Unit 1: Essential Database Concepts

Q. What is a database? (Nov 22)

Ans. A database is an organized collection of related data stored electronically. It allows easy access, management, and updating of information.

Q. Which key is used in relational database? Define Primary and Foreign keys. (Nov 22)

Ans. Primary and foreign keys are used in RDBMS to establish relationships between tables. A **Primary Key** uniquely identifies each record; a **Foreign Key** links one table to another using the primary key of the related table.

Q. Data (Nov 23)

Ans. Data refers to raw facts and figures that can be processed into meaningful information. In databases, data is stored in structured formats like rows and columns.

Q. Records (Nov 23)

Ans. A record is a complete set of related data items stored in a single row of a table. Each record represents one unit of data, such as a customer or order.

Q. Foreign Key (Nov 23)

Ans. A foreign key is a field in one table that refers to the primary key in another table. It is used to maintain referential integrity between related tables.

Q. Define the terms 'Record', 'Field' and 'Table' in terms of Database. (Nov 24)

Ans. A **Record** is a row in a table containing related data items; a **Field** is a column representing a single attribute; a **Table** is a collection of records organized in rows and columns.

Q. Why RDBMS is called as relational? (Nov 24)

Ans. RDBMS is called relational because it stores data in tables that are related to each other using keys. These relationships help in organizing and retrieving data efficiently.

Q. Define levels of database systems. (Nov 24)

Ans. Database systems have three levels: **internal (physical storage)**, **conceptual (logical structure)**, and **external (user view)**. These levels separate data abstraction and management.

Long Questions:

Q. Define the term DBMS and its functions. Discuss the responsibilities of DBA. Also explain the three level architecture of DBMS. (Nov 22), (Nov 23)

Ans. A **Database Management System (DBMS)** is software that allows users to define, create, manage, and manipulate databases. It provides an interface between the user and the database, enabling easy data storage, retrieval, and management while maintaining data integrity and security.

Functions of DBMS include:

- Data storage, retrieval, and update
- Ensuring data integrity and security
- Transaction management and concurrency control
- Backup and recovery
- Enforcing data constraints and relationships

The **Database Administrator (DBA)** has key responsibilities such as:

- Installing and configuring the DBMS
- Creating and managing user access and permissions
- Monitoring performance and optimizing queries
- Performing backups and restores
- Ensuring security and data consistency

The **three-level architecture of DBMS** includes:

1. **Internal Level** – Describes physical storage of data on hardware.

2. **Conceptual Level** – Describes the logical structure of the entire database.
3. **External Level** – Describes user views of the database, allowing different users to access the same data in different ways.

This architecture provides data abstraction and ensures independence between application and physical storage.

Q. What do you mean by Relational database system? Explain with example. (Nov 22)

Ans. A **Relational Database System** is a type of DBMS that stores data in the form of related tables (also called relations). Each table consists of **rows (records)** and **columns (fields)**. This model is based on the mathematical concept of relations and was introduced by **E.F. Codd**.

Each table has a **primary key** that uniquely identifies each row, and relationships between tables are established using **foreign keys**. The relational model allows for **structured querying** using SQL (Structured Query Language), which makes data manipulation easy and efficient.

Example:

Consider two tables:

Students Table

StudentID	Name	Age
1	Anya	21
2	Raj	22

Courses Table

CourseID	CourseName	StudentID
101	Math	1
102	English	2

Here, **StudentID** in the Courses table is a **foreign key** referring to the Students table. This relational setup allows efficient data organization and retrieval using queries.

Q. Differentiate between a file and a database. (Nov 23)

Ans. A **file** is a simple method of storing data, usually in formats like .txt or .csv, without any built-in structure or relationships. In contrast, a **database** is a structured system that stores data in an organized format using tables, allowing relationships between different types of data.

