow-paging 1. **Write-Ahead Logging (WAL)** ○ Two types of log entry –log record- information for a write command. The information needed for UNDO. ○ A UNDO-type log entries including the old values (BFIM). ○ Since this is needed to undo the effect of the operations from the log. ○ The information needed for REDO. ○ A REDO-type log entries including the new values (AFIM). ○ Since it is needed to redo the effect of the operations from the log ○ In UNDO/REDO algorithms, both types of log entries are combined. ○ The log includes read commands only when cascading rollback is possible ○ WAL protocol for a recovery algorithm that requires both UNDO and REDO. ○ The before image of an item cannot be overwritten by its after image in the database on disk until all UNDO-type log records for the updating transaction – up to this point in time- have been force-written to disk. Ensures atomicity. ○ The commit operation of a transaction cannot be completed until all the REDO-type and UNDO-type log records for that transaction have been force-written to disk. Ensures durability. 2. **Steal/no-steal-- Force/no-force Approaches**	Remember	CACS005.20

No-steal approach

1) A cache page updated by a transaction cannot be written to disk before the transaction commits.-deferred update follows this approach-

2) The pin-unpin bit indicates if a page cannot be written back to disk

-

Steal approach

1) An updated buffer can be written before the transaction commits.

Used when the buffer manager replaces an existing page in the cache, that has been updated by a transaction not yet committed, by another page requested by another transaction.

2) Advantage: avoid the need for a very large buffer space.

-

Force/No-Force approaches

1) Force approach if all pages updated by a transaction are immediately written to disk when the transaction commits

2) No-force approach otherwise.

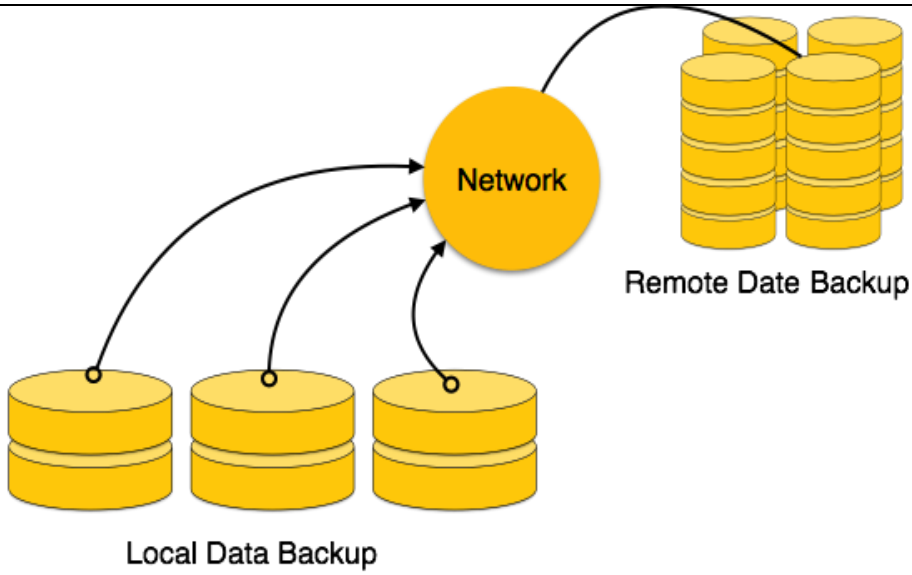
-

  Advantage: an updated page of a committed transaction may still be in the buffer when another transaction needs to update it.-save in I/O cost. Typical database systems follow a steal/no-force strategy.

3. **Checkpoints**
   - In case of failure, the recovery manager requires that the entire log be examined to process recovery→ time consuming
   - A quick way to limit the amount of log to scan on recovery can be achieved using checkpoints.
   - A [checkpoint] record is written into the log periodically at that point when the system writes out to the database on disk all DBMS buffers that have been modified.
   - Hence, all transactions with [commit, T] entry in the log before [checkpoint] entry do not need to have their WRITE operations redone in case of crash.
   - Since all their update have been recorded in the DB on disk during check-pointing.
4. **Shadow Paging**
   - A recovery scheme
     - In a single-user environment, doesn't require the use of log.
     - In multi-user environment, the log may be needed for concurrency control method.
   - The DB is made up of n fixed-size disk pages -

	blocks-
	A directory with n entries where the ith entry points to the ith DB page on disk.
	`All references –reads or writes- to the DB pages on disk go through the directory.`
	`The directory is kept in main memory if not too large.`
	o
	When a transaction begins executing, the current directory is copied into a shadow directory and the shadow directory is saved on disk
	`The current directory entries point to the most recent or current DB pages on disk`
	o
	○ During transaction execution, all updates are performed using the current directory and the shadow directory is never modified.
	○ When a write_item operation is performed A new copy of the modified DB page is created and the old copy is not overwritten. Two version, of the pages updated by the transaction, are kept. The new page is written elsewhere on some unused disk block. The current directory entry is modified to point to the new disk block. The shadow directory is not modified.
	○ To recover from a failure Free the modified database pages and discard the current directory. Reinstate the shadow directory to recover the state of the DB before transaction execution. Return to the state prior to the transaction that was executing when the crash occurred.
	○ To commit a transaction Discard the previous shadow directory.
	○ NO-UNDO/NO-REDO technique since neither undoing or redoing of data items		

14 ans	Write in detail about Remote Backup systems. # Remote Backup Remote backup provides a sense of security in case the primary location where the database is located gets destroyed. Remote backup can be offline or real-time or online. In case it is offline, it is maintained manually.	Understand	CACS005.20



Online backup systems are more real-time and lifesavers for database administrators and investors. An online backup system is a mechanism where every bit of the real-time data is backed up simultaneously at two distant places. One of them is directly connected to the system and the other one is kept at a remote place as backup.

As soon as the primary database storage fails, the backup system senses the failure and switches the user system to the remote storage. Sometimes this is so instant that the users can't even realize a failure.

15 ans

Explain the Check point log based recovery scheme for recovering the database.

Remember

CACS005.20

Log-based Recovery

Log is a sequence of records, which maintains the records of actions performed by a transaction. It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe.

Log-based recovery works as follows –

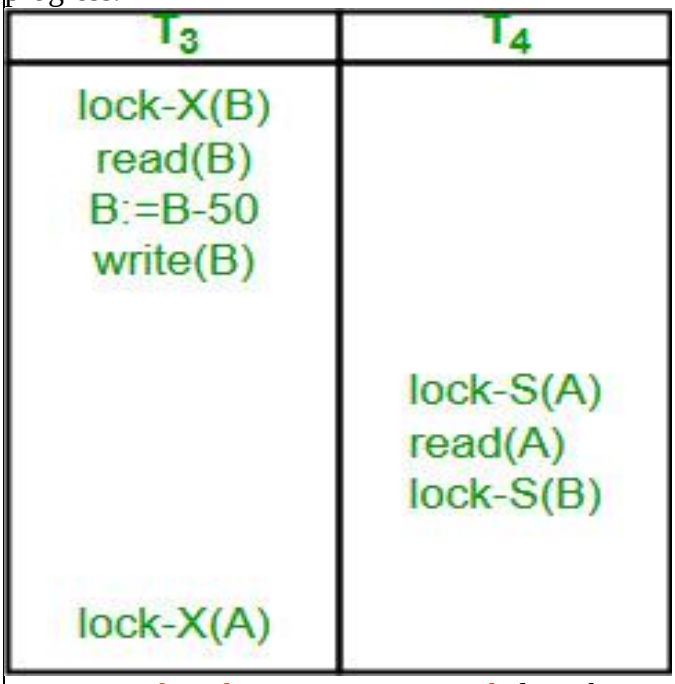
- The log file is kept on a stable storage media.
- When a transaction enters the system and starts execution, it writes a log about it.

<T_n, Start>

	• When the transaction modifies an item X, it write logs as follows – $<T_n, X, V_1, V_2>$ It reads T_n has changed the value of X, from V_1 to V_2. • When the transaction finishes, it logs – $<T_n, commit>$ The database can be modified using two approaches – • Deferred database modification – All logs are written on to the stable storage and the database is updated when a transaction commits. • Immediate database modification – Each log follows an actual database modification. That is, the database is modified immediately after every operation.		
16 ans	When a transaction is rolled back under timestamp ordering, it is assigned a new timestamp. Why can it not simply keep its old timestamp?	Remember	CACS005.20
PART – C (Problem Solving and Critical Thinking Questions)			

1	Consider the following transactions with data items P and Q initialized to zero:
	T1: read(P);
	read(Q);
	If P=0 then Q:=Q+1;
	write(Q);
	T2: read(Q);
	read(P);
	If Q=0 then P:=P+1;
	write(P);
	Solve and find any non-serial interleaving of T1 and T2 for concurrent execution leads to a serializable schedule or non serializable schedule. Explain.
		Understand	CACS005.21

ans	Two or more actions are said to be in conflict if: 1) The actions belong to different transactions. 2) At least one of the actions is a write operation. 3) The actions access the same object (read or write). The schedules S1 and S2 are said to be conflict-equivalent if the following conditions are satisfied: 1) Both schedules S1 and S2 involve the same set of transactions (including ordering of actions within each transaction). 2) The order of each pair of conflicting actions in S1 and S2 are the same. *A schedule is said to be conflict-serializable when the schedule is conflict-equivalent to one or more serial schedules.* In the given scenario, there are two possible serial schedules: 1) T1 followed by T2 2) T2 followed by T1. In both of the serial schedules, one of the transactions reads the value written by other transaction as a first step. Therefore, any non-serial interleaving of T1 and T2 will not be conflict serializable.		

