Explain block nested loop join implementation for natural join operation of relational algebra and compute the cost of its implementation.

Block Nested-Loop Join is a variant of nested-loop join in which each block of the inner relation is paired with each block of the outer relation. Within each pair of blocks, the block nested-loop join pairs each tuple of one block with each tuple in the other block to produce all pairs of tuples. It pairs only those tuples that satisfy the given join condition and them to the result.

```
for each block $b_r$ of r do begin
        for each block $b_s$ of s do begin
          for each tuple $t_r$ in $b_r$ do begin
            for each tuple $t_s$ in $b_s$ do begin
              test pair ($t_r$, $t_s$) to determine if they pass the given join condition
               if test passed
              add $t_r$ . $t_s$ to the result;
        end
      end
    end
  end
```

Cost Analysis of Block Nested-Loop Join Algorithm

In the worst case, each block in the inner relation s is read only for one time for each block in the outer relation r. Thus,

Total number of block transfers in worst case = $b_r * b_s + b_r$

Total number of seeks required = $2 * b_r$

Here, b_r and b_s are the number of blocks holding records of the given relation r and s, respectively.

In the best case, the inner relation fits entirely into memory. Thus,

Total number of block transfers in best case = $b_r + b_s$

Total number of seeks required = $2(n_r + b_r)$

Write any two differences between object relational model and object model in a database system.

Feature	Object-Relational Model (ORM)	Object Model
Data Representation	Data is represented in relational format using tables, rows, and columns.	Data is represented using objects, encapsulating both data and behaviour.
Relationship Handling	Relationships between entities are established using foreign key constraints.	Relationships between objects are established using references or pointers.
Data Manipulation	Data manipulation is done using SQL queries	Data manipulation is done using methods associated with objects
Extension	Requires alteration of underlying schema.	Done by adding new classes and methods.
Complexity	Less complex	More complex
Performance	Less efficient	More efficient

Explain various mining techniques such as Prediction, Classification clustering, Co-relation, Association in brief.

- **Prediction:** Prediction, also known as regression analysis, involves analyzing historical data to identify trends and patterns to make predictions about future outcomes. Common algorithms used for prediction include decision trees, linear regression and neural networks.
- **Classification:** Classification is a supervised learning technique that involves categorizing data into predefined classes or categories based on input features. Popular classification algorithms include decision trees, logistic regression, support vector machines (SVM), and naive Bayes.
- **Clustering:** Clustering is an unsupervised learning technique that involves grouping similar data points together into clusters or segments based on their characteristics. Clustering algorithms include k-means, hierarchical clustering, and DBSCAN.
- **Correlation:** Correlation analysis involves identifying relationships between variables in a dataset. It measures the strength and direction of the linear relationship between two variables. Correlation analysis helps in understanding how changes in one variable affect another variable.
- **Association:** Association analysis, also known as market basket analysis, involves discovering associations between items in a dataset. Association rules are used to find hidden patterns, such as "if A, then B" relationships. Popular algorithms for association analysis include Apriori and FP-growth.

Write an algorithm to compute closure of an attribute.

```
Algorithm to compute a⁺, the closure of a under F


Result:= a;
while (changes to Result) do
    for each B → Y in F do
        Begin
            if B ⊆  Result  then  Result := Result ∪  Y
        End
```

What do you mean by loss less joint decomposition of a relation?

To check for lossless join decomposition using the FD set, the following conditions must hold:

1. The Union of Attributes of R1 and R2 must be equal to the attribute of R. Each attribute of R must be either in R1 or in R2.

Att(R1) U Att(R2) = Att(R)

2. The intersection of Attributes of R1 and R2 must not be NULL.

Att(R1) ∩ Att(R2) ≠ Φ

3. The common attribute must be a key for at least one relation (R1 or R2)

Att(R1) ∩ Att(R2) -> Att(R1) or Att(R1) ∩ Att(R2) -> Att(R2)

For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is a lossless join decomposition as:

1. First condition holds true as Att(R1) U Att(R2) = (ABC) U (AD) = (ABCD) = Att(R).
2. Second condition holds true as Att(R1) ∩ Att(R2) = (ABC) ∩ (AD) ≠ Φ
3. The third condition holds as Att(R1) ∩ Att(R2) = A is a key of R1(ABC) because A->BC is given.

Suppose a relation scheme R (ABC D) is decomposed in to R1 (A,B,C) and R2 (A, D) with valid functional dependencies A→BC and A→D in R1 and R2 respectively. How will you define attribute A in relation R?

Given the functional dependencies, we can determine the candidate key for R.

Closure of A:

- $A^+=\{A\}$
- Using A→BC, we get $A^+ = \{A,B,C\}$
- Using A→D, we get $A^+ = \{A,B,C,D\}$

Since the closure of A includes all attributes of R, A is a superkey. Furthermore, since A alone determines all other attributes and no subset of A can determine the whole set of attributes, A is the candidate key of R.

