**CS3492 dms 16 mark important Questions**

**UNIT-1**

**1. What is Normalization? Explain all Normal Forms**

**Normalization**

Normalization is a **systematic process of organizing data in a database** to reduce redundancy and ensure data integrity. It involves dividing a database into two or more tables and defining relationships between the tables.

**Objectives:**

* Remove data redundancy
* Avoid update anomalies
* Ensure logical data storage

**Normal Forms:**

**1. First Normal Form (1NF):**

* A relation is in 1NF if:
  + It contains **only atomic (indivisible)** values.
  + No repeating groups or arrays.
* **Example:**

| **Roll** | **Name** | **Subjects** |
| --- | --- | --- |
| 101 | Arjun | Math, Science |

* 👉 Not 1NF due to multivalued field.  
  👉 Convert to:

| **Roll** | **Name** | **Subject** |
| --- | --- | --- |
| 101 | Arjun | Math |
| 101 | Arjun | Science |

**2. Second Normal Form (2NF):**

* Relation must be in **1NF** and have **no partial dependency**.
* Partial dependency: Non-prime attribute depends only on part of a composite key.
* Solved by decomposing the relation.

**3. Third Normal Form (3NF):**

* Relation must be in **2NF**, and **no transitive dependency** should exist.
* Transitive dependency: A → B, B → C ⇒ A → C

**4. Boyce-Codd Normal Form (BCNF):**

* Stronger version of 3NF.
* For every FD **X → Y**, X should be a **superkey**.
* Eliminates anomalies not covered by 3NF.

**5. Fourth Normal Form (4NF):**

* In 4NF if it is in BCNF and has **no multivalued dependencies**.

**6. Fifth Normal Form (5NF):**

* Deals with **join dependency**.
* If a relation can’t be decomposed further without loss, it's in 5NF.

**2. Write About Decomposition Preservation Algorithm for All FDs**

**Decomposition** is the process of breaking down a relation into two or more sub-relations.

**Decomposition Should Satisfy:**

1. **Lossless Join Property**
2. **Dependency Preservation Property**

**Dependency Preservation Algorithm:**

**Step 1:** Decompose the original relation R into R1, R2, ..., Rn  
**Step 2:** Let **F** be the set of original FDs, and **Fi** be FDs on each Ri  
**Step 3:** Combine all Fi to form **F’ = F1 ∪ F2 ∪ ... ∪ Fn**

Now check:

* If **F+ = F’+** (i.e., closure of both is same), then dependency is preserved.

**Importance:**

* If not preserved, some constraints may need to be **enforced through joins**, which affects performance.

**3. Explain Functional Dependency Concepts**

**Functional Dependency (FD)**

A **functional dependency** occurs when the value of one attribute determines the value of another attribute.

**Notation:**  
If A → B, then B is functionally dependent on A.

**Types of Functional Dependencies:**

1. **Trivial FD**: A → A or A → part of A
2. **Non-trivial FD**: A → B, where B is not a subset of A
3. **Fully Functional Dependency**: A → B, and B is not dependent on part of A
4. **Transitive Dependency**: A → B, B → C, then A → C

**Closure of FDs:**

Set of all FDs that can be derived from a given FD set using **Armstrong’s Axioms**:

* **Reflexivity**: If Y ⊆ X, then X → Y
* **Augmentation**: If X → Y, then XZ → YZ
* **Transitivity**: If X → Y and Y → Z, then X → Z

**4. Explain 2NF and 3NF in Detail**

**Second Normal Form (2NF):**

* A relation is in 2NF if:
  1. It is in **1NF**
  2. There is **no partial dependency** on a composite primary key

**Example (Not 2NF):**

| **RollNo** | **Subject** | **Name** |
| --- | --- | --- |
| 101 | Math | Arjun |
| 101 | Science | Arjun |

* Partial dependency: RollNo → Name

👉 Solution: Decompose

* R1(RollNo, Name)
* R2(RollNo, Subject)

**Third Normal Form (3NF):**

* A relation is in 3NF if:
  1. It is in **2NF**
  2. No **transitive dependencies**

**Example (Not 3NF):**

| **EmpID** | **DeptID** | **DeptName** |
| --- | --- | --- |
| 201 | 10 | HR |

* Transitive dependency: EmpID → DeptID → DeptName

👉 Solution: Decompose

* R1(EmpID, DeptID)
* R2(DeptID, DeptName)

**5. Define BCNF. How Does it Differ from 3NF?**

**Boyce-Codd Normal Form (BCNF):**

* A relation is in **BCNF** if:
  + For every functional dependency **X → Y**,  
    **X is a superkey**.