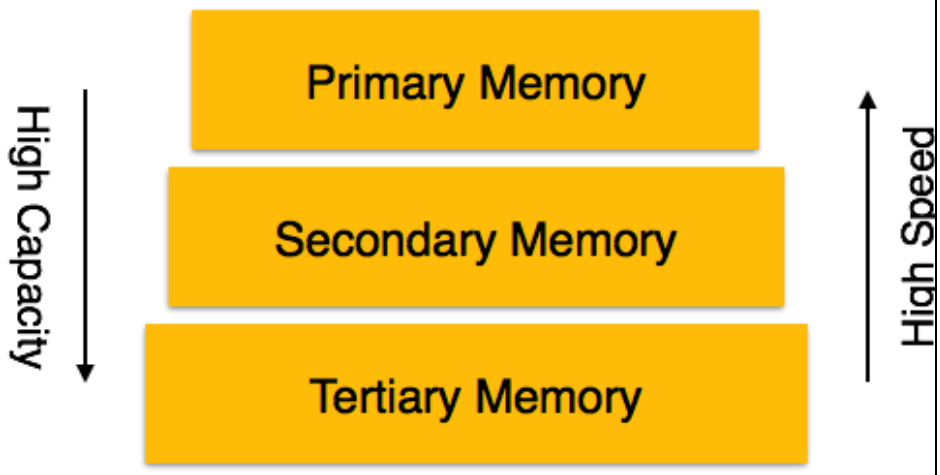
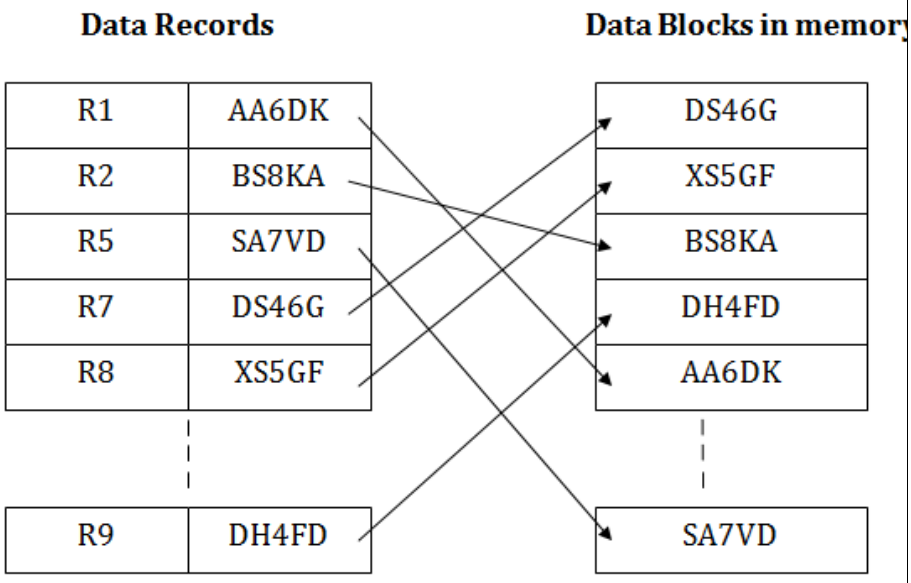
2 ans	Analyze which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock? Explain the following: a. 2-phase locking b. Time-stamp ordering 2 Phase Locking (2PL) is a concurrency control method that guarantees serializability. The protocol utilizes locks, applied by a transaction to data, which may block (interpreted as signals to stop) other transactions from accessing the same data during the transaction's life. 2PL may be lead to deadlocks that result from the mutual blocking of two or more transactions. See the following situation, neither T3 nor T4 can make progress.  Timestamp-based concurrency control algorithm is a non-lock concurrency control method. In Timestamp based method, deadlock cannot occur as no transaction ever waits.	Remember	CACS005.19
3	Suppose that we have only two types of transactions, T1 and T2. Transactions preserve database consistency when run individually. We have defined several integrity constraints such that the DBMS never executes any SQL statement that brings the database into an inconsistent state. Assume that the DBMS does not perform any concurrency control. Give an example schedule of two transactions T 1 and T 2 that satisfies all these conditions, yet produces a database instance that is not the result of any serial execution of T 1 and T 2.	Understand	CACS005.19
4 ans	Suppose that there is a database system that never fails.Analyze whether a recovery manager required for this system. **Answer:** Even in this case the recovery manager is needed to perform roll-back of aborted transactions.	Remember	CACS005.21
5 ans	Explain the 'Immediate database Modification' technique for using the Log to Ensure transaction atomicity despite failures.	Remember	CACS005.19

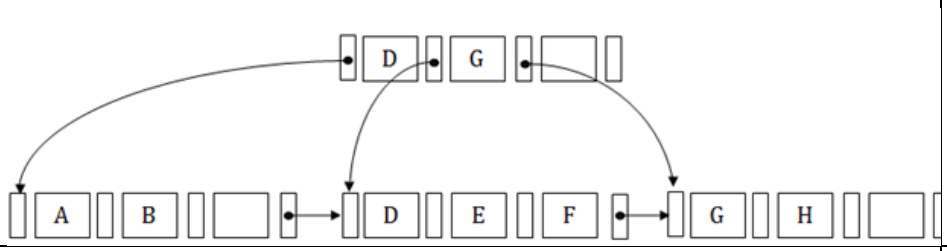
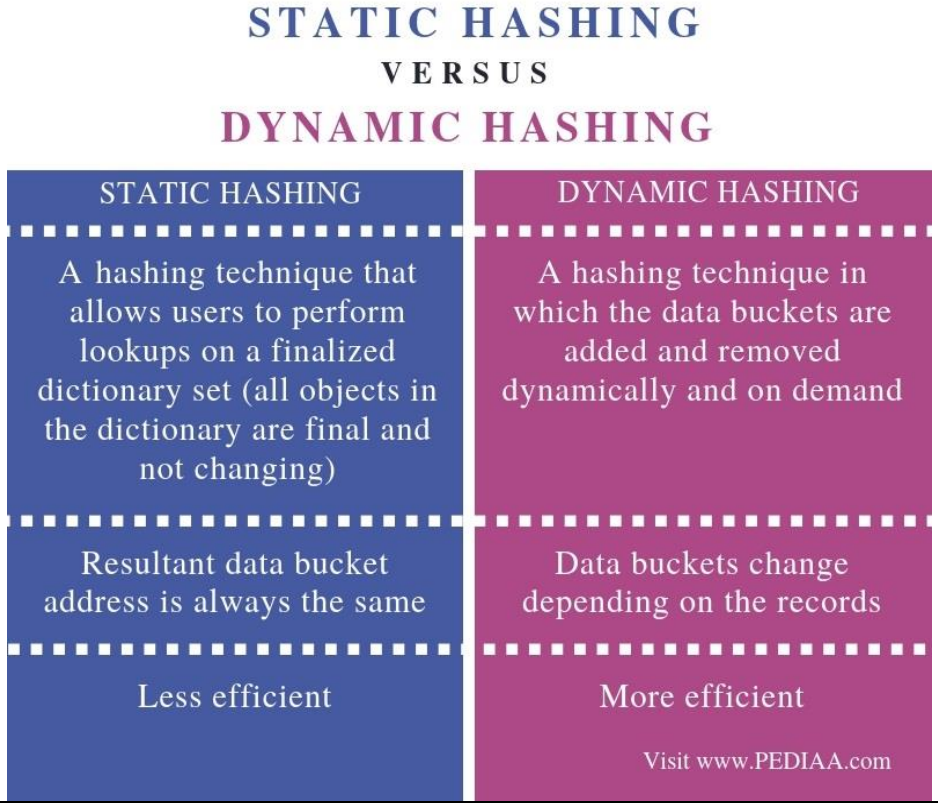
6 ans	Consider the following actions taken by transaction T 1 on database objects X and Y : R(X), W(X), R(Y), W(Y) 1. Give an example of another transaction T 2 that, if run concurrently to transaction T without some form of concurrency control, could interfere with T 1. 2. Explain how the use of Strict 2PL would prevent interference between the two transactions. 3. Strict 2PL is used in many database systems. Give two reasons for its popularity. 1. If the transaction T2 performed W(Y) before T1 performed R(Y), and then T2 aborted, the value read by T1 would be invalid and the abort would be cascaded to T1 (i.e. T1 would also have to abort.). 2. Strict 2PL would require T2 to obtain an exclusive lock on Y before writing to it. This lock would have to be held until T2 committed or aborted; this would block T1 from reading Y until T2 was finished, but there would be no interference. 3. Strict 2PL is popular for many reasons. One reason is that it ensures only 'safe' interleaving of transactions so that transactions are recoverable, avoid cascading aborts, etc. Another reason is that strict 2PL is very simple and easy to implement. The lock manager only needs to provide a lookup for exclusive locks and an atomic locking mechanism (such as with a semaphore).	Remember	CACS005.13
7 ans	Suppliers(sid: integer, sname: string, address: string) Parts(pid: integer, pname: string, color: string) Catalog(sid: integer, pid: integer, cost: real) The Catalog relation lists the prices charged for parts by Suppliers. For each of the following transactions, state the SQL isolation level that you would use and explain why you chose it. 1. A transaction that adds a new part to a supplier's catalog. 2. A transaction that increases the price that a supplier charges for a part. 3. A transaction that determines the total number of items for a given supplier. 4. A transaction that shows, for each part, the supplier that supplies the part at the lowest price	Understand	CACS005.19
8	Answer each of the following questions briefly. The questions are based on the following relational schema: Emp(eid: integer, ename: string, age: integer, salary: real, did: integer) Dept(did: integer, dname: string, floor: integer) and on the following update command: replace (salary = 1.1 * EMP.salary) where EMP.ename = 'Santa' Give an example of a query that would conflict with this command (in a concurrency control sense) if both were run at the same time. 1.Explain what could go wrong, and how locking tuples would solve the problem.	Remember	CACS005.19

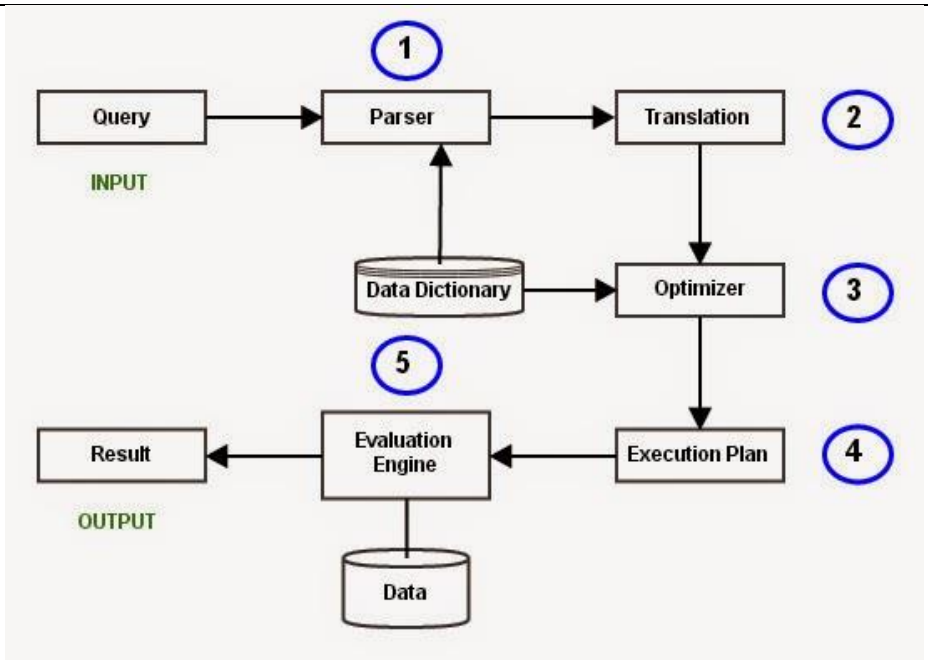
	2. Give an example of a query or a command that would conflict with this command, such that the conflict could not be resolved by just locking individual tuples or pages but requires index locking. 3. Explain what index locking is and how it resolves the preceding conflict.		
9	Suppose that we have only two types of transactions, T 1 and T 2. Transactions preserve database consistency when run individually. We have defined several integrity constraints such that the DBMS never executes any SQL statement that brings the database into an inconsistent state. Assume that the DBMS does not perform any concurrency control. Give an example schedule of two transactions T 1 and T 2 that satisfies all these conditions, yet produces a database instance that is not the result of any serial execution of T 1 and T 2.	Remember	CACS005.21
10 ans	What are the roles of the Analysis, Redo, and Undo phases in ARIES? ● In ARIES, a log consists of multiple log files. Each file has a specific capacity. If the 1st log file grows to some limit, then further log records are appended to a new log file. Each log file maintains a file n, which monotonically increases ● Each of this log file has been associated with an LSN, which is the combination of file no. & the offset of that particular file. ● Earn file contains a particular data page & the latest event related to that data page. Thus the corresponding LSN of a log file indicates particular operation of the stored data page during recovery procedure. It includes 3 phases:- **ANALYSIS PHASE :-** ● The dirty page table has been formed by analysing the pages from the buffer and also a set of active transactions has been identified. When the system encounters crash, ARIES recovery manager starts the analysis phone by finding the last checkpoint log record after that, it prepares dirty pages tables this phone mainly prepares a set of active transactions that are needed to be undo analysis phone, after getting the last checkpoint log record, log record is scanned in forward direction, & update of the set of active transactions, transaction value, & dirty page table are done in the following manner :- ● 1st recovery manager finds, any transaction which is not in the active transaction set, then add that transaction in that set if it finds an end log record, then that record has been deleted from the transaction table. ● If it finds a log record that describes an update operation, the log record has been added to the dirty page table. **REDO PHASE:-** ● Redo phase is the second phase where all the transactions that are needed to be executed again take place. ● It executes those operations whose results are not reflected in the disk. ● It can be done by finding the smallest LSN of all the dirty page in dirty page table that defines the log positions, & the Redo operation will start from this position ● This position indicates that either the changes that are made earlier are I the main memory or they have already	Understand	CACS005.20

