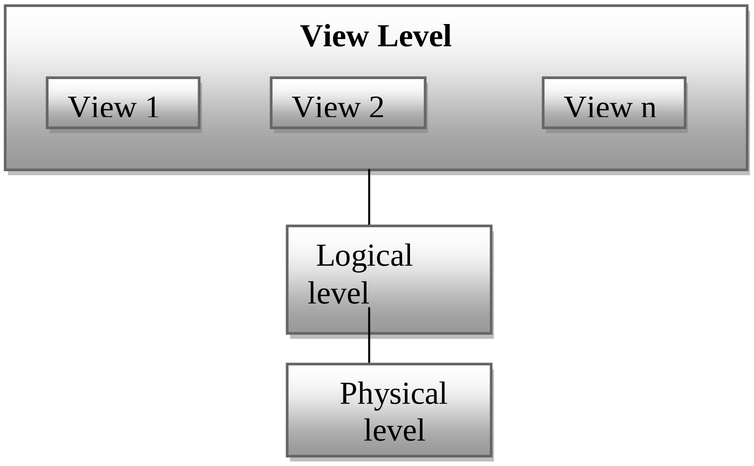
**Q.2a. Explain different levels of data abstraction.**

**Answer:** (a) **Levels of Data Abstraction**
There are three levels of data abstraction.

**Physcial Level** : It describes **how a record** (e.g., customer) is stored.

**Features :**

(a) Lowest level of abstraction.
(b) It describes how data are actually stored.
(c) It describes low-level complex data structures in detail.
(d) At this level, efficient algorithms to access data are defined.



II **Logical level** : It describes **what data stored** in database, and the relationships among the data.

**Features:**

(a) It is next-higher level of abstraction. Here whole Database is divided into small simple structures.
(b) Users at this level need not be aware of the physical-level complexity used to implement the simple structures.
(c) Here the aim is ease of use.
(d) Generally, database administrators (DBAs) work at logical level of abstraction.

I. **View level** : Application programs hide details of data types. Views can also hide information (e.g. salary) for security purposes.

**Features:**

(a) It is the highest level of abstraction.
(b) It describes only a part of the whole Database for particular group of users.
(c) This view hides all complexity.
(d) It exists only to simplify user interaction with system.

The system may provide many views for the whole system

**b.We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Why, then do we have weak entity sets?**

A 2(b) We have weak entities for several reasons:

- 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.

**c.Explain the distinction between total and partial constraints.**

A2(c) In a total design constraint, each higher-level entity must belong to a lower-level entity set. The same need not be true in a partial design constraint. For instance, some employees may belong to no work-team.

**Q.3a. Given the following relations:**

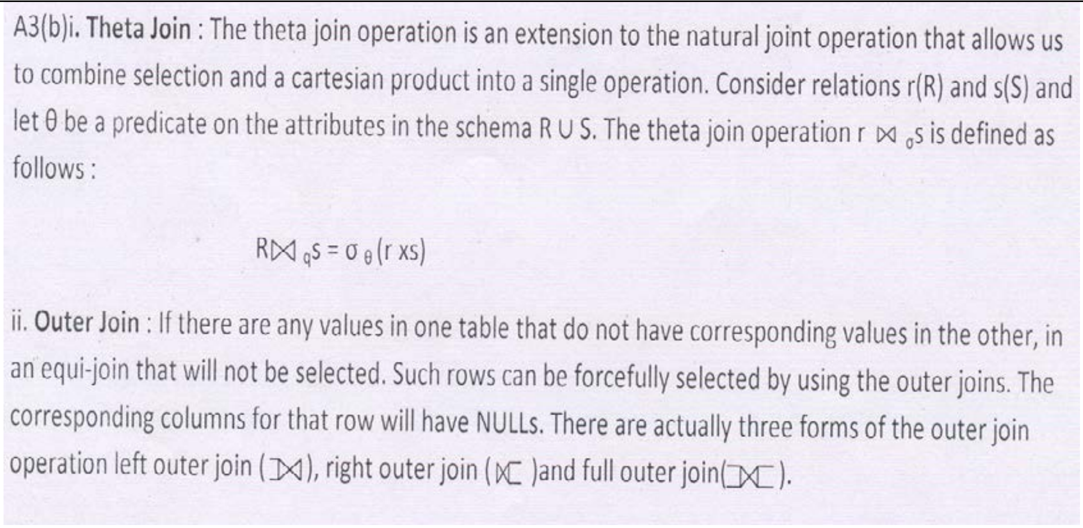
**Vehicle (<u>reg-no</u>, make, colour)**
**Person (<u>eno</u>, name, address)**
**Owner (<u>eno, reg-no</u>)**
**Write expressions in relational algebra to answer the following queries: List the**
**(i) names of persons who do not own any car.**
**(ii) List the names of persons who own only Maruti Cars.**