Differences:

1. **Structure:**
   - File: Flat and unstructured.
   - Database: Highly structured with tables, rows, and columns.
2. **Data Redundancy:**
   - File: Higher chances of data duplication.
   - Database: Reduced redundancy using normalization and keys.
3. **Data Integrity:**
   - File: No inbuilt support.
   - Database: Supports constraints and rules.
4. **Security and Access:**
   - File: Basic access control.
   - Database: Advanced user management and encryption.
5. **Concurrency:**
   - File: Difficult to handle multiple users.
   - Database: Supports multiple users simultaneously.
6. **Querying:**
   - File: Limited to manual or program-based access.
   - Database: Querying possible using SQL.

Thus, databases offer better data management, retrieval, and control compared to traditional file systems.

Q. What are the advantages of DBMS over traditional file processing system? Explain. (Nov 22), (Nov 24)

Ans. A **Database Management System (DBMS)** offers several advantages over the traditional **file processing system**, which lacks structure, security, and scalability. In the file system, data is stored in individual files with limited connectivity and no centralized control, often leading to data redundancy, inconsistency, and inefficiency.

Advantages of DBMS:

1. **Data Redundancy Control**: DBMS minimizes data duplication by centralizing data storage and allowing shared access.
2. **Data Consistency and Integrity**: It enforces rules and constraints, ensuring accuracy and consistency across the database.
3. **Data Security**: DBMS provides user authentication, authorization, and encryption to protect sensitive data.
4. **Efficient Data Access**: Structured Query Language (SQL) allows fast, flexible data retrieval using queries.
5. **Concurrent Access**: Multiple users can access and modify data simultaneously with proper control.
6. **Backup and Recovery**: DBMS supports automated backup and restore functionalities to prevent data loss.
7. **Scalability and Flexibility**: It supports large amounts of data and adapts easily to changing needs.
8. **Data Independence**: Changes in the data structure do not affect application programs.

Overall, DBMS provides a robust and organized approach to data management, overcoming most of the limitations of the file-based system.

Q. Compare and contrast hierarchical, network and relational data models. (Nov 24)

Ans. **Hierarchical, network, and relational data models** are three classic models used to structure and manage data in databases.

1. **Hierarchical Model**:
    - Data is organized in a tree-like structure with a parent-child relationship.
    - Each child has only one parent (one-to-many relationship).
    - Navigation is rigid and data retrieval is fast but inflexible.
    - Example: File systems, old IBM databases.
2. **Network Model**:
    - Data is represented using graph structures with records connected by links.
    - Supports many-to-many relationships.
    - More flexible than hierarchical but complex to manage.
    - Example: CODASYL databases.
3. **Relational Model**:
    - Data is stored in tables (relations) with rows and columns.
    - Relationships are maintained using **primary** and **foreign keys**.
    - Uses SQL for data manipulation and provides better flexibility, security, and simplicity.
    - Most commonly used today (e.g., MySQL, MS SQL Server).

Comparison:

- **Ease of Use**: Relational > Network > Hierarchical
- **Flexibility**: Relational is the most flexible and scalable.
- **Performance**: Hierarchical and Network may perform better for fixed, pre-defined queries but lack relational model's versatility.

In summary, the **relational model** is widely preferred for its simplicity, data integrity, and ease of use.

Q. Construct an E-R diagram for a hospital with a set of patients and set of medical doctors. Associate with each patient, a log of various tests and examinations conducted. Construct the appropriate tables for the E-R diagram. (Nov 22)

Ans. An **Entity-Relationship (E-R) diagram** helps model real-world entities and their relationships. For a **hospital system**, the main entities are:

1. **Patient** – Attributes: Patient_ID (PK), Name, Age, Gender, Contact.
2. **Doctor** – Attributes: Doctor_ID (PK), Name, Specialization, Phone.
3. **Test/Examination** – Attributes: Test_ID (PK), Test_Name, Date, Result.

Relationships:

- Each **Patient** can undergo multiple **Tests** (one-to-many).
- Each **Test** is **conducted by a Doctor** (many-to-one or many-to-many depending on design).