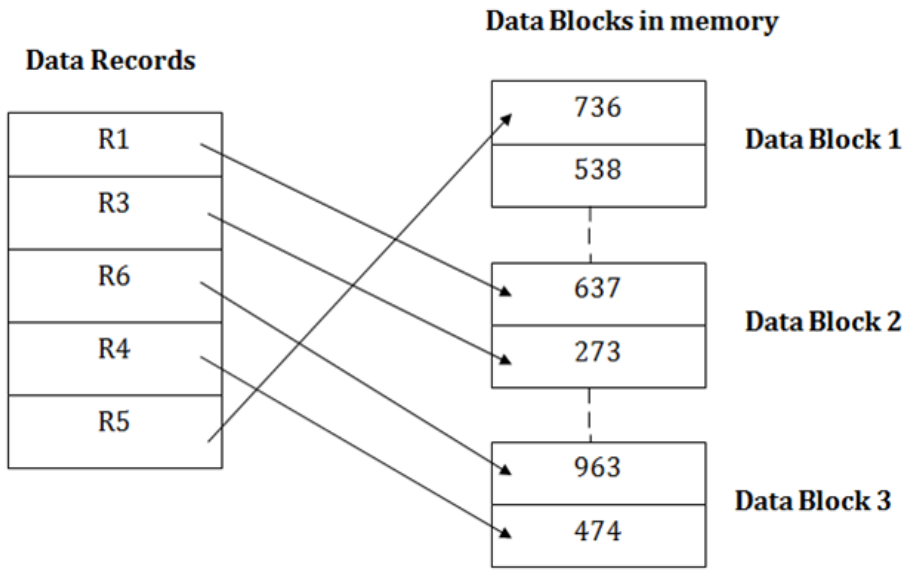
	been flaunted to the disk. ● Thus, for each change recorded in the log, the Redo phase determines whether or not the operations have been re-executed. **UNDO PHASE:-** ● In the Undo phase, all the transaction, that is listed in the active transaction set here to be undone. ● Thus the log should be scanned background from the end & the recovery manages should Undo the necessary operations. ● Each time an operations is undone, a compensation log recorded has been written to the log. ● This process continues until there is no transaction left in the active transaction set. ● After the successful competition of this phase, database can resume its normal operations.		
	UNIT – V **Data Storage and Query Processing**		
	PART – A (Short Answer Questions)		
1 ans	Discuss about data on External storage. *Data on External Storage* ❖ Disks: Can retrieve random page at fixed cost ▪ But reading several consecutive pages is much cheaper than reading them in random order ❖ Tapes: Can only read pages in sequence ▪ Cheaper than disks; used for archival storage ❖ File organization: Method of arranging a file of records on external storage. ▪ Record id (rid) is sufficient to physically locate record ▪ Indexes are data structures that allow us to find the record ids of records with given values in index search key fields ❖ Architecture: Buffer manager stages pages from external storage to main memory buffer pool. File and index layers make calls to the buffer manager.	Understand	CACS005.22
2 ans	Illustrate Clustered Indexes. Clustering Index ● A clustered 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 get the unique value and create index out of them. This method is called a clustering index.	Remember	CACS005.23

	• The records which have similar characteristics are grouped, and indexes are created for these group.		
3 ans	Discuss the Primary and Secondary indexes. **Primary Index** • If the index is created on the basis of the primary key of the table, then it is known as primary indexing. These primary keys are unique to each record and contain 1:1 relation between the records. • As primary keys are stored in sorted order, the performance of the searching operation is quite efficient. • The primary index can be classified into two types: Dense index and Sparse index. **Secondary Index** In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the 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).	Understand	CACS005.23
4 ans	Define Tree Indexing. *Tree-Based Indexing* ❖ The data entries are arranged in sorted order by search key value. ❖ A hierarchical search data structure (tree) is maintained that directs searches to the correct page of data entries. ❖ Tree-structured indexing techniques support both *range searches* and *equality searches*.	Understand	CACS005.22
5 ans	Describe Storage Hierarchy. Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device. These storage	Remember	CACS005.23

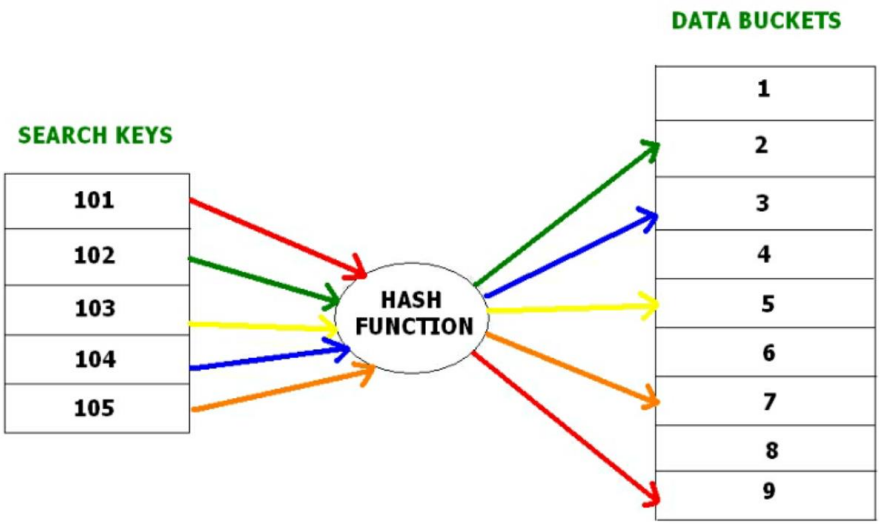
	devices can be broadly categorized into three types –  		
6 ans	Discuss the intuition for Tree Indexes.	Understand	CACS005.22
7 ans	Define Indexed Sequential Access Method. **Indexed sequential access method (ISAM)** ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key. An index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file.  If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.	Understand	CACS005.23
8 ans	Discuss about Overflow pages and Locking considerations of ISAM.	Understand	CACS005.23
9 ans	Describe structure of B+ tree node.	Understand	CACS005.24

			
10 ans	Compare dynamic and static hash techniques. 	Understand	CACS005.24
11	List steps in Query processing.	Understand	CACS005.24

			
12	Discuss the advantages of Heap file organization. **Heap file organization** • It is the simplest and most basic type of organization. It works with data blocks. In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records. • When the data block is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file. • In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.	Remember	CACS005.24



PART – B (Long Answer Questions)

1 ans	Write in detail about Hash based Indexing and Tree based Indexing. HASH BASED INDEXING: **Hashing** is an efficient technique to directly search the location of desired data on the disk without using index structure. Data is stored at the data blocks whose address is generated by using hash function. The memory location where these records are stored is called as data block or data bucket. **Hash File Organization :**  Hashing is further divided into two sub categories : **Static Hashing –**	Understand	CACS005.23

In static hashing, when a search-key value is provided, the hash function always computes the same address. For example, if we want to generate address for STUDENT_ID = 76 using mod (5) hash function, it always result in the same bucket address 4. There will not be any changes to the bucket address here. Hence number of data buckets in the memory for this static hashing remains constant throughout.

Operations –

- **Insertion –** When a new record is inserted into the table, The hash function h generate a bucket address for the new record based on its hash key K.
  Bucket address = h(K)
- **Searching –** When a record needs to be searched, The same hash function is used to retrieve the bucket address for the record. For Example, if we want to retrieve whole record for ID 76, and if the hash function is mod (5) on that ID, the bucket address generated would be 4. Then we will directly got to address 4 and retrieve the whole record for ID 104. Here ID acts as a hash key.
- **Deletion –** If we want to delete a record, Using the hash function we will first fetch the record which is supposed to be deleted. Then we will remove the records for that address in memory.
- **Updation –** The data record that needs to be updated is first searched using hash function, and then the data record is updated.

Now, If we want to insert some new records into the file But the data bucket address generated by the hash function is not empty or the data already exists in that address. This becomes a critical situation to handle. This situation in the static hashing is called **bucket overflow**.

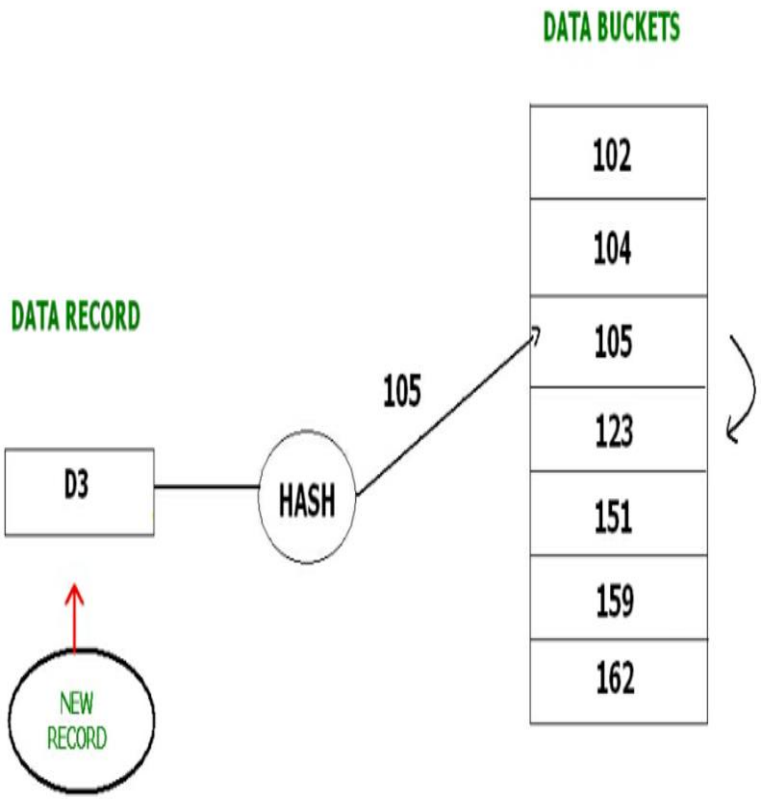
How will we insert data in this case?

There are several methods provided to overcome this situation. Some commonly used methods are discussed below:

1. **Open Hashing –**

   In Open hashing method, next available data block is used to enter the new record, instead of overwriting older one. This method is also called linear probing.

   For example, D3 is a new record which needs to be inserted , the hash function generates address as 105. But it is already full. So the system searches next available data bucket, 123 and assigns D3 to it.

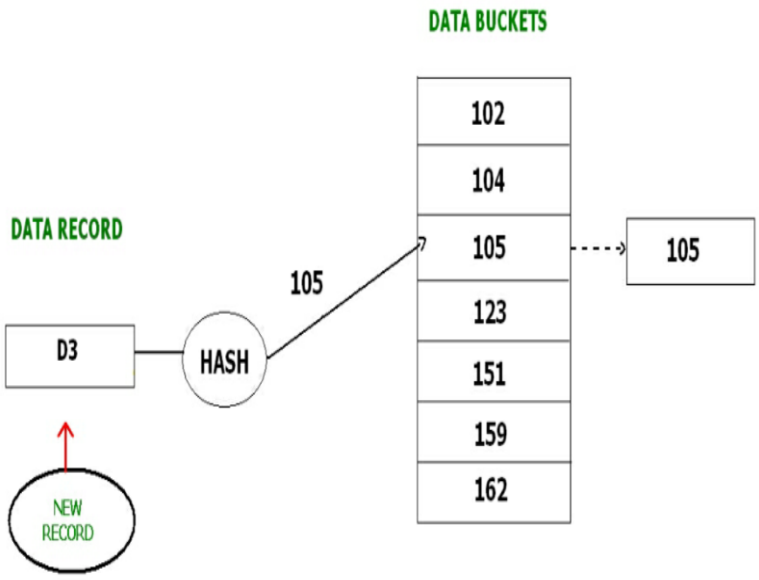


2. **Closed hashing –**

   In Closed hashing method, a new data bucket is allocated with same address and is linked it after the full data bucket. This method is also known as overflow chaining.