**Answer:**

(i).$\pi_{name}(PERSON \bowtie \pi_{eno}(PERSON)-\pi_{eno}(OWNER))$

(ii). $\pi_{name}(((\pi_{eno}(OWNER \bowtie \sigma_{make='maruti'}(VEHICLE))$
$\pi_{eno}(OWNER \bowtie \sigma_{make \neq 'maruti'}(VEHICLE)) \bowtie PERSON)$

**b. Define the following:**
**(i) Theta Join**
**(ii) Outer Join**


A3(b)i. **Theta Join** : The theta join operation is an extension to the natural joint operation that allows us to combine selection and a cartesian product into a single operation. Consider relations r(R) and s(S) and let θ be a predicate on the attributes in the schema R ∪ S. The theta join operation $r \bowtie_{\theta} s$ is defined as follows :

$$R \bowtie_{q} s = \sigma_{\theta}(r\ xs)$$

ii. **Outer Join** : If there are any values in one table that do not have corresponding values in the other, in an equi-join that will not be selected. Such rows can be forcefully selected by using the outer joins. The corresponding columns for that row will have NULLs. There are actually three forms of the outer join operation left outer join (⟕), right outer join (⟖ )and full outer join(⟗ ).

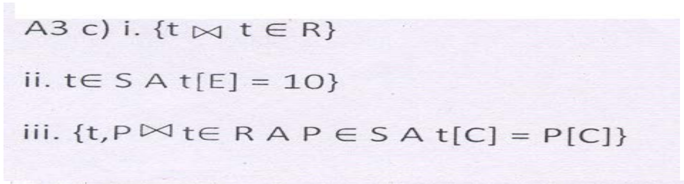
**c. Given the relations R(A,B,C) and S(C,D,E,F) give an expression in tuple relational calculus that is equivalent to each of the following :**

