HEXAWARE PHASE-2 FOUNDATION TRAINING

DATABASE MANAGEMENT ASSIGNMENT

Section A: Introduction to SQL/NoSQL

1. You are working on a project where you need to store large amounts of structured and semi-structured data. Which type of database (SQL or NoSQL) would you choose and why? Explain with a practical example.

NoSQL is the preferred choice because it is more scalable and flexible for handling large amounts of structured and semi-structured data.

Example: A social media platform like Facebook stores diverse data types such as user posts, comments, and images. These data types are semi-structured and benefit from NoSQL databases like MongoDB, which allows fast storage and retrieval.

---

2. A company wants to migrate from a relational database to a NoSQL database for better scalability. What challenges might they face? Discuss with an example.

Challenges:

- Schema Design Differences – SQL databases require structured schemas, while NoSQL databases are schema-less.
- Query Language – SQL databases use structured queries (SQL), whereas NoSQL databases have their own query methods.
- Data Consistency – SQL databases follow ACID principles, while NoSQL follows eventual consistency, which may not be ideal for transactional systems.

Example: A financial company migrating from MySQL to Cassandra may face difficulties in maintaining strong consistency for real-time transactions.

Section B: Advantages and Disadvantages of SQL/NoSQL

3. You are designing an e-commerce website's database. Explain the advantages and disadvantages of using SQL vs. NoSQL in this scenario.

Database Type	Advantages	Disadvantages
SQL (MySQL, PostgreSQL)	1. Ensures data integrity and ACID compliance. 2. Complex queries with JOINs are possible. 3. Well-structured for transactional data.	1. Less scalable for large datasets. 2. Slow performance when handling unstructured data.
NoSQL (MongoDB, DynamoDB)	1. Highly scalable for handling large traffic. 2. Supports unstructured data (customer reviews, product recommendations). 3. Faster reads/writes for real-time updates.	1. Lacks ACID compliance, making transactional consistency difficult. 2. No standard query language (different databases use different syntaxes).

Conclusion: Use SQL for transactions (orders, payments) and NoSQL for product catalogs, customer reviews, and recommendations.

4. A banking system requires high consistency and ACID compliance. Which database system (SQL or NoSQL) would you recommend? Justify your answer with a real-world use case.

SQL is the best choice because it ensures strong consistency, data integrity, and ACID compliance, which are essential for financial transactions.

Example: A banking system (e.g., Citibank, HSBC) needs MySQL or PostgreSQL to handle secure transactions, ensuring that customer deposits and withdrawals are accurate and consistent.

Section C: Managing Databases

5. You are a database administrator and need to perform routine maintenance on a production database. Describe at least three essential database management tasks you would perform.

Essential Tasks:

- Backup and Recovery: Regularly back up databases to prevent data loss.
- Index Optimization: Optimize indexes for faster query performance.
- Security Audits: Implement access controls and check for vulnerabilities.

6. An online streaming service needs to optimize its database performance. What strategies can be used for effective database management in this case?

Strategies:

- Use Caching: Reduce load on the database by caching frequently accessed content (e.g., Redis, Memcached).
- Partitioning: Distribute data across multiple servers to improve read/write performance.
- Load Balancing: Distribute queries across multiple database instances.

Section D: Identifying System Databases in SQL Server

7. List and describe the system databases in SQL Server. Provide one practical use case for each system database.

System Database	Description	Use Case
master	Stores system-level information such as login accounts, configurations.	Used for managing user accounts and system settings.
model	Serves as a template for new databases.	Any new database inherits settings from the model database.
msdb	Manages jobs, schedules, and alerts.	Used for automating backup tasks.
tempdb	Stores temporary objects and intermediate query results.	Used for sorting operations in large queries.

---

8. You have accidentally deleted a user database in SQL Server. Which system database would you use to recover it, and how?

The msdb database contains backup history, which can help restore the deleted database.

Steps to Recover:

RESTORE DATABASE UserDB FROM DISK = 'C:\backup\UserDB.bak'

Alternative: If no backup exists, use recovery tools like SQL Server Management Studio (SSMS).

Section E: Normalization Forms (1NF, 2NF, 3NF, BCNF)

9. Given the following unnormalized table:

OrderID	CustomerName	Product	Quantity	SupplierName	SupplierContact
101	John Doe	Laptop	1	ABC Ltd.	1234567890
102	Jane Smith	Phone	2	XYZ Inc.	9876543210

1NF (First Normal Form - Atomicity)

Remove duplicate/multi-valued attributes and ensure each column contains atomic values.

Split order and product details into separate tables.

Orders Table:

OrderID	CustomerName
101	John Doe
102	Jane Smith

OrderDetails Table:

OrderID	Product	Quantity	SupplierName	SupplierContact
101	Laptop	1	ABC Ltd.	1234567890
102	Phone	2	XYZ Inc.	9876543210

2NF (Second Normal Form - No Partial Dependencies)

Remove partial dependencies by separating suppliers into a new table.

Suppliers Table:

SupplierID	SupplierName	SupplierContact
1	ABC Ltd.	1234567890
2	XYZ Inc.	9876543210

OrderDetails Table (Updated):

OrderID	Product	Quantity	SupplierID
101	Laptop	1	1

OrderID	Product	Quantity	SupplierID
102	Phone	2	2

3NF (Third Normal Form - No Transitive Dependencies)

Remove transitive dependencies by separating customers into a new table.

Customers Table:

CustomerID	CustomerName
1	John Doe
2	Jane Smith

Orders Table (Updated):

OrderID	CustomerID
101	1
102	2

---

10. A company is facing redundancy issues in their database. How would applying BCNF help reduce redundancy? Explain with an example.

Before BCNF (Redundant Data)

Course	Instructor	Department
DBMS	Prof. A	CS
Networks	Prof. B	CS
AI	Prof. A	CS
DBMS	Prof. C	IT

Here, Instructor → Department causes redundancy.

After BCNF (Redundancy Removed)

Courses Table:

CourseID	CourseName
1	DBMS
2	Networks
3	AI

Instructors Table:

InstructorID	InstructorName	Department
1	Prof. A	CS
2	Prof. B	CS
3	Prof. C	IT

CourseInstructor Table (New Relationship Table):

CourseID	InstructorID
1	1
1	3
2	2
3	1

This ensures that each determinant (Instructor) is a candidate key, eliminating redundancy.