   For example, we have to insert a new record D3 into the

tables. The static hash function generates the data bucket address as 105. But this bucket is full to store the new data. In this case is a new data bucket is added at the end of 105 data bucket and is linked to it. Then new record D3 is inserted into the new bucket.



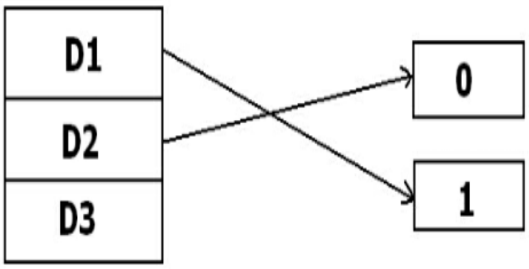
- **Quadratic probing :**
  Quadratic probing is very much similar to open hashing or linear probing. Here, The only difference between old and new bucket is linear. Quadratic function is used to determine the new bucket address.
- **Double Hashing :**
  Double Hashing is another method similar to linear probing. Here the difference is fixed as in linear probing, but this fixed difference is calculated by using another hash function. That's why the name is double hashing.

Dynamic Hashing –

The drawback of static hashing is that that it does not expand or shrink dynamically as the size of the database grows or shrinks. In Dynamic hashing, data buckets grows or shrinks (added or removed dynamically) as the records increases or decreases. Dynamic hashing is also known as extended hashing.

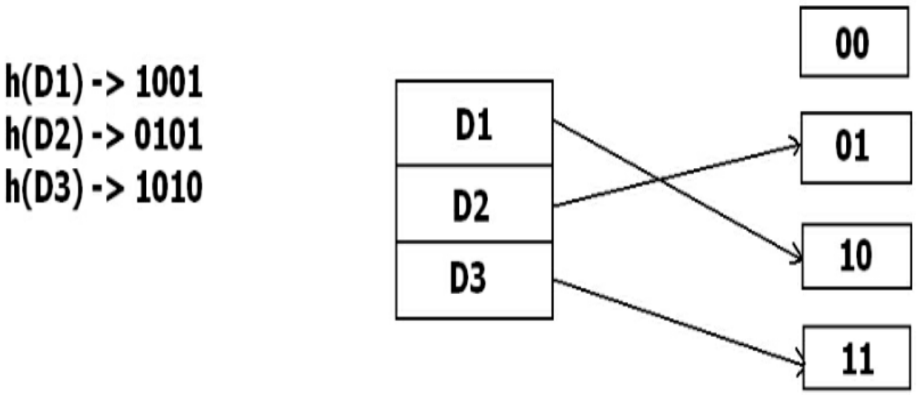
In dynamic hashing, the hash function is made to produce a large number of values. For Example, there are three data records D1, D2 and D3 . The hash function generates three addresses 1001, 0101 and 1010 respectively. This method of storing considers only part of this address – especially only first one bit to store the data. So it tries to load three of them at address 0 and 1.

h(D1) -> 1001
h(D2) -> 0101
h(D3) -> 1010



But the problem is that No bucket address is remaining for D3. The bucket has to grow dynamically to accommodate D3. So it changes the

address have 2 bits rather than 1 bit, and then it updates the existing data to have 2 bit address. Then it tries to accommodate D3.



TREE BASED INDEXING:

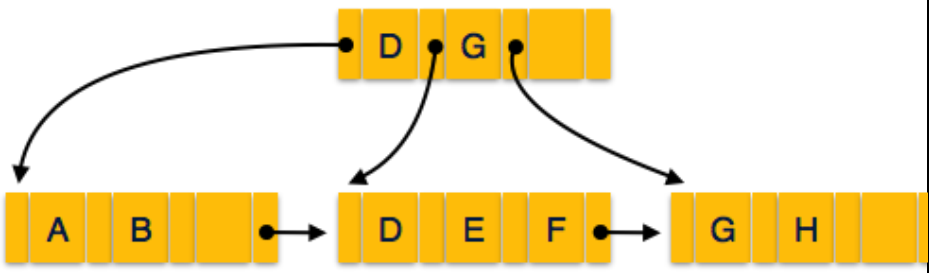
B^+ Tree

A B^+ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B^+ tree denote actual data pointers. B^+ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B^+ tree can support random access as well as sequential access.

Structure of B^+ Tree

Every leaf node is at equal distance from the root node. A B^+ tree is of

the order n where n is fixed for every B^+ tree.



Internal nodes –

- Internal (non-leaf) nodes contain at least ⌈n/2⌉ pointers, except the root node.
- At most, an internal node can contain **n** pointers.

Leaf nodes –

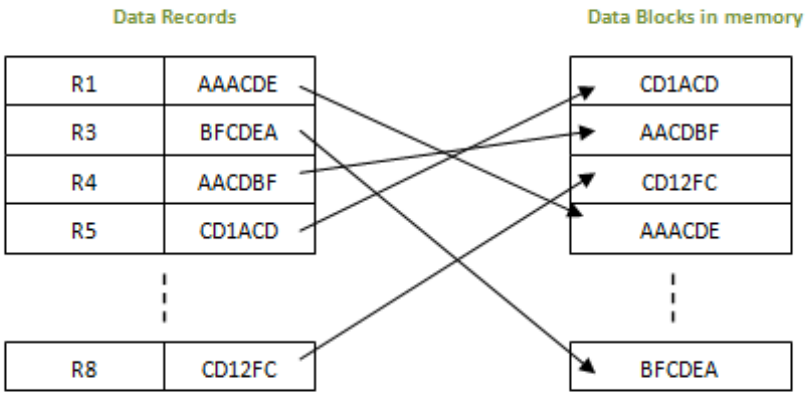
- Leaf nodes contain at least ⌈n/2⌉ record pointers and ⌈n/2⌉ key values.
- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

B^+ Tree Insertion

- B^+ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows –
    - Split node into two parts.
    - Partition at i = ⌊(m+1)/2⌋.
    - First i entries are stored in one node.
    - Rest of the entries (i+1 onwards) are moved to a new node.
    - i^{th} key is duplicated at the parent of the leaf.

	• If a non-leaf node overflows – ○ Split node into two parts. ○ Partition the node at i = ⌈(m+1)/2⌉. ○ Entries up to i are kept in one node. ○ Rest of the entries are moved to a new node. ## B^+ Tree Deletion • B^+ tree entries are deleted at the leaf nodes. • The target entry is searched and deleted. ○ If it is an internal node, delete and replace with the entry from the left position. • After deletion, underflow is tested, ○ If underflow occurs, distribute the entries from the nodes left to it. • If distribution is not possible from left, then ○ Distribute from the nodes right to it. • If distribution is not possible from left or from right, then ○ Merge the node with left and right to it.		
2 ans	Compare I/O costs for all File Organizations. **Comparison of I/O Costs** • A heap file has good storage efficiency and supports fast scanning and insertion of records. However, it is slow for searches and deletions. • A sorted file also offers good storage efficiency, but insertion and deletion of records is slow. Searches are faster than in heap files. • A clustered file offers all the advantages of a sorted file and supports inserts and deletes efficiently. Searches are even faster than in sorted files, although a sorted file can be faster when a large number of records are retrieved sequentially, because of blocked I/O efficiencies. • Unclustered tree and hash indexes offer fast searches, insertion, and deletion, but scans and range searches with many matches are slow. Hash indexes are a little faster on equality searches, but they do not support range searches.	Understand	CACS005.22
3 ans	Explain in detail about ISAM. ## Indexed Sequential Access Method (ISAM) This is an advanced sequential file organization method. Here	Remember	CACS005.22

records are stored in order of primary key in the file. Using the primary key, the records are sorted. For each primary key, an index value is generated and mapped with the record. This index is nothing but the address of record in the file.

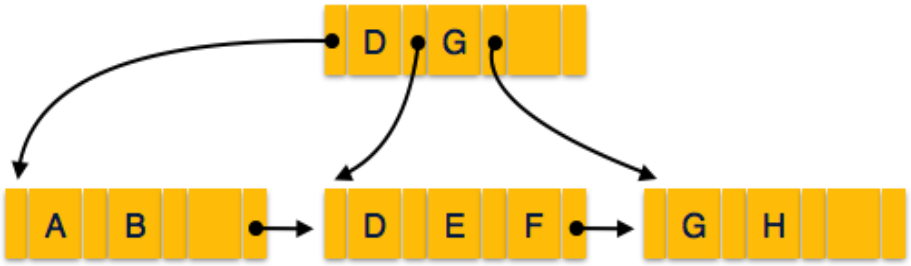


In this method, if any record has to be retrieved, based on its index value, the data block address is fetched and the record is retrieved from memory.

Advantages of ISAM

- Since each record has its data block address, searching for a record in larger database is easy and quick. There is no extra effort to search records. But proper primary key has to be selected to make ISAM efficient.
- This method gives flexibility of using any column as key field and index will be generated based on that. In addition to the primary key and its index, we can have index generated for other fields too. Hence searching becomes more efficient, if there is search based on columns other than primary key.
- It supports range retrieval, partial retrieval of records. Since the index is based on the key value, we can retrieve the data for the given range of values. In the same way, when a partial key value is provided, say student names starting with ‘JA’ can also be searched easily.

Disadvantages of ISAM

	• An extra cost to maintain index has to be afforded. i.e.; we need to have extra space in the disk to store this index value. When there is multiple key-index combinations, the disk space will also increase. As the new records are inserted, these files have to be restructured to maintain the sequence. Similarly, when the record is deleted, the space used by it needs to be released. Else, the performance of the database will slow down.		
4 ans	Explain B+ trees? Discuss about this Dynamic Index Structure. **Structure of B^+ Tree** Every leaf node is at equal distance from the root node. A B^+ tree is of the order n where n is fixed for every B^+ tree.  Internal nodes – • Internal (non-leaf) nodes contain at least ⌈n/2⌉ pointers, except the root node. • At most, an internal node can contain **n** pointers. Leaf nodes – • Leaf nodes contain at least ⌈n/2⌉ record pointers and ⌈n/2⌉ key values. • At most, a leaf node can contain **n** record pointers and **n** key values. • Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.	Understand	CACS005.23

B^+ Tree Insertion

- B^+ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows –
    - Split node into two parts.
    - Partition at i = $\lfloor (m+1)/2 \rfloor$.
    - First i entries are stored in one node.
    - Rest of the entries (i+1 onwards) are moved to a new node.
    - i^{th} key is duplicated at the parent of the leaf.
- If a non-leaf node overflows –
    - Split node into two parts.
    - Partition the node at i = $\lceil (m+1)/2 \rceil$.
    - Entries up to i are kept in one node.
    - Rest of the entries are moved to a new node.

B^+ Tree Deletion

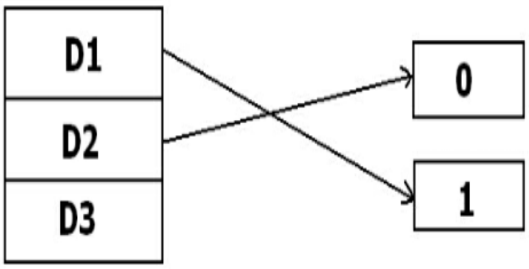
- B^+ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
    - If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,
    - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
    - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
    - Merge the node with left and right to it.