**(i)** $\prod_{A,B,C}(R)$

**(ii)** $\sigma_{E=10}(S)$

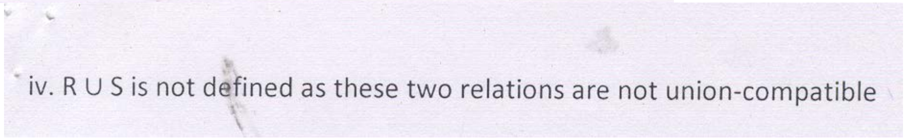
**(iii)** $R \bowtie S$

**(iv)** $R \cup S$

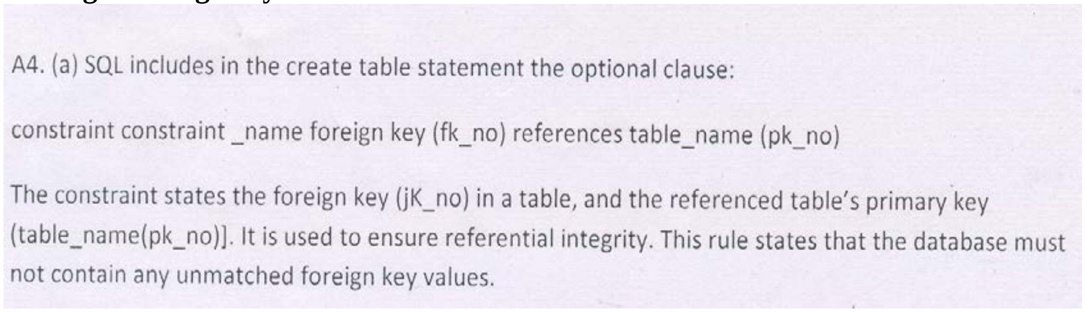

A3 c) i. {t ⋈ t ∈ R}

ii. t∈ S ∧ t[E] = 10}

iii. {t,P ⋈ t∈ R ∧ P ∈ S ∧ t[C] = P[C]}


iv. R ∪ S is not defined as these two relations are not union-compatible

**Q.4a. Explain the purpose of the integrity constraint, in the SQL create table statement, relating to foreign keys.**

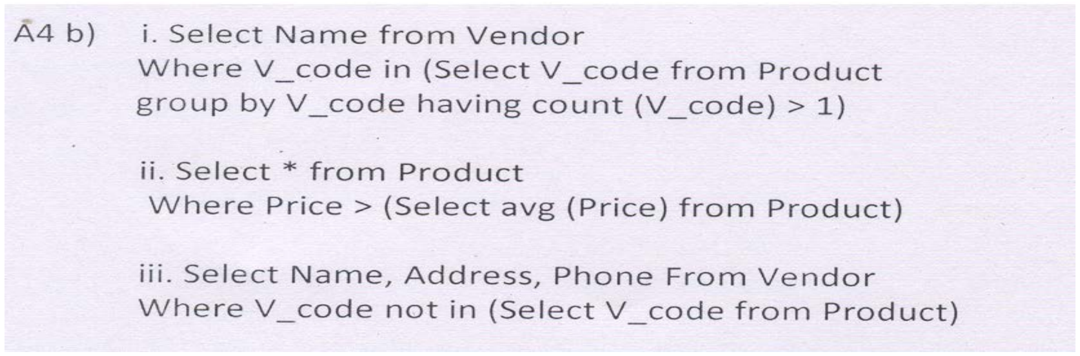

A4. (a) SQL includes in the create table statement the optional clause:

constraint constraint _name foreign key (fk_no) references table_name (pk_no)

The constraint states the foreign key (jK_no) in a table, and the referenced table's primary key (table_name(pk_no)]. It is used to ensure referential integrity. This rule states that the database must not contain any unmatched foreign key values.

**b. Consider the following relations with underlined primary keys. +3)**
**Product (P_code, Description, Stocking_date, QtyOnHand, MinQty, Prices, Discount, V_code)**

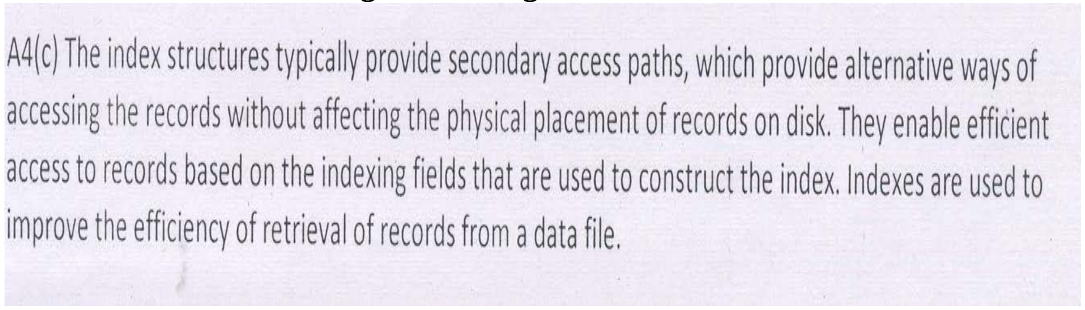
**Vendor (V_code, Name, Address, Phone)**
**Here a vendor can supply more than one product but a product is supplied by only one vendor.**
**Write SQL queries for the following:**
**List the names of all the vendors who supply more than one product.**
**(i) List the details of the products whose prices exceed the average product price.**
**(ii) List the Name, Address and Phone of the vendors who are currently not supplying any product.**


A4 b) i. Select Name from Vendor
Where V_code in (Select V_code from Product
group by V_code having count (V_code) > 1)