**Difference Between BCNF and 3NF:**

| **3NF** | **BCNF** |
| --- | --- |
| Allows some FDs where X is not a superkey | Requires X to always be a superkey |
| Less strict | Stricter form of normalization |
| Easier to achieve without decomposition | May need more decomposition |

**Example:**

| **StudentID** | **Course** | **Instructor** |
| --- | --- | --- |
| 101 | Math | Dr. A |

FDs:

* StudentID, Course → Instructor (Primary Key)
* Instructor → Course (Not a superkey)

✅ 3NF: Satisfied  
❌ BCNF: Violated (Instructor is not a superkey)

**UNIT -2**

1.Consider the following tables: Employee (Emp\_no, Name, Emp\_city) Company (Emp\_no, Company\_name, Salary)

i. Write a SQL query to display Employee name and company name.

ii. Write a SQL query to display employee name, employee city ,company name and salary of all the employees whose salary >10000

iii. Write a query to display all the employees working in “XYZ company

**Given Tables:**

**Employee(Emp\_no,Name,Emp\_city)**  
**Company (Emp\_no, Company\_name, Salary)**

**i. Write a SQL query to display Employee name and company name.**

sql

SELECT E.Name, C.Company\_name

FROM Employee E

JOIN Company C ON E.Emp\_no = C.Emp\_no;

**ii. Write a SQL query to display employee name, employee city, company name, and salary of all the employees whose salary > 10000**

sql

SELECT E.Name, E.Emp\_city, C.Company\_name, C.Salary

FROM Employee E

JOIN Company C ON E.Emp\_no = C.Emp\_no

WHERE C.Salary > 10000;

**iii. Write a query to display all the employees working in “XYZ company”**

sql

SELECT E.\*

FROM Employee E

JOIN Company C ON E.Emp\_no = C.Emp\_no

WHERE C.Company\_name = 'XYZ company';

**Unit 3**

**1. Immediate Update and Deferred Update Recovery Techniques**

**Introduction:**

In database systems, failures can occur due to hardware malfunctions, system crashes, or software bugs. To ensure **data integrity**, recovery techniques are implemented. **Immediate update** and **deferred update** are two such recovery strategies that manage how and when changes made by transactions are reflected in the database.

**A. Deferred Update**

**Definition:**

In **deferred update**, the system **does not immediately apply changes** to the database. Instead, all changes are written to a **log or buffer** during transaction execution. Actual updates are made **only after the transaction successfully commits**.

**How it Works:**

1. Changes made by a transaction are recorded in a temporary log.
2. No changes are made to the database until the transaction commits.
3. Once the transaction commits:
   * All recorded changes are written to the database (called **write-ahead logging**).
   * This ensures atomicity and durability.

**Advantages:**

* Easy to implement.
* No need for undo operations as no changes are made before commit.
* Suitable for systems with low concurrency.

**Disadvantages:**

* Delay in reflecting updates in the database.
* High memory usage for buffering updates.

**Recovery Process:**

* If system crashes before commit: **no changes** made to DB; just discard the log.
* If system crashes after commit: use the log to **redo** the changes.

**B. Immediate Update**

**Definition:**

In **immediate update**, changes made by transactions are **applied to the database as soon as they are executed**, **before** the transaction commits.

**How it Works:**

1. Each update is recorded in the **log** before it is applied to the database (**Write-Ahead Logging - WAL**).
2. If a transaction fails or the system crashes, **undo** operations are performed to revert changes of uncommitted transactions.

**Advantages:**

* Faster feedback and update reflection.
* Better for systems requiring real-time data.

**Disadvantages:**

* More complex recovery process (both **undo and redo** may be required).
* Riskier as database may have partial changes from uncommitted transactions.

**Recovery Process:**

* If transaction is **not committed**, undo changes using the log.
* If transaction is **committed**, redo the changes if needed.

**Comparison Table:**

| **Feature** | **Deferred Update** | **Immediate Update** |
| --- | --- | --- |
| When changes are applied | After commit | Immediately after execution |
| Undo needed? | No | Yes (for uncommitted data) |
| Redo needed? | Yes (for committed data) | Yes |
| Complexity | Simple | Complex |
| Efficiency | Better for low concurrency | Better for real-time systems |

**2. Explain the Concept of Deadlock Avoidance and Prevention in Detail**

**What is a Deadlock?**

In a DBMS, a **deadlock** occurs when two or more transactions are waiting for each other to release resources, creating a **circular wait** that prevents all of them from proceeding.