Dynamic Hashing –

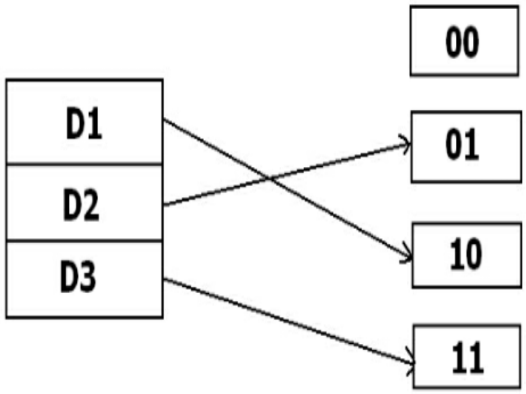
The drawback of static hashing is that that it does not expand or shrink dynamically as the size of the database grows or shrinks. In Dynamic hashing, data buckets grows or shrinks (added or removed dynamically) as the records increases or decreases. Dynamic hashing is also known as extended hashing.

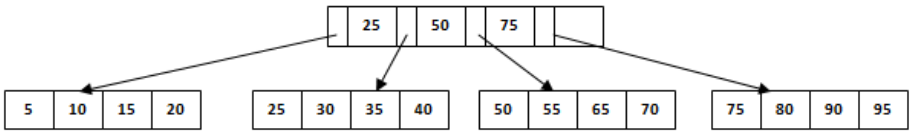
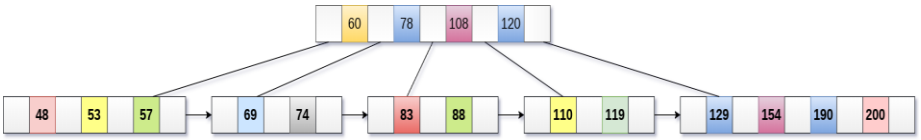
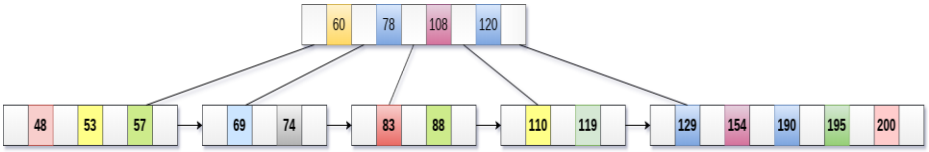
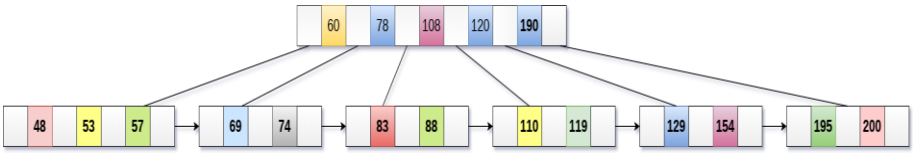
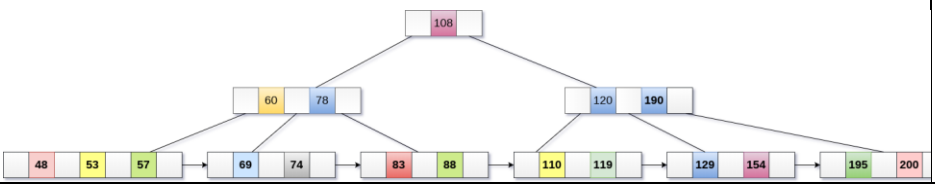
In dynamic hashing, the hash function is made to produce a large number of values. For Example, there are three data records D1, D2 and D3 . The hash function generates three addresses 1001, 0101 and 1010 respectively. This method of storing considers only part of this address – especially only first one bit to store the data. So it tries to load three of them at address 0 and 1.

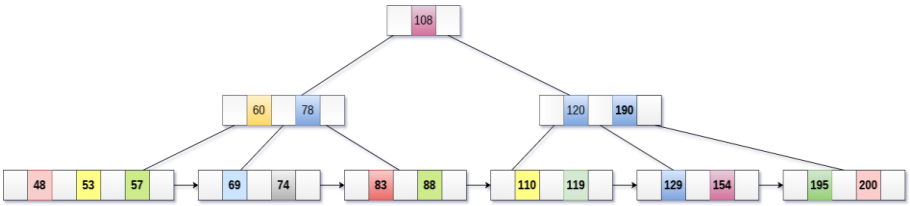
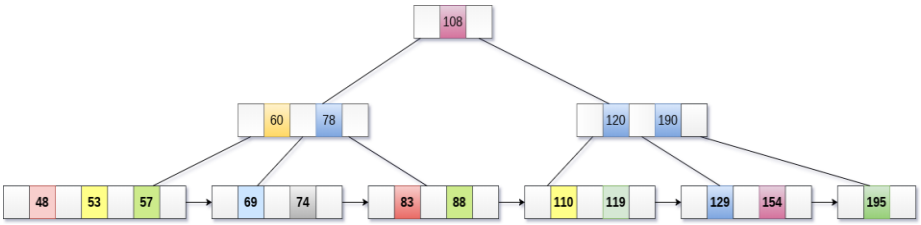
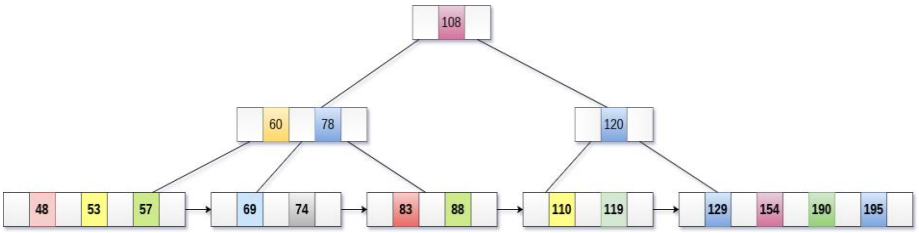
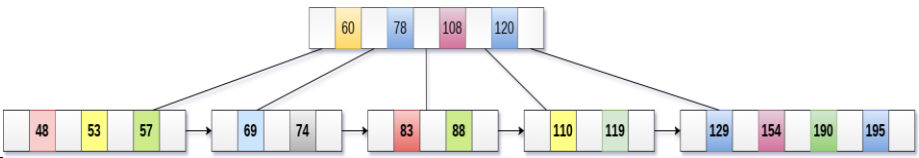
h(D1) -> 1001
h(D2) -> 0101
h(D3) -> 1010



But the problem is that No bucket address is remaining for D3. The bucket has to grow dynamically to accommodate D3. So it changes the

	address have 2 bits rather than 1 bit, and then it updates the existing data to have 2 bit address. Then it tries to accommodate D3. h(D1) -> 1001 h(D2) -> 0101 h(D3) -> 1010 		
5 ans	Demonstrate searching a given element in B+ trees. Explain with example. **Searching a record in B+ Tree** Suppose we want to search 65 in the below B+ tree structure. First we will fetch for the intermediary node which will direct to the leaf node that can contain record for 65. So we find branch between 50 and 75 nodes in the intermediary node. Then we will be redirected to the third leaf node at the end. Here DBMS will perform sequential search to find 65. Suppose, instead of 65, we have to search for 60. What will happen in this case? We will not be able to find in the leaf node. No insertions/update/delete is allowed during the search in B+ tree.	Understand	CACS005.23

			
6 ans	Illustrate insertion of an element in B+ trees with example. ## Insertion in B+ Tree **Step 1:** Insert the new node as a leaf node **Step 2:** If the leaf doesn't have required space, split the node and copy the middle node to the next index node. **Step 3:** If the index node doesn't have required space, split the node and copy the middle element to the next index page. ## Example : Insert the value 195 into the B+ tree of order 5 shown in the following figure.  195 will be inserted in the right sub-tree of 120 after 190. Insert it at the desired position.  The node contains greater than the maximum number of elements i.e. 4, therefore split it and place the median node up to the parent.  Now, the index node contains 6 children and 5 keys which violates the B+ tree properties, therefore we need to split it, shown as follows. 	Remember	CACS005.23
7 ans	Illustrate deletion of an element in B+ trees with example. ## Deletion in B+ Tree **Step 1:** Delete the key and data from the leaves.	Understand	CACS005.23

	Step 2: if the leaf node contains less than minimum number of elements, merge down the node with its sibling and delete the key in between them. **Step 3:** if the index node contains less than minimum number of elements, merge the node with the sibling and move down the key in between them. ## Example Delete the key 200 from the B+ Tree shown in the following figure. Delete the key 200 from the B+ Tree shown in the following figure.  200 is present in the right sub-tree of 190, after 195. delete it.  Merge the two nodes by using 195, 190, 154 and 129.  Now, element 120 is the single element present in the node which is violating the B+ Tree properties. Therefore, we need to merge it by using 60, 78, 108 and 120. Now, the height of B+ tree will be decreased by 1.  		
8 ans	Write in detail about Static Hashing. **Static Hashing –** In static hashing, when a search-key value is provided, the hash function always computes the same address. For example, if we want to generate address for STUDENT_ID = 76 using mod (5) hash function, it always result in the same bucket address 4. There will not be any changes to the	Understand	CACS005.24

bucket address here. Hence number of data buckets in the memory for this static hashing remains constant throughout.

Operations –

- **Insertion –** When a new record is inserted into the table, The hash function h generate a bucket address for the new record based on its hash key K.
  Bucket address = h(K)
- **Searching –** When a record needs to be searched, The same hash function is used to retrieve the bucket address for the record. For Example, if we want to retrieve whole record for ID 76, and if the hash function is mod (5) on that ID, the bucket address generated would be 4. Then we will directly got to address 4 and retrieve the whole record for ID 104. Here ID acts as a hash key.
- **Deletion –** If we want to delete a record, Using the hash function we will first fetch the record which is supposed to be deleted. Then we will remove the records for that address in memory.
- **Updation –** The data record that needs to be updated is first searched using hash function, and then the data record is updated.

Now, If we want to insert some new records into the file But the data bucket address generated by the hash function is not empty or the data already exists in that address. This becomes a critical situation to handle. This situation in the static hashing is called **bucket overflow**.

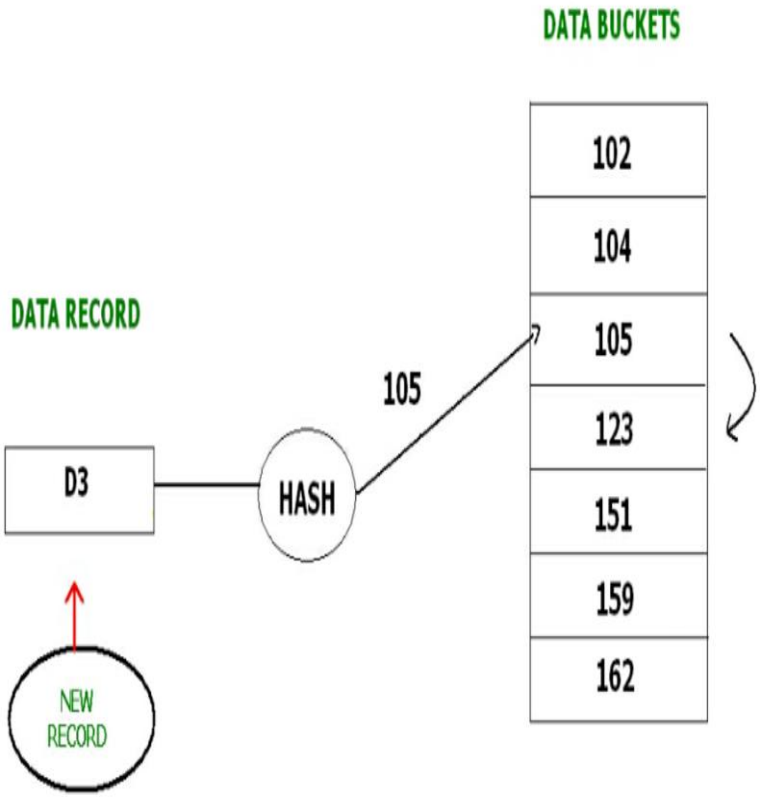
How will we insert data in this case?