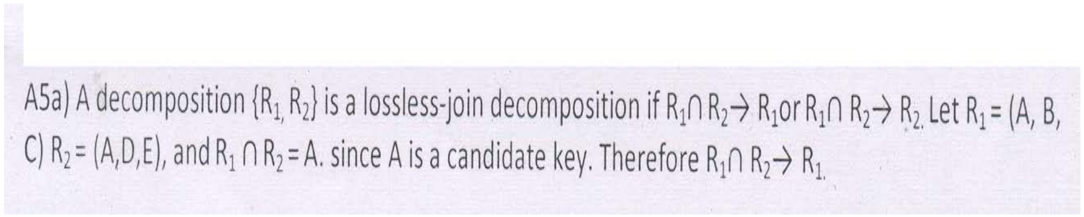
ii. Select * from Product
Where Price > (Select avg (Price) from Product)

iii. Select Name, Address, Phone From Vendor
Where V_code not in (Select V_code from Product)

**c. What are the advantages of having an index structure?**

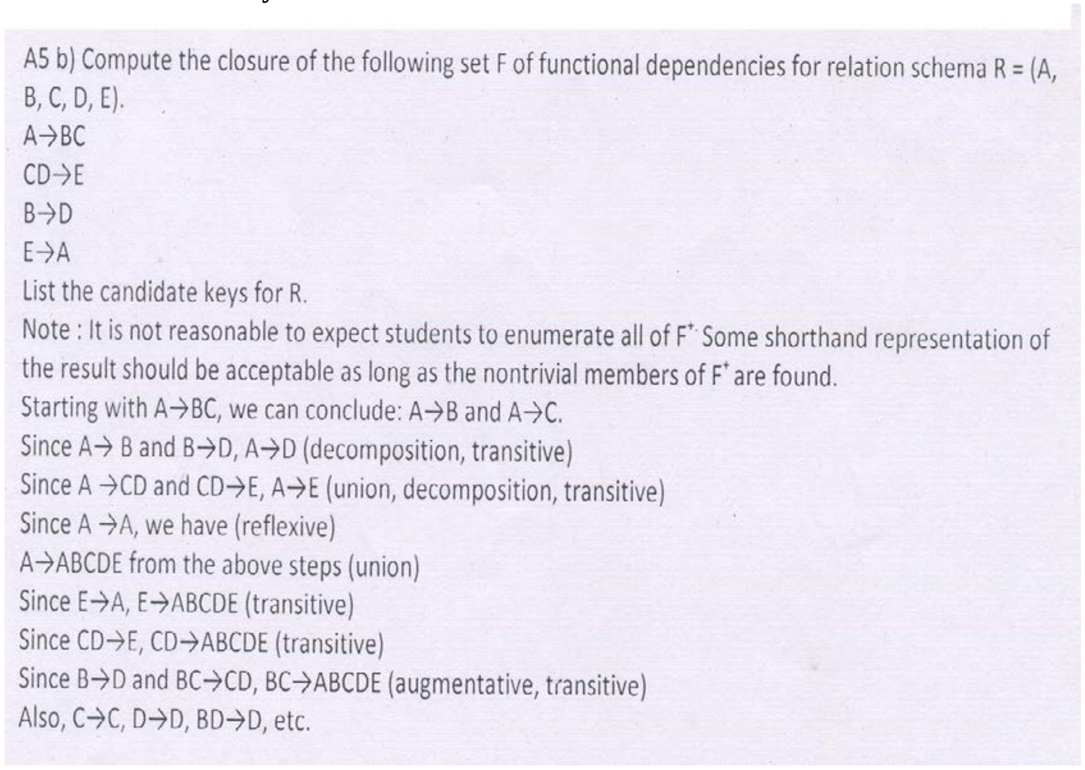

A4(c) The index structures typically provide secondary access paths, which provide alternative ways of accessing the records without affecting the physical placement of records on disk. They enable efficient access to records based on the indexing fields that are used to construct the index. Indexes are used to improve the efficiency of retrieval of records from a data file.