**Example:**

* T1 locks data item A, T2 locks data item B.
* T1 requests B (held by T2), T2 requests A (held by T1).
* Both wait forever ⇒ **Deadlock**.

**Deadlock Handling Techniques:**

There are 3 main approaches:

1. Deadlock **Prevention**
2. Deadlock **Avoidance**
3. Deadlock **Detection and Recovery**

Here we discuss **Prevention** and **Avoidance** in detail.

**A. Deadlock Prevention**

Deadlock prevention ensures that at least one of the necessary conditions for deadlock **never occurs**.

**Necessary Conditions for Deadlock:**

1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait

**How to Prevent Deadlock:**

1. **Eliminate Hold and Wait:**
   * Require transactions to **request all needed resources at once**.
   * If not available, transaction waits, holding nothing.
2. **Eliminate No Preemption:**
   * If a transaction is waiting and a resource is not available, **forcibly release all held resources**.
3. **Eliminate Circular Wait:**
   * Impose a **total ordering** on resources (e.g., T must request resources in a specific order).
   * Prevents circular wait by design.

**Advantages:**

* Simple logic
* Avoids deadlocks proactively

**Disadvantages:**

* Can be overly conservative
* May lead to **resource underutilization**

**B. Deadlock Avoidance**

Deadlock avoidance requires the system to **dynamically examine resource allocation requests** and determine if they lead to a deadlock **before granting** them.

**Technique Used:**

**Banker’s Algorithm** (similar to memory allocation in OS):

* System keeps track of:
  + **Available resources**
  + **Allocated resources**
  + **Max demand of each transaction**
* Before granting a resource, the system checks whether the state is **safe** (i.e., all transactions can complete).

**Safe State:**

A state is **safe** if there is at least one sequence in which all transactions can execute to completion **without getting into deadlock**.

**Example:**

If granting a resource to a transaction might lead to a potential deadlock, the system **denies** the request until it’s safe.

**Advantages:**

* More flexible than prevention
* Resources are used efficiently

**Disadvantages:**

* Requires advance knowledge of maximum resource needs
* Complex to implement

**Comparison:**

| **Feature** | **Deadlock Prevention** | **Deadlock Avoidance** |
| --- | --- | --- |
| Method | Prevent one of the four conditions | Analyze before granting resources |
| Resource utilization | Less efficient | More efficient |
| Complexity | Simpler | More complex (needs safety check) |
| Example Algorithm | Resource ordering, preemption | Banker’s Algorithm |

**UNIT-4**

**1. Write about the Various Levels of RAID with Neat Diagrams**

**Introduction:**

RAID (Redundant Array of Independent Disks) is a data storage virtualization technology that combines multiple physical disk drives into one logical unit to improve **performance, fault tolerance, and reliability**.

**Types/Levels of RAID:**

**RAID 0 (Striping):**

* **Data is split across multiple disks.**
* No redundancy, no fault tolerance.
* **Improves performance** (read/write).

**Diagram:**

mathematica

CopyEdit

Disk 1: Block A1 | A3 | A5

Disk 2: Block A2 | A4 | A6

**Use case:** High-speed access with no concern for failure.

**RAID 1 (Mirroring):**

* **Exact copy** of data on two or more disks.
* **High fault tolerance**, but storage efficiency is 50%.

**Diagram:**

yaml

CopyEdit

Disk 1: A1 | A2 | A3

Disk 2: A1 | A2 | A3 ← Mirror

**Use case:** Systems requiring high availability (e.g., servers).

**RAID 2:**

* Uses **Hamming code for error correction**.
* Requires **synchronized disks**.
* Rarely used today.

**RAID 3:**

* **Byte-level striping** with a **dedicated parity disk**.
* All read/writes go to all disks.

**Diagram:**

mathematica

CopyEdit

Disk 1: Byte 1 | Byte 4

Disk 2: Byte 2 | Byte 5

Disk 3: Byte 3 | Byte 6

Disk 4: Parity | Parity

**Use case:** High throughput systems like video editing.

**RAID 4:**

* **Block-level striping** with a **dedicated parity disk**.
* Better than RAID 3 for read operations.

**RAID 5:**

* **Block-level striping** with **distributed parity**.
* **No single parity disk** → Better fault tolerance and performance.