There are several methods provided to overcome this situation. Some commonly used methods are discussed below:

3. **Open Hashing –**

In Open hashing method, next available data block is used to enter the new record, instead of overwriting older one. This method is also called linear probing.

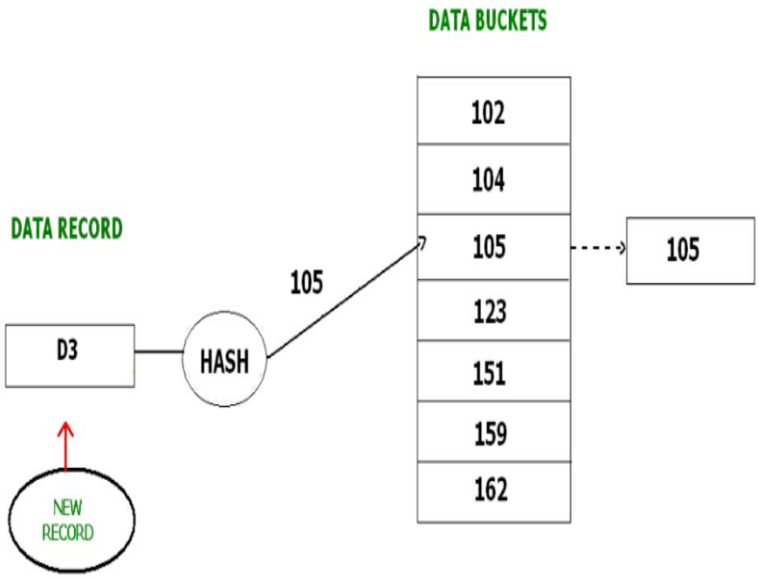
For example, D3 is a new record which needs to be inserted , the hash function generates address as 105. But it is already full. So the system searches next available data bucket, 123 and assigns D3 to it.



4. **Closed hashing –**

In Closed hashing method, a new data bucket is allocated with same address and is linked it after the full data bucket. This method is also known as overflow chaining.

For example, we have to insert a new record D3 into the tables. The static hash function generates the data bucket address as 105. But this bucket is full to store the new data. In this case is a new data bucket is added at the end of 105 data bucket and is linked to it. Then new record D3 is

	inserted into the new bucket.  ○ **Quadratic probing :** Quadratic probing is very much similar to open hashing or linear probing. Here, The only difference between old and new bucket is linear. Quadratic function is used to determine the new bucket address. ○ **Double Hashing :** Double Hashing is another method similar to linear probing. Here the difference is fixed as in linear probing, but this fixed difference is calculated by using another hash function. That's why the name is double hashing.		
9 ans	Explain in detail about Extendible Hashing.	Remember	CACS005.24

	• It is an approach that tries to make hashing **dynamic**, i.e. to allow insertions and deletions to occur without resulting in poor performance after many of these operations. Why this is not the case for ordinary hashing? • Extendible hashing combines two ingredients: **hashing** and **tries**. (tries are digital trees like the one used in Lempel-Ziv) • Keys are placed into buckets, which are independent parts of a file in disk. Keys having a hashing address with the same prefix share the same bucket. A trie is used for fast access to the buckets. It uses a prefix of the hashing address in order to locate the desired bucket. **What makes it extendible?** So far we have not discussed how this approach can be dynamic, making the table expand or shrink as records are added or deleted. The dynamic aspects are handled by two mechanisms: • Insertions and bucket splitting. • Deletions and bucket combination.		
10 ans	Explain in detail about Linear Hashing. ● Full buckets are not necessarily split ● Buckets split are not necessarily full ● Every bucket will be split sooner or later and so all Overflows will be reclaimed and rehashed. ● Split pointer s decides which bucket to split ○ *s* is independent to overflowing bucket ○ At level *i*, *s* is between 0 and 2_i ○ *s* is incremented and if at end, is reset to 0. ● h_i (k)= h(k) mod(2_i n) ● h_{i+1} doubles the range of h_i **Insertion and Overflow condition** Algorithm for inserting 'k': 1. b = h0(k) 2. if b < split-pointer then	Understand	CACS005.24

3. b = h1(k)

Searching in the hash table for ‘k’:

1. b = h0(k)

2. if b < split-pointer then

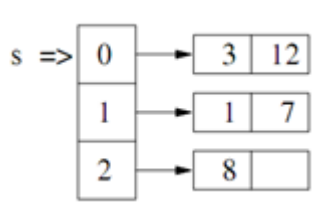
3. b = h1(k)

4. read bucket b and search there

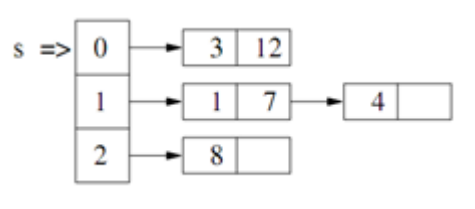
Example:

- In the following M=3 (initial # of buckets)
- Each bucket has 2 keys. One extra key for overflow.
- s is a pointer, pointing to the split location. This is the place where next split should take place.
- Insert Order: 1,7,3,8,12,4,11,2,10,13

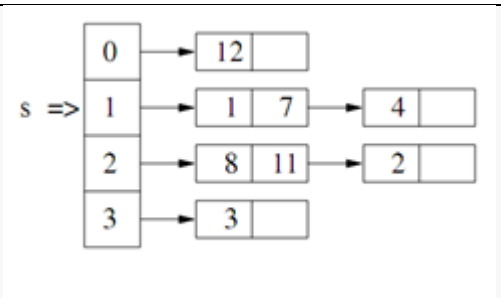
After insertion till 12:



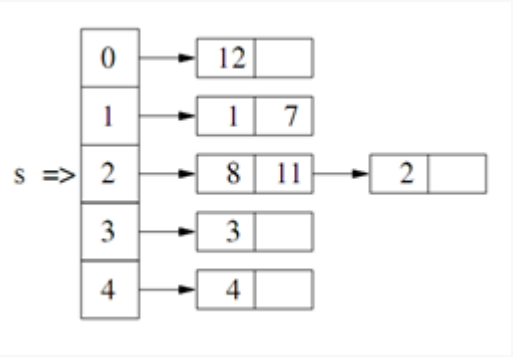
When 4 inserted overflow occurred. So we split the bucket (no matter it is full or partially empty). And increment pointer.



So we split bucket 0 and rehashed all keys in it. Placed 3 to new bucket as (3 mod 6 = 3) and (12 mod 6 = 0). Then 11 and 2 are inserted. And now overflow. *s* is pointing to bucket 1, hence split bucket 1 by re- hashing it.

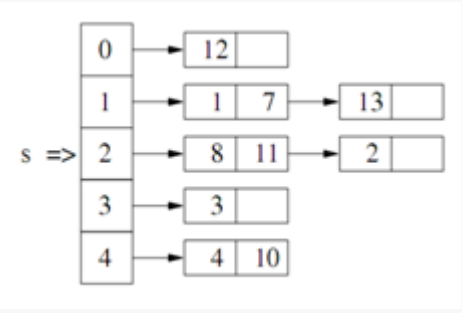


After split:

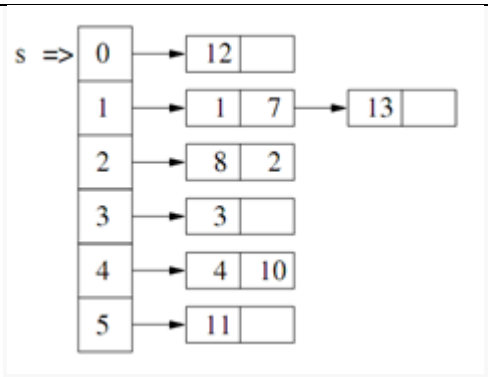


Insertion of 10 and 13: as (10 mod 3 = 1) and bucket 1 < s, we need to hash 10 again using h1(10) = 10 mod 6 = 4th bucket.

When 13 is inserted same process is done, but it end up to the same bucket. But here is an overflow, we need to split 2nd bucket.



Here is the final hash table.



s is moved to the top again as one cycle is completed. Now *s* will travel from 0 to 5th bucket, then 0 to 12, etc;

11 ans	Compare and Contrast Extendible Hashing with Linear Hashing.	Understand	CACS005.24

Extendible Hashing:

It is assumed that the entire directory is kept in main memory. The following notations have been used:

b Bucket Capacity
n Total number of records
ln Logarithmic value with base 2
e Inverse of ln
Some important costs are Insertion Cost = 1 + [1/(b ln 2)]
Search Cost = 1
Directory Size= [e n**(l+l/b)] I (b ln 2)
Storage Utilization = ln 2
Number of Buckets = n / (b ln 2)

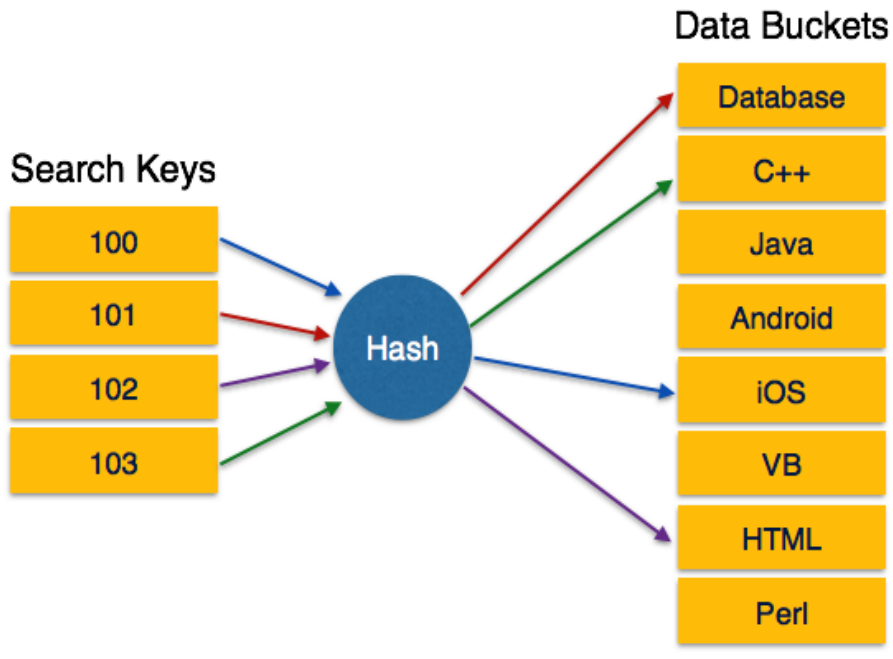
linear hashing:

```
y         Number of records/bucket
b         Primary bucket capacity
c         Secondary bucket capacity
z         Load factor = y/b
P(i,z)    probability that i records hash to a bucket,
          given the load factor z.  This implies binomial
          probability. For infinite number of records
          and buckets, binomial probabilities converge to
          Poisson probabilities [Lar83].  Hence
          P(i,z) = [e**(-zb) * (zb)**i] / i!
k         Number of buckets on a bucket chain, k>=1
j         Number of overflow records in the last overflow
          bucket only if overflow bucket exists
s(z),S(z,x)  Cost for successful search
u(z),U(z,x)  cost for unsuccessful search
a(z),A(z,x)  Cost for insertion
t(z),T(z,x)  Number of slots allocated per bucket
E(z,x)       Cost for expansion
V(z,x)       Overflow space per record
$$           Sign for infinity

     Some important costs are (Sources: [Mul81] and

[Lar85]):

                          $$    c
     s(z) = 1+(1/zb) ∑    k  ∑   [(k-1)c/2+j]P(b+(k-1)c+j,z)
                     k=1     j=1
```