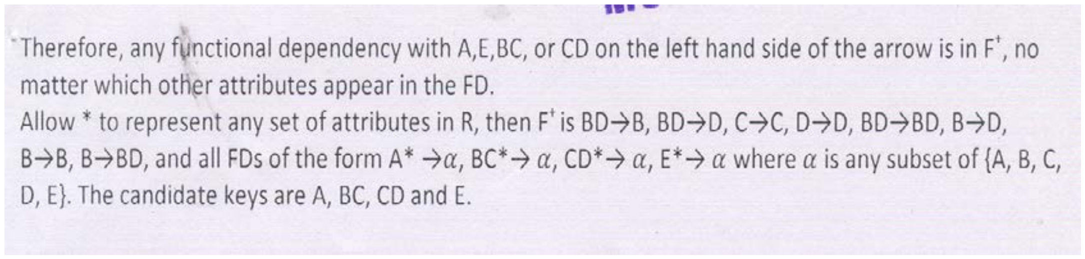
**Q.5 a. Suppose that we decompose the schema R = (A, B, C, D, E) into**
**(A,B,C) (A,D,E).**
**Show that this decomposition is a lossless – join decomposition if the following set F of functional dependencies holds:**
**$A \rightarrow BC$**
**$CD \rightarrow E$**
**$B \rightarrow D$**
**$E \rightarrow A$**


A5a) A decomposition $\{R_1, R_2\}$ is a lossless-join decomposition if $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$. Let $R_1 = (A, B, C)$ $R_2 = (A,D,E)$, and $R_1 \cap R_2 = A$. since A is a candidate key. Therefore $R_1 \cap R_2 \rightarrow R_1$.

**b.Compute the closure of the following set F of functional dependencies for relation schema**
**R = (A, B, C, D, E)**

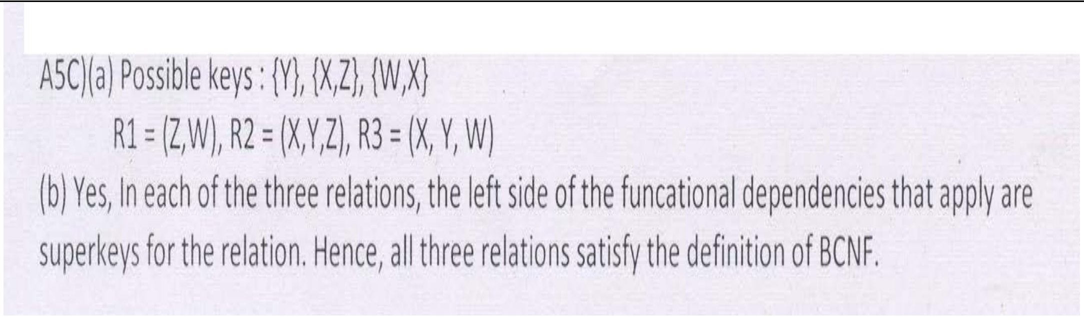
**A→BC**
**CD→E**
**B→D**
**E→A**
**List the candidate keys for R.**


A5 b) Compute the closure of the following set F of functional dependencies for relation schema R = (A, B, C, D, E).
A→BC
CD→E
B→D
E→A
List the candidate keys for R.
Note : It is not reasonable to expect students to enumerate all of $F^+$ Some shorthand representation of the result should be acceptable as long as the nontrivial members of $F^+$ are found.
Starting with A→BC, we can conclude: A→B and A→C.
Since A→ B and B→D, A→D (decomposition, transitive)
Since A →CD and CD→E, A→E (union, decomposition, transitive)
Since A →A, we have (reflexive)
A→ABCDE from the above steps (union)
Since E→A, E→ABCDE (transitive)
Since CD→E, CD→ABCDE (transitive)
Since B→D and BC→CD, BC→ABCDE (augmentative, transitive)
Also, C→C, D→D, BD→D, etc.


Therefore, any functional dependency with A,E,BC, or CD on the left hand side of the arrow is in $F^+$, no matter which other attributes appear in the FD.
Allow * to represent any set of attributes in R, then $F^+$ is BD→B, BD→D, C→C, D→D, BD→BD, B→D, B→B, B→BD, and all FDs of the form A* →$\alpha$, BC*→ $\alpha$, CD*→ $\alpha$, E*→ $\alpha$ where $\alpha$ is any subset of {A, B, C, D, E}. The candidate keys are A, BC, CD and E.

**c.Given relation R(W,X,Y,Z) and set of functional dependencies G = {Z→W, Y→XZ, XW→Y}, where G is a minimal cover:**

**(i) Decompose R into a set of relations in Third Normal Form.**
**(ii) Is your decomposition in part (i) also in Boyce Codd Normal Form? Explain your answer.**


A5C)(a) Possible keys : {Y}, {X,Z}, {W,X}

R1 = (Z,W), R2 = (X,Y,Z), R3 = (X, Y, W)

(b) Yes, In each of the three relations, the left side of the funcational dependencies that apply are superkeys for the relation. Hence, all three relations satisfy the definition of BCNF.