**Diagram:**

less

CopyEdit

Disk 1: A1 | A2 | P

Disk 2: A2 | P | A3

Disk 3: P | A3 | A1

**Use case:** Most common for business and enterprise.

**RAID 6:**

* Similar to RAID 5, but with **double distributed parity**.
* Can withstand **two disk failures**.

**RAID 10 (1+0):**

* Combination of RAID 1 (mirroring) and RAID 0 (striping).
* High performance and redundancy.

**Comparison Table:**

| **RAID Level** | **Redundancy** | **Performance** | **Storage Efficiency** | **Fault Tolerance** |
| --- | --- | --- | --- | --- |
| RAID 0 | No | High | 100% | None |
| RAID 1 | Yes | Moderate | 50% | 1 disk |
| RAID 5 | Yes | High | (N-1)/N | 1 disk |
| RAID 6 | Yes | Moderate | (N-2)/N | 2 disks |
| RAID 10 | Yes | High | 50% | Up to 2 disks |

**2. Construct a B+ Tree (Order 3) for the Keys: 5, 3, 4, 9, 7, 15, 14, 21, 22, 23**

**B+ Tree Basics (Order = 3):**

* Each **internal node** can have **at most 2 keys** and **3 children**.
* **Leaf nodes** contain actual values and are **linked**.
* Internal nodes **guide the search** but **do not store full records**.

**Step-by-Step Construction:**

**Step 1: Insert 5**

* Tree has only one node with key:  
  [5]

**Step 2: Insert 3**

* [3, 5] → No overflow

**Step 3: Insert 4**

* [3, 4, 5] → Overflow! Split the node.
* Middle key = 4 → Move **up**.

css

CopyEdit

[4]

/ \

[3] [5]

**Step 4: Insert 9**

* Goes to right child → [5, 9]

**Step 5: Insert 7**

* Right child: [5, 9, 7] → Sort = [5, 7, 9] → Overflow!
* Split → Middle key 7 goes up → [4, 7]

less

CopyEdit

[4, 7]

/ | \

[3] [5] [9]

**Step 6: Insert 15**

* Goes to rightmost child → [9, 15]

**Step 7: Insert 14**

* Rightmost: [9, 15, 14] → Sort = [9, 14, 15] → Overflow!
* Split → Middle 14 goes up

less

CopyEdit

[4, 7, 14]

/ | | \

[3] [5] [9] [15]

**Step 8: Insert 21**

* Goes to [15] → [15, 21]

**Step 9: Insert 22**

* [15, 21, 22] → Sort = [15, 21, 22] → Overflow!
* Split → 21 goes up
* Now, parent is full: [4, 7, 14] → Insert 21 → Overflow!
* Split root → [7] becomes new root.

less

CopyEdit

[7]

/ \

[4] [14, 21]

/ \ / | \

[3] [5] [9] [15] [22]

**Step 10: Insert 23**

* Goes to [22] → [22, 23]

✅ **Final B+ Tree Structure (Order 3):**

less

CopyEdit

[7]

/ \

[4] [14, 21]

/ \ / | \

[3] [5] [9] [15] [22,23]

* All **leaf nodes** are **linked** for sequential access.

**UNIT -5**

**1. Classification and Clustering Techniques**

**Introduction to Data Mining:**

Data mining refers to extracting useful patterns and knowledge from large datasets. Two major types of data analysis in data mining are **Classification** and **Clustering**.

**A. Classification Techniques**

**Definition:**

Classification is a **supervised learning** technique where the goal is to predict the **class label** of an object based on its attributes.

* Input: Training data with known labels
* Output: A model to classify new/unlabeled data

**Common Classification Techniques:**

1. **Decision Tree Classifier:**
   * Tree-like structure where each node tests an attribute.
   * Final leaves represent class labels.
   * Example: ID3, C4.5 algorithms.
2. **Naïve Bayes Classifier:**
   * Based on Bayes Theorem.
   * Assumes independence between features.
   * Fast and efficient for large datasets.
3. **Support Vector Machine (SVM):**
   * Finds optimal hyperplane to separate different classes.
   * Good for high-dimensional data.
4. **K-Nearest Neighbor (KNN):**
   * Classifies data based on the ‘k’ closest training examples.
   * No training time, but slow at runtime.
5. **Neural Networks:**
   * Inspired by the human brain.
   * Uses layers of neurons to learn complex relationships.
   * Good for large and nonlinear data.

**B. Clustering Techniques**

**Definition:**

Clustering is an **unsupervised learning** technique that groups data into **clusters** so that objects in the same cluster are more similar to each other than to those in other clusters.