E-R Diagram Summary:

- Entities: Patient, Doctor, Test
- Relationships:
  a) Patient–Test: One-to-Many
  b) Doctor–Test: One-to-Many

Tables:

Patient Table

| Patient_ID | Name | Age | Gender | Contact |

Doctor Table

| Doctor_ID | Name | Specialization | Phone |

Test Table

| Test_ID | Test_Name | Date | Result | Patient_ID (FK) | Doctor_ID (FK) |

This design maintains relationships using **foreign keys**, ensuring that tests are correctly linked to both the patient and the doctor who conducted them.

Q. What is the use of E-R Diagram in DBMS? List and explain various symbols and components used in E-R Diagrams. (Nov 24)

Ans. An **Entity-Relationship (E-R) Diagram** is a visual representation of the data model in a DBMS. It is used during the **database design phase** to understand and define the logical structure of the system. It helps identify entities, their attributes, and relationships among them, enabling better normalization and planning of tables.

Uses of E-R Diagrams:

- Simplifies complex database design.
- Helps in understanding data flow and relationships.
- Acts as a blueprint for creating relational schemas.
- Aids communication between developers and stakeholders.

Common Symbols and Components in E-R Diagrams:

1. **Entity**: Represented by a rectangle. Example: STUDENT, DOCTOR.
2. **Attribute**: Represented by an oval. Describes properties of entities. Example: Name, Age.
3. **Primary Key**: Underlined attribute identifying each entity uniquely.
4. **Relationship**: Represented by a diamond shape connecting entities. Example: “Enrolled”, “Assigned”.
5. **Cardinality**: Specifies how many instances of one entity relate to another (1:1, 1:N, M:N).
6. **Weak Entity**: Shown with double rectangle; relies on another entity.
7. **Multivalued Attribute**: Represented by double ovals.

E-R diagrams guide the transition from conceptual design to logical schema in relational databases.

Unit 2: Introduction to SQL Server

Q. List basic features of SQL server. (Nov 22)

Ans. SQL Server offers features like data security, scalability, transaction control, and support for stored procedures. It also provides tools for backup, replication, and data analysis.

Q. What is a view in SQL? (Nov 22)

Ans. A **view** is a virtual table based on a SQL query that presents selected data from one or more tables. It simplifies complex queries and enhances security by limiting access.

Q. What is a JOIN? Mention its types. (Nov 22)

Ans. A **JOIN** is used to combine rows from two or more tables based on a related column. Types include INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN.

Q. DML (Nov 23)

Ans. **DML (Data Manipulation Language)** includes commands like SELECT, INSERT, UPDATE, and DELETE. It is used to manage and manipulate data in SQL Server tables.

Q. Indexing (Nov 23)

Ans. Indexing improves the speed of data retrieval in a database table. SQL Server supports clustered and non-clustered indexes for optimized performance.

Q. TRIM and LTRIM (SQL Server Functions) (Nov 23)

Ans. **TRIM** removes both leading and trailing spaces from a string. **LTRIM** removes only the leading spaces in SQL Server.

Q. Components of SQLServer (Nov 23)

Ans. Key components include the **Database Engine**, **SQL Server Agent**, **Analysis Services**, **Reporting Services**, and **Integration Services**, each handling specific tasks in data processing.

Q. What is SQL server? (Nov 24)

Ans. SQL Server is a relational database management system developed by Microsoft. It is used to store, manage, and retrieve data using structured query language (SQL).

Q. List four tools used for monitoring components of SQL Server? (Nov 24)

Ans. Tools include **SQL Server Profiler**, **Activity Monitor**, **Performance Monitor**, and **SQL Server Management Studio (SSMS)** for real-time tracking and analysis.

Q. What is the difference between DELETE and TRUNCATE commands? (Nov 24)

Ans. **DELETE** removes selected rows and logs each row for rollback, while **TRUNCATE** removes all rows quickly without logging each row and cannot be rolled back easily.