	A linear hash table consists of 2 parts: (1) the buckets that have not yet been split during the current expansion, and (2) the buckets that have been split during current expansion (see Figure 12). Split part and newly allocated part should be considered identical. Split / Unsplit / Newly Allocated Fraction: x/2 / 1-x / x/2 Expected no. of records: z/2 / z / z/2		
12	How does Extendable hashing use a directory of buckets? How does it handles insert and delete operations?	Understand	CACS005.24
13	Explain how insert and delete operations are handled in a static hash index. # Static Hashing In static hashing, when a search-key value is provided, the hash function always computes the same address. For example, if mod-4 hash function is used, then it shall generate only 5 values. The output address shall always be same for that function. The number of buckets provided remains unchanged at all times. 	Understand	CACS005.24

	Operation • Insertion – When a record is required to be entered using static hash, the hash function h computes the bucket address for search key K, where the record will be stored. Bucket address = h(K) • Search – When a record needs to be retrieved, the same hash function can be used to retrieve the address of the bucket where the data is stored. • Delete – This is simply a search followed by a deletion operation		
PART – C (Problem Solving and Critical Thinking Questions)			

1.
Consider a B+-tree in which the maximum number of keys in a node is 5 Calculate the minimum number of keys in any non-root node?
Ans:
order= maximum no of children = maximum number of keys + 1

= 5+1 = 6

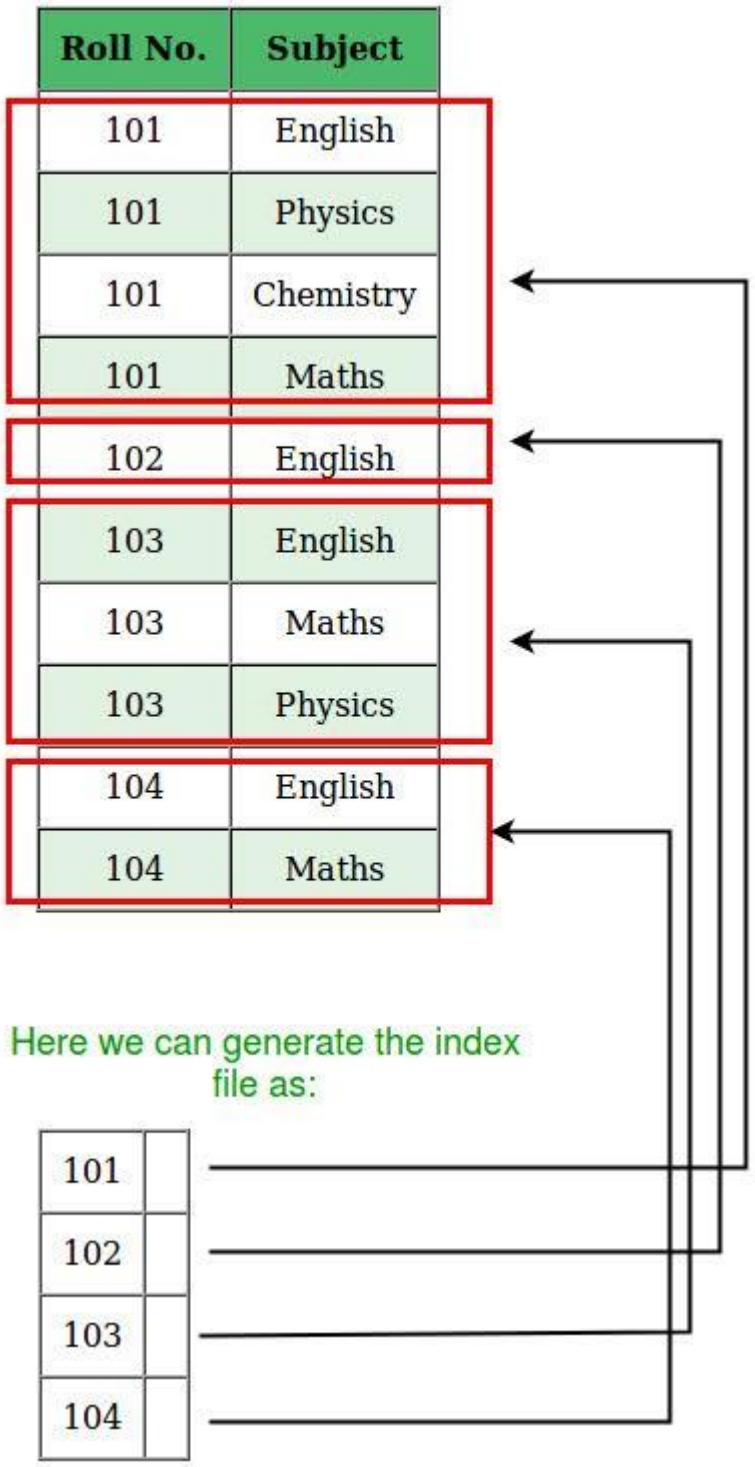
degree, t= order/2 = 6/2= 3

max key = (2t-1) = 2*3-1=5

min key = t-1=3-1=2

2. **In the index allocation scheme of blocks to a file, Calculate on what maximum possible size of the file depends?**
Ans: maximum possible size of the file depends upon **the number of blocks used for the index, and the size of the blocks.**
In the index allocation method, an index block stores the address of all the blocks allocated to a file.

When indexes are created, the maximum number of blocks given to a file depends upon the size of the index which tells how many blocks can be there and size of each block(i.e. same as depending upon the number of blocks for storing the indexes and size of each index block)
3.**A clustering index is defined on the fields of which type? Analyze them**
Ans:
A clustering index as the name suggests is created when the data can be grouped in the form of clusters.



For example: A small database file storing roll no and subjects enrolled for a particular student. Here data can be grouped on the basis of Roll No.s.
Hence to create such kind of index files, fields could be non-key attributes and which are in ordered form so as to form clusters easily.
Hence option (A) – non key and ordering.

Additional points for Clustered Indexing:

1. The number of entries in the index file are the unique values of the attribute on which indexing is done.

2. The pointer in the index file will give the base address of the block in which the value appear for the first time.

4.**Calculate the minimum space utilization for a B+ tree index?**

Ans:

B+ tree of order n have i) n pointers ii) n-1 keys

In B+ tree values are stored only at leaf nodes it is a enhancement over B tree so number of values leaf nodes can stores

Ceil(n/2) to n so n/2 space is empty that's why space utilization is 50%.

5.**Consider the B+ tree index of order d = 2 shown in Figure 10.1. 1.**

Show the tree that would result from inserting a data entry with key 9 into this tree. 2. Show the B+ tree that would result from inserting a data entry with key 3 into the original tree.

How many page reads and page writes does the insertion require?

3. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the left sibling is checked for possible edistribution.

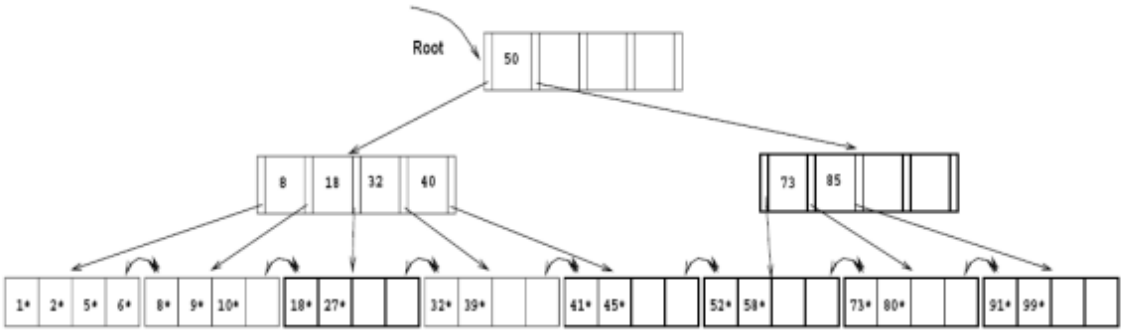
4. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the right sibling is checked for possible redistribution.

5. Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 46 and then deleting the data entry with key 52.

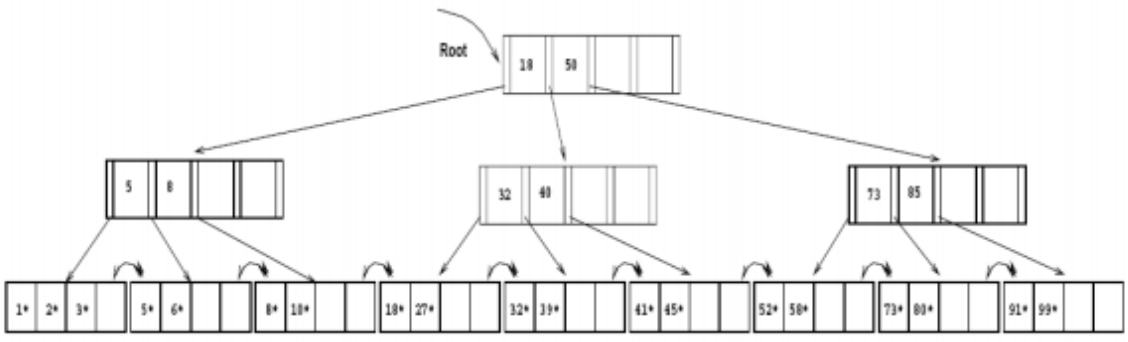
6. Show the B+ tree that would result from deleting the data entry with key 91 from the original

Ans:

a.

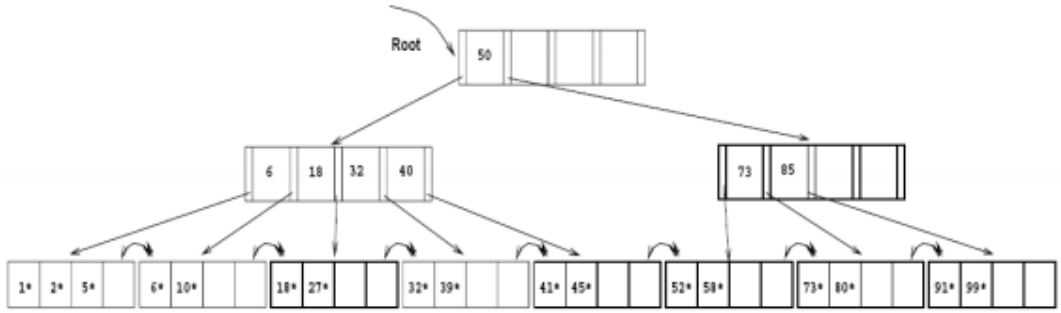


b.