* Input: Unlabeled data
* Output: Clusters with similar items

**Common Clustering Techniques:**

1. **K-Means Clustering:**
   * Divides data into ‘k’ clusters.
   * Iteratively updates cluster centroids.
   * Fast and scalable but sensitive to initial centroids.
2. **Hierarchical Clustering:**
   * Builds a hierarchy of clusters.
   * Can be:
     + Agglomerative (bottom-up)
     + Divisive (top-down)
3. **DBSCAN (Density-Based Spatial Clustering):**
   * Groups together dense areas of data points.
   * Can find clusters of arbitrary shape.
   * Can detect noise and outliers.
4. **Fuzzy C-Means:**
   * Data points can belong to **multiple clusters** with degrees of membership.
   * Useful in situations with overlapping categories.

**Comparison Table:**

| **Feature** | **Classification** | **Clustering** |
| --- | --- | --- |
| Learning Type | Supervised | Unsupervised |
| Output | Class Labels | Grouped Clusters |
| Labelled Data | Required | Not Required |
| Example Algorithm | Decision Tree, SVM | K-Means, DBSCAN |

**2. Association and Regression Techniques**

**A. Association Rule Mining**

**Definition:**

Association rule mining is used to **discover interesting relationships** between variables in large databases. Commonly used in **market basket analysis**.

**Key Concepts:**

* **Support:** Frequency of itemset in the database.
* **Confidence:** Likelihood that B occurs when A occurs (A ⇒ B).
* **Lift:** Measures importance of the rule beyond random chance.

**Example:**

If 70% of the time, people who buy bread also buy butter, the rule is:

* Bread ⇒ Butter (Support = 0.7, Confidence = 0.7)

**Algorithms:**

1. **Apriori Algorithm:**
   * Uses breadth-first search.
   * Removes infrequent itemsets early using support threshold.
2. **FP-Growth Algorithm:**
   * Uses a compact structure called FP-Tree.
   * More efficient than Apriori.

**B. Regression Techniques**

**Definition:**

Regression is a **supervised learning** technique used for **predicting a continuous value** based on input variables.

**Types of Regression:**

1. **Linear Regression:**
   * Models relationship between dependent and independent variables using a straight line.
   * Example: Predicting house price based on area.
2. **Multiple Linear Regression:**
   * Uses more than one predictor variable.
3. **Logistic Regression:**
   * Used for classification problems.
   * Outputs probabilities using the sigmoid function.
4. **Polynomial Regression:**
   * Fits a curve instead of a line to the data.
   * Useful for nonlinear relationships.

**Use Cases:**

* Predicting sales, stock prices, temperatures.
* Estimating customer lifetime value.

**Comparison Table:**

| **Feature** | **Association Rules** | **Regression** |
| --- | --- | --- |
| Type | Pattern Discovery | Predictive Modeling |
| Output | Item Relationships | Numeric Values |
| Example Use Case | Market Basket Analysis | Price Prediction |

**3. Explain Briefly the Retrieval of Information**

**Introduction to Information Retrieval (IR):**

Information Retrieval is the process of obtaining relevant data or documents from large collections based on a **user query**. It's commonly used in **search engines, digital libraries, and recommendation systems**.

**Components of an IR System:**

1. **Document Collection:**
   * A large repository of documents (e.g., web pages, PDFs, books).
2. **Query:**
   * User input requesting information.
3. **Search Engine:**
   * Matches the query to documents using keywords, indexing, and ranking.
4. **Indexing:**
   * IR systems create an **inverted index** (term → list of documents containing the term).
   * Improves speed and efficiency of searches.

**Key Techniques in IR:**

1. **Keyword-Based Retrieval:**
   * Searches based on exact words in the query.
   * Example: Google search.
2. **Boolean Retrieval:**
   * Uses logical operators (AND, OR, NOT).
   * Example: (AI AND healthcare) NOT pandemic
3. **Vector Space Model (VSM):**
   * Documents and queries are represented as vectors.
   * **Cosine similarity** is used to find relevance.
4. **Relevance Feedback:**
   * The system improves results based on user interactions.

**Performance Measures:**

* **Precision:** Fraction of retrieved documents that are relevant.
* **Recall:** Fraction of relevant documents that are retrieved.
* **F-Measure:** Harmonic mean of precision and recall.

**Applications of IR:**

* Web search engines (Google, Bing)
* Digital libraries
* E-commerce search (Amazon, Flipkart)
* Content-based filtering (YouTube, Netflix)