Long Questions:

Q. List the different categories of SQL Commands. Also write and explain the commands used in Data Definition Language, Data Manipulation Language and Data Control Language. (Nov 22)

Ans. SQL (Structured Query Language) commands are grouped into five major categories:

1. **Data Definition Language (DDL)** – Defines and modifies database structure.
2. **Data Manipulation Language (DML)** – Handles data stored in tables.
3. **Data Control Language (DCL)** – Manages access and permissions.
4. **Transaction Control Language (TCL)** – Manages transactions.
5. **Data Query Language (DQL)** – Mainly consists of SELECT for querying data.

DDL Commands:

- **CREATE**: Used to create tables or databases. Example: CREATE TABLE Student (ID INT, Name VARCHAR(50));

- **ALTER**: Modifies the structure of a table. Example: ALTER TABLE Student ADD Age INT;

DML Commands:

- **INSERT**: Adds new records. Example: INSERT INTO Student VALUES (1, 'Raj');
- **UPDATE**: Modifies existing data. Example: UPDATE Student SET Name='Rahul' WHERE ID=1;

DCL Commands:

- **GRANT**: Provides access privileges. Example: GRANT SELECT ON Student TO User1;
- **REVOKE**: Removes access rights. Example: REVOKE SELECT ON Student FROM User1;

These command types together provide full control over database management, structure, security, and operations.

Q. Differentiate Data Definition Language and Data Manipulation Language. Also write and explain two commands from each category. (Nov 24)

Ans. **DDL (Data Definition Language)** and **DML (Data Manipulation Language)** are both crucial parts of SQL but serve different purposes.

Difference:

- **DDL** defines the structure of database objects like tables and schemas.
- **DML** is used to manage and manipulate the data stored in those structures.

DDL Commands:

1. **CREATE**: Creates new database objects. Example: CREATE TABLE Books (BookID INT, Title VARCHAR(100));
2. **DROP**: Deletes objects from the database permanently. Example: DROP TABLE Books;

DML Commands:

1. **INSERT**: Adds new rows to a table. Example: INSERT INTO Books VALUES (101, 'SQL Basics');
2. **DELETE**: Removes rows based on a condition. Example: DELETE FROM Books WHERE BookID = 101;

DDL changes the database structure and is auto-committed, while DML changes data and can be rolled back. Both are essential for full database operation—from building to data handling.

Q. Explain five SQL commands with examples. (Nov 24)

Ans. Here are five commonly used SQL commands:

1. **CREATE** (DDL): Used to create new tables or databases. Example: CREATE TABLE Employee (EmpID INT, Name VARCHAR(50));
2. **INSERT** (DML): Adds new data to a table. Example: INSERT INTO Employee VALUES (1, 'Aman');
3. **SELECT** (DQL): Retrieves data from one or more tables. Example: SELECT * FROM Employee;
4. **UPDATE** (DML): Modifies existing records in a table. Example: UPDATE Employee SET Name='Ravi' WHERE EmpID=1;
5. **GRANT** (DCL): Provides user access to database objects. Example: GRANT SELECT ON Employee TO User1;

These commands are used for different tasks—defining structure, inserting and editing data, querying information, and controlling access. Mastery of these is essential for effective database management in SQL Server or any RDBMS.

Q. Explain the difference between SQL-DDL, DML and DCL with respect to SQL Server. (Nov 23)

Ans. SQL commands are grouped into various categories based on their function. The three major types are **DDL (Data Definition Language)**, **DML (Data Manipulation Language)**, and **DCL (Data Control Language)**. Each plays a distinct role in SQL Server operations.

DDL (Data Definition Language) is used to define and modify database structures like tables and schemas. Commands include:

- **CREATE** – creates database objects

- **ALTER** – modifies existing objects
- **DROP** – deletes objects

Example:
CREATE TABLE Student (ID INT, Name VARCHAR(50));
DML (Data Manipulation Language) is used to manipulate the data inside database tables. Commands include:

- **INSERT** – adds new records
- **UPDATE** – modifies existing records
- **DELETE** – removes records

Example:
UPDATE Student SET Name='Ravi' WHERE ID=1;
DCL (Data Control Language) deals with permissions and access control. Commands include:

- **GRANT** – gives access rights
- **REVOKE** – removes access rights

Example:
GRANT SELECT ON Student TO User1;
In SQL Server, these command types ensure that users can define, manage, and secure data in a structured and controlled manner.