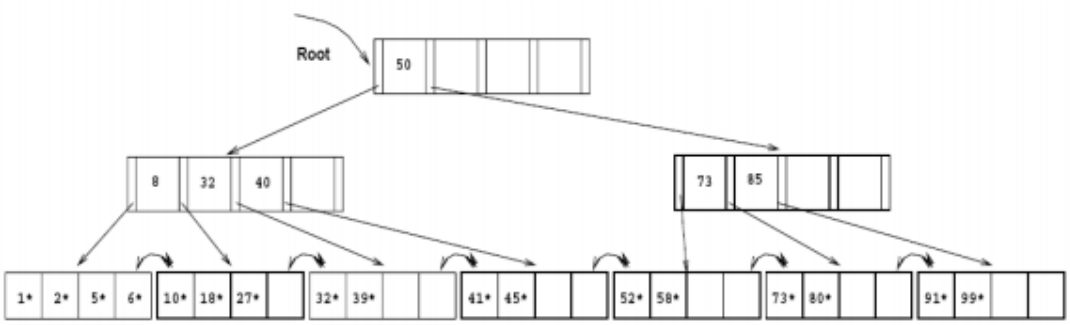


The insertion will require 5 page writes, 4 page reads and allocation of 2 new pages.

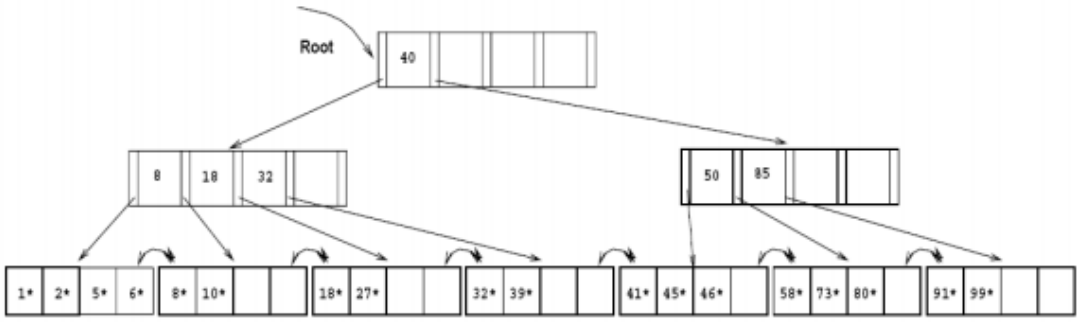
c.



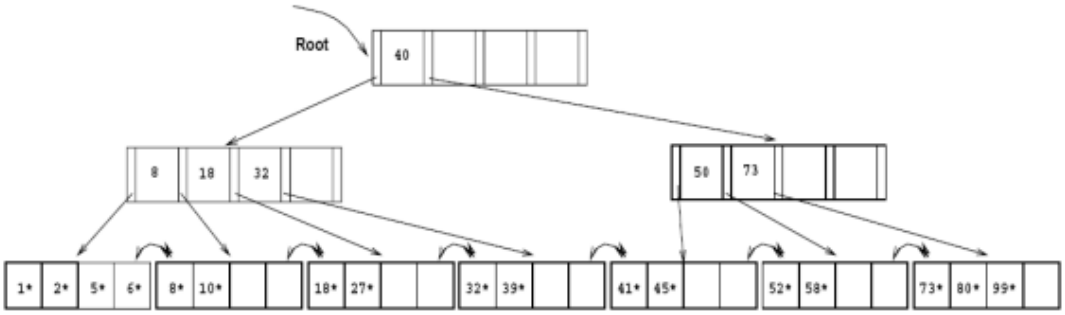
d.



e.



f.



8)

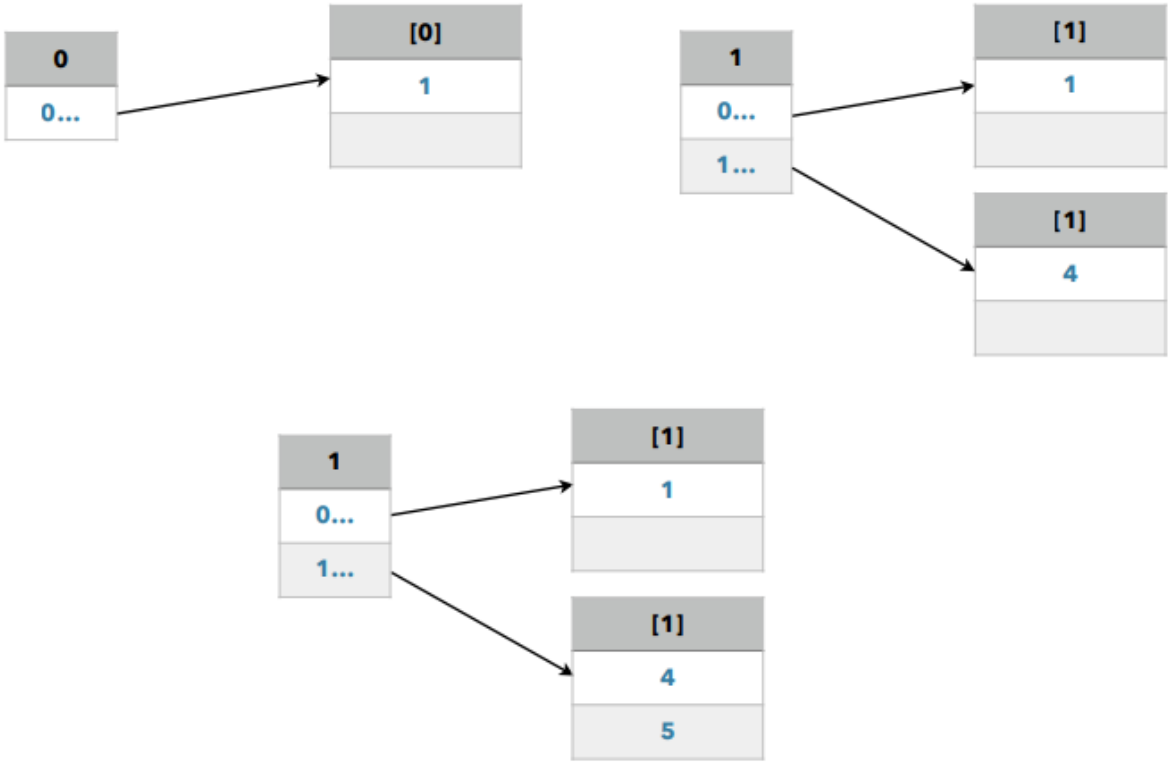
Explain the distinction between closed an open hashing Discuss the relative merits of each technique in database application?

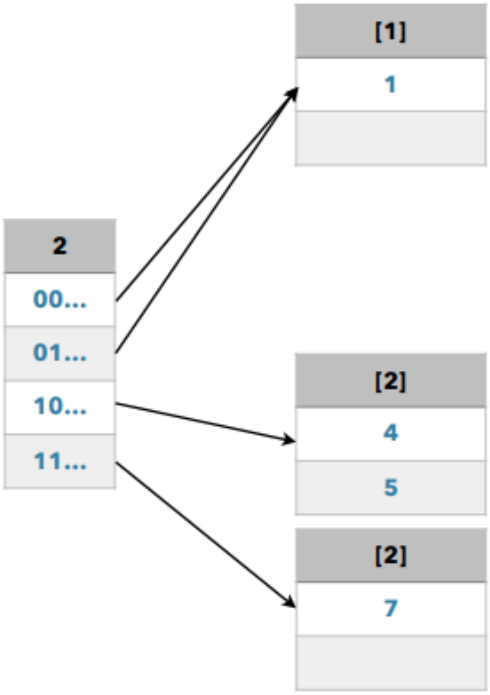
Ans:

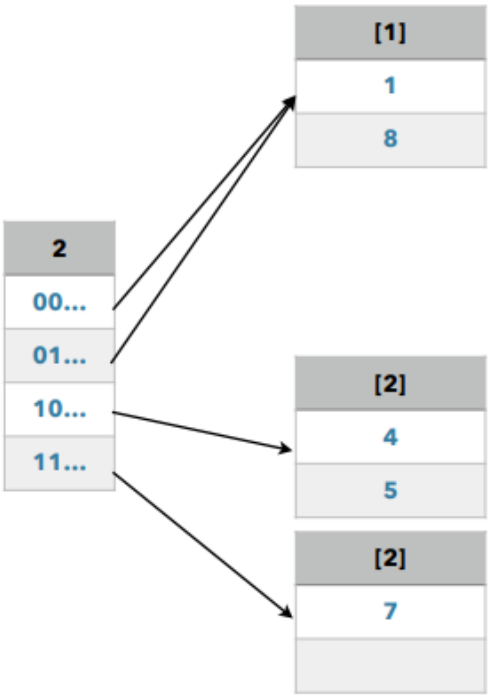
A hash table is where data storage for a key-value pair is done by generating an index using a hash function.

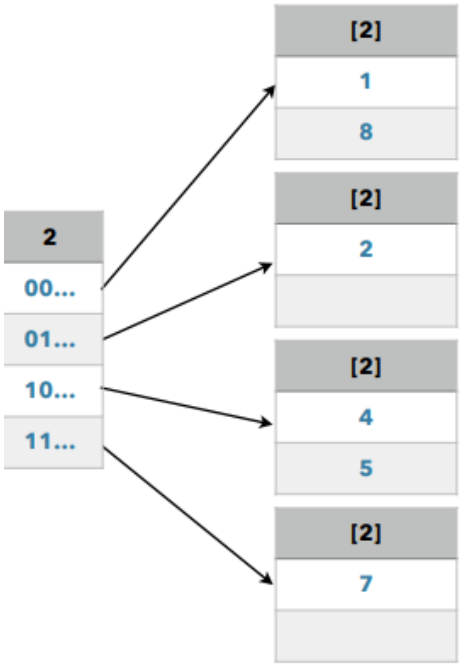
Open Hashing (aka Separate chaining) is simpler to implement, and more efficient for large records or sparse tables.
Closed Hashing (aka Open Addressing) is more complex but can be more efficient, especially for small data records.

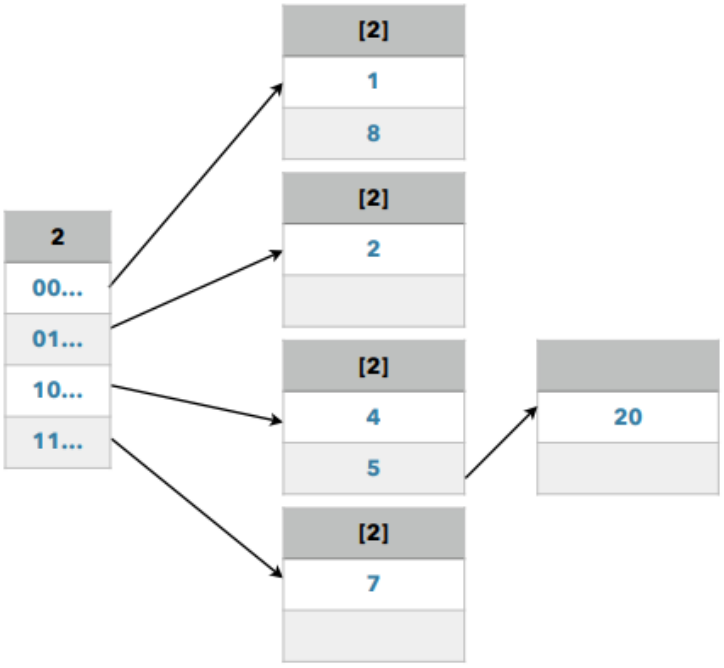
10)
Suppose that we are using extendable hashing on a file that contains records with the following search-key values: 2,3,5,7,11,17,19,23,29,31 Show the extendable hash structure for this file if the hash function is h(x) = x mod 8 and buckets can hold three records.




2
00...
01...
10...
11...
[1]
1
[2]
4
5
[2]
7


2
00...
01...
10...
11...
[1]
1
8
[2]
4
5
[2]
7


2
00...
01...
10...
11...
[2]
1
8
[2]
2
[2]
4
5
[2]
7


2
00...
01...
10...
11...
[2]
1
8
[2]
2
[2]
4
5
[2]
7
20

HOD, CSE