Decomposition and Preservation

When we decompose R into R_1 (A,B,C) and R_2 (A,D):

- R_1 preserves the dependency $A \rightarrow BC$.
- R_2 preserves the dependency $A \rightarrow D$.

Both decomposed relations R_1 and R_2 maintain the functional dependencies of the original relation, and they both include A as a key.

Conclusion: Attribute A in relation R can be defined as the candidate key.

How does 3NF differ from BCNF? Further, through an example prove that BCNF is stronger than 3NF.

A relation is in 3NF if:

- It is in 2NF.
- It has no transitive dependencies.

A relation is in BCNF if:

- It is in 3NF, and
- For every functional dependency $X \rightarrow Y$, X is a superkey.

Suppose a relational schema R (A B C) and set of functional dependency

F: $AB \rightarrow C$

$C \rightarrow B$

There are two candidate key in above table i.e. AB and AC.

$(AB)^+ = ABC$

$(AC)^+ = ABC$

Closure of AB has all the attributes of R, and closure of AC also has all the attributes of R.
Prime attributes: A B C

Non prime attributes: NIL

The table is in 2NF because there is no partial dependency.

The table is in 3NF because there is no transitive dependency.

But $C \rightarrow B$ functional dependency is not following rule of BCNF because C is not candidate key

Hence BCNF is stronger than 3NF.

Explain multivalued dependency and its role in forth normal form. Give a schema which satisfies all normal form based on functional dependency but still has anomalies.

A **multivalued dependency (MVD)** occurs in a relation when one attribute determines a set of values for another attribute independently of other attributes. Formally, for a relation R with attributes A, B, and C, an MVD A→→B indicates that the set of B values associated with a given A is independent of the set of C values associated with the same A.

Fourth Normal Form (4NF)

A relation is in Fourth Normal Form (4NF) if it is in Boyce-Codd Normal Form (BCNF) and there are no non-trivial multivalued dependencies other than a candidate key. This means:

- The relation must be in BCNF.
- For every non-trivial MVD X→→Y, X must be a superkey.

Example to Illustrate 4NF and Anomalies

Consider a relation R that satisfies BCNF but still has anomalies due to multivalued dependencies.

Schema Example

Let's take a relation schema R with attributes: R(Student,Course,Hobby)

Assume the following functional dependencies (FDs):

- Student→→Course (A student can enroll in multiple courses independently of their hobbies)
- Student→→Hobby (A student can have multiple hobbies independently of their courses)

This schema satisfies 1NF, 2NF, 3NF, and BCNF because there are no FDs that violate these normal forms. However, it does not satisfy 4NF due to the MVDs.

Anomalies in the Schema

The relation R might look like this:

Student	**Course**	**Hobby**
Alice	Math	Painting
Alice	Math	Dancing
Alice	Science	Painting
Alice	Science	Dancing

The above table shows redundancy because:

- Alice's courses (Math, Science) are listed multiple times for each hobby.
- Alice's hobbies (Painting, Dancing) are listed multiple times for each course.

This redundancy can lead to following anomalies:

- **Insert Anomaly:** To add a new course for Alice, you need to insert multiple tuples, one for each hobby.
- **Delete Anomaly:** Deleting a course could accidentally remove a hobby if tuples are not carefully managed.
- **Update Anomaly:** Updating Alice's hobby requires changes in multiple tuples.

Decomposing to Achieve 4NF

To resolve these anomalies, we decompose R into two relations:

- R1(Student,Course)
- R2(Student,Hobby)

This decomposition results in:

R1:

Student	**Course**
Alice	Math
Alice	Science

R2:

Student	**Hobby**
Alice	Painting
Alice	Dancing

This decomposition removes the redundancy and satisfies 4NF because:

- R1 has no multivalued dependencies other than the candidate key Student.
- R2 has no multivalued dependencies other than the candidate key Student.

Explain various concurrency control protocols in distributed database.

Concurrency control ensures the simultaneous execution of data by several users without causing data inconsistency.

1. Two-Phase Locking Protocol: A transaction in the Two Phase Locking Protocol can assume one of the 2 phases:

1. Expanding or growing phase: during which new locks on items can be acquired but none can be released;
2. Shrinking phase: during which existing locks can be released but no new locks can be acquired.

If lock conversion is allowed, then upgrading of locks (from read-locked to write-locked) must be done during the expanding phase, and downgrading of locks (from write-locked to read-locked) must be done in the shrinking phase.

2. Time Stamp Ordering Protocol: In this method, several versions X_1, X_2, ..., X_k of each data item X are maintained. For each version, the value of version X_i and the following two timestamps are kept:

1. **read_TS(X_i):** The read timestamp of X_i is the largest of all the timestamps of transactions that have successfully read version X_i .
2. **write_TS(X_i):** The write timestamp of Xi is the timestamp of the transaction that wrote the value of version X_i .