**Q.6a. Define and differentiate between ordered indexing and hashing. Give illustrative examples.**

A6 a)**Ordered indexing** : To gain fast random access to records in a file, we can use an index structure. Each index structure is associated with a particular search key. An ordered indeed stores the values of the search keys in sorted order and associates with each search key records that contain it. The records in the indexed file may themselves be stored in some sorted order. A file may have several indices on different search keys.
**Hashing**: It also provides a way of constructing indices. In hashing, the address of the disk lock containing a desired record directly is computed by a function on the search key value of the record.
**Differentiate between ordered indexing and hashing**: Ordered indexing is stored in sorted order while in Hashing search keys are distributed uniformly across “buckets” using ‘hash function’.

**b.What is indexed sequential file organization? What are the applications of this organization? How are indexes implemented in the case of multiple keys?**

A6 b) Static Hashing has the number of primary pages in the directory fixed. Thus, when a bucket is full, we need an overflow bucket to store any additional records that hash to the full bucket. This can be done with a link to an overflow page, or a linked list of overflow pages. The linked list can be separate for each bucket, or the same for all buckets that overflow. When searching for a record, the original bucket is accessed first, then the overflow buckets. Provided there are many keys that hash to the same bucket, locating a record may require accessing multiple pages on disk, which greatly degrades performance.
The problem of lengthy searching of overflow buckets is solved by Dynamic Hashing. In Dynamic Hashing the size of the directory grows with the number of collisions to accommodate new records and avoid long overflow page chains. Extendible and Linear Hashing are two dynamic hashing techniques.

An index file can be used to effectively overcome the problem of storing and to speed up the key search as well. The simplest indexing structure is the single-level one: a file whose records are pairs key-pointer is the position in the data file of the record with the given key. Only a subset of data records, evenly spaced along the data file, are indexed, so to mark intervals of data records.

A key search then proceeds as follows : the search key is compared with the index ones to find the highest index key points onward, until the search key is matched or until the record pointed by the next index entry is reached. In spite of the double file access (index + data) needed by this kind of search, the decrease in access time with respect to a sequential file is significant.

**Q.7 a. What is meant by heuristic optimization? Discuss the main heuristics that are applied during query optimization.**

A7 a) In heuristic optimization, heuristics are used to reduce the cost of optimization instead of analyzing the number of different plans to find out the optimal plan. For example. A analyzing optimizer would use the rule 'perform selection operation as early as possible' without finding out whether the cost is reduced by this transformation.

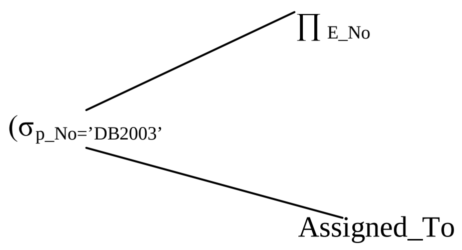
Heuristics approach usually helps to reduce the cost but not always. The heuristics that are applied during query optimization are :

- Pushes the selection and projection operations down the query tree
- Left-deep join trees- convenient for pipelined evaluation
- Non-left-deep join trees

**b. How does a query tree represent a relational algebra expression? Discuss any three rules for query optimization, giving example as to when should each rule be applied.**

A7 b) A query tree is a tree structure that corresponds to a relational algebra expression. It represents the input relations as leaf nodes of the tree, and represents the relational algebra operations as internal nodes. An execution of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation. The execution terminates when the root node is executed and produces the result relation for the query. For example, the query tree for the relational algebra expression

$\prod_{E\_No}$=($\sigma_{p\_No='DB2003'}$ (Assigned_To):



The tree rules for query optimization are as follows:

- In a cascade (sequence) of P operations, all but the last one can be ignored:
  $\prod_1(\prod_2 c\ldots\prod_n)))\equiv \prod_1(R)$
- If a selection condition c involves only those attributes $A_1$, $A_2$……$A_n$ in the projection list, the two operations can be commuted:
  $\prod_{A1, A2, \ldots\ldots An}(sc(R)\equiv sc(R)\equiv sc(\prod_{A1, A2, \ldots\ldots An}(R))\ldots))$
- A conjuctive selection condition can be broken up into a cascade of individual s operations:
  $\sigma_{C1 \text{ AND } C2 \text{ AND}\ldots\text{AND} Cn}(R)\equiv\sigma_{C1}(\sigma_{C2}(\ldots(\sigma_{Cn}(R)\ldots))$

**Q.8 a. Describe each of the following locking protocols: (5)**
**(i) 2PL**
**(ii) Strict 2PL**

A8 a) 2PL: a way DBMS to ensure only serializable schedules are allowed. The rules are:
Each transaction must get an S-lock on an object before reading it;
Each transaction must get an X-lock on an object before writing it;
Once a transaction releases a lock, it can not acquire any new locks.
Strict 2PL: besides all the rules mentioned above, there is one additional rule to ensure only 'safe' interleavings of transactions are allowed, which is: All locks held by a transaction are released when the transaction is completed.

**b.What is a deadlock and how do most DBMS handle deadlocks?**

A8 b) A deadlock is a problem of mutual waiting. One transaction has a resource that another transaction needs, and a second transaction holds a resource that the first transaction needs. To control deadlocks, most DBMS use a time-out policy. The concurrency control manager aborts (with a ROLLBACK statement) any transaction waiting for more than a specified time.

**c.How does the two phase protocol ensure serializability in database schedules?**

A8c) A transaction is said to follow the two-phase locking protocol if all locking operations precede the first unlock operation in the transaction. Such transaction can be divided into two phases: and expanding or growing (first) phase and a shrinking (second) phase. The two-phase locking protocol ensures serializability in database schedules. Consider any transaction. The point in the schedule where the transaction has obtained its final lock (the end of its growing phase) is called the lock point of the transaction. Now, transactions can be ordered according to their lock points – this ordering is, in fact, a serializability ordering for the transactions.

**Q.9 a. Compare shadow paging with log based recovery methods.**

A9 a) **Shadow paging** – In the shadow page scheme, the database is considered to be made up of logical units of storage called pages. The shadow paging scheme uses two page tables for transaction that is going to modify the database. The original page table is called the shadow page table and the transaction addressed the database using another page table known as the current page table. In case of a system crash, before the transaction commits, the shadow page table and the corresponding blocks containing the old database, which was assumed to be in a consistent state, will continue to be accessible. The recovery from system crash is relatively inexpensive, is an advantage of this scheme. The main disadvantages of the shadow scheme are: (1) the problem of scattering, and (2) the original version of the changed blocks pointed to by the shadow page table have to be returned to the pool of free blocks.
**Log Based Recovery Method**: The system log, which is usually written on stable storage, contains the redundant data required to recover from volatile storage failures and also from errors discovered by the transaction or the database system. System log is also called as log and has sometimes been called the DBMS journal. In case of system crash the log is processed from the beginning to the point where

system crashes. All the transaction, those have been recorded their-of-transaction marker, will be committed otherwise rolled back. The simplicity and easier to use the main advantages of this method. The disadvantage are : (1) inefficient for the application, which have large volume of information, (2) in case of system crash with loss of volatile information, the log information collected in buffers will also be lost and transaction that had been completed for some period to the system crash may be missing their respective end-of-transaction markers in the log; if such transactions are rolled back then likely to be partially undone.

**b.Define check point and its impact on data base recovery.**

A9 b) In a large on-line database system there could be hundreds of transactions handled per minute. The log for this type of database contains a very large volume of information. A scheme called checkpoint is used to limit the volume of log information that has to be handled and processed in the event of a system failure involving the loss of volatile information. The checkpoint scheme is an additional component of the logging scheme. A checkpoint operation, performed periodically, copies log information onto stable storage. For all transactions active at checkpoint, their identifiers and their database modification actions, which at that time are reflected only in the database buffers, will be propagated to the appropriate storage.

## TEXTBOOK

**I. Fundamentals of Database Systems, Elmasri, Navathe, Somayajulu, Gupta, Pearson Education, 2006 (TB-I)**