Whenever a transaction T is allowed to execute a write_item(X) operation, a new version X_{k+1} of item X is created, with both the write_TS(X_{k+1}) and the read_TS(X_{k+1}) set to TS(T). Correspondingly, when a transaction T is allowed to read the value of version X_i, the value of read_TS(X_i) is set to the larger of the current read_TS(X_i) and TS(T).

3. Multiversion 2 phase locking: In this multiple-mode locking scheme, there are three locking modes for an item: read, write, and certify. The idea behind multiversion 2PL is to allow other transactions T to read an item X while a single transaction T holds a write lock on X. This is accomplished by allowing two versions for each item X:

1. One version must always have been written by some committed transaction.
2. The second version X is created when a transaction T acquires a write lock on the item.

Other transactions can continue to read the committed version of X while T holds the write lock. Transaction T can write the value of X as needed, without affecting the value of the committed version X. However, once T is ready to commit, it must obtain a certify lock on all items that it currently holds write locks on before it can commit. The certify lock is not compatible with read locks, so the transaction may have to delay its commit until all its write-locked items are released by any reading transactions in order to obtain the certify locks. Once the certify locks are acquired, the committed version X of the data item is set to the value of version X, version X is discarded, and the certify locks are then released.

4. Validation Concurrency Control: In optimistic concurrency control techniques, also known as validation or certification techniques, no checking is done while the transaction is executing. There are three phases for this concurrency control protocol:

1. **Read phase:** A transaction can read values of committed data items from the database. However, updates are applied only to local copies (versions) of the data items kept in the transaction workspace.
2. **Validation phase:** Checking is performed to ensure that serializability will not be violated if the transaction updates are applied to the database.
3. **Write phase:** If the validation phase is successful, the transaction updates are applied to the database; otherwise, the updates are discarded and the trans action is restarted.

What do you mean by ACID properties of a transaction?

The following are the ACID properties:

- **Atomicity:** A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.
- **Consistency preservation:** A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.
- **Isolation:** The execution of a transaction should not be interfered with any other transactions executing concurrently.
- **Durability:** The changes applied to the database by a com mitted transaction must persist in the database. These changes must not be lost because of any failure.

Explain Multilevel transactions?

Multilevel transactions extend the concept of nested transactions by allowing transactions to be nested to multiple levels. In a multilevel transaction model, transactions can have multiple levels of nesting, with each level representing a hierarchy of transactions. Similar to nested transactions, each level can have its own set of operations and changes to the database, and it can commit or abort independently of its parent or child transactions. Multilevel transactions provide even greater flexibility in organizing and managing transactional operations, allowing for more complex transactional structures.

Describe the terms support and confidence with the help of suitable examples.

Support refers to the relative frequency of an item set in a dataset. The support of an itemset can be calculated using the following formula:

Support(X) = (Number of transactions containing X) / (Total number of transactions)

where X is the itemset for which you are calculating the support.

For example, suppose we have a dataset of 1000 transactions, and the itemset {milk, bread} appears in 100 of those transactions. The support of the itemset {milk, bread} would be calculated as follows:

Support({milk, bread}) = Number of transactions containing

{milk, bread} / Total number of transactions

= 100 / 1000

= 10%

So the support of the itemset {milk, bread} is 10%. This means that in 10% of the transactions, the items milk and bread were both purchased.

Confidence is a measure of the likelihood that an itemset will appear if another itemset appears. The confidence of a rule can be calculated using the following formula:

Confidence(X => Y) = (Number of transactions containing X and Y) / (Number of transactions containing X)

where X and Y are the itemsets for which you are calculating the confidence of the rule X => Y (meaning “If X, then Y”).

For example, suppose we have a dataset of 1000 transactions, and the itemset {milk, bread} appears in 100 of those transactions. The itemset {milk} appears in 200 of those transactions. The confidence of the rule “If a customer buys milk, they will also buy bread” would be calculated as follows:

```
Confidence("If a customer buys milk, they will also buy bread")
 = Number of transactions containing
  {milk, bread} / Number of transactions containing {milk}
 = 100 / 200
 = 50%
```

So the confidence of the rule “If a customer buys milk, they will also buy bread” is 50%. This means that in 50% of the transactions where milk was purchased, bread was also purchased.

k-means algorithm for clustering

```
Begin
    randomly choose k records as the centroids for the k clusters;
    repeat
    assign each record, ri , to a cluster such that the distance between ri and the cluster centroid (mean) is the smallest among the k clusters;
    recalculate the centroid (mean) for each cluster based on the records assigned to the cluster;
    until no change;
End;
```