Q. How do you start and stop SQL Server instances or services? (Nov 23)
Ans. In SQL Server, starting and stopping instances or services is essential for maintenance, updates, and troubleshooting. This can be done using various methods provided by Microsoft SQL Server.
1. Using SQL Server Configuration Manager:

- Open "SQL Server Configuration Manager".
- Under "SQL Server Services", you'll see the list of installed instances.
- Right-click on **SQL Server (InstanceName)** and choose **Start** or **Stop** as needed.

2. Using Services.msc:

- Press Win + R, type services.msc, and press Enter.
- Locate **SQL Server (InstanceName)** in the list.
- Right-click and select **Start**, **Stop**, or **Restart**.

3. Using Command Line (CMD):

- To start: net start "SQL Server (InstanceName)"
- To stop: net stop "SQL Server (InstanceName)"

4. Using SQL Server Management Studio (SSMS):

- You can only **connect** to running instances here.
- To control services, SSMS will redirect you to Configuration Manager.
  Properly managing SQL Server services helps ensure database performance, availability, and safety, especially during backups, updates, or server restarts.

Q. Consider the following schema : (Nov 24)
Employee (EmpNo , Name, Dno, Salary)
Department Dno, Deptt _ name, Headno)
Dependent (EmpNo, Dependent _ name, DOB)
In the above schema, domain of Headno is same as that of EmpNo. Write queries in SQL for the following to:
a)Retrieve the name of all the employees of each department getting maximum salary in the department.
b) Display the name of each department having employees greater than two.
c) Retrieve the name of each employee along with his dependent names.
d) Add a new constraint on salary (salary should lie in the range 1000 to 50000).
Ans. **Schema:**

- **Employee(EmpNo, Name, Dno, Salary)**
- **Department(Dno, Deptt_name, Headno)**
- **Dependent(EmpNo, Dependent_name, DOB)**

(*Note: Headno has the same domain as EmpNo*)

SQL Queries:

a) Retrieve the name of all the employees of each department getting maximum salary in the department:

```
SELECT E.Name, E.Dno
FROM Employee E
WHERE E.Salary = (
  SELECT MAX(Salary)
  FROM Employee
  WHERE Dno = E.Dno
);
```

b) Display the name of each department having employees greater than two:

```
SELECT D.Deptt_name
FROM Department D
JOIN Employee E ON D.Dno = E.Dno
GROUP BY D.Deptt_name
HAVING COUNT(E.EmpNo) > 2;
```

c) Retrieve the name of each employee along with his dependent names:

```
SELECT E.Name AS Employee_Name, D.Dependent_name
FROM Employee E
JOIN Dependent D ON E.EmpNo = D.EmpNo;
```

d) Add a new constraint on salary (salary should lie in the range 1000 to 50000):

```
ALTER TABLE Employee
ADD CONSTRAINT chk_salary
CHECK (Salary BETWEEN 1000 AND 50000);
```

These SQL queries help enforce data integrity, extract valuable employee insights, and implement business rules in the given database schema.

Unit 3: PL/SQL

Q. List the features of PL/SQL. (Nov 22)

Ans. PL/SQL supports procedural programming, exception handling, and modular code using functions and procedures. It integrates seamlessly with SQL for efficient data manipulation.

Q. SQL Implicit Cursor (Nov 23)

Ans. An **implicit cursor** is automatically created by SQL Server when a DML statement (like SELECT or INSERT) is executed. It handles data row-by-row behind the scenes without user control.

Q. SQL Procedure (Nov 23)

Ans. A **SQL procedure** is a stored set of SQL statements that can be executed repeatedly. It helps in reducing code duplication and improving performance and security.

Q. What are the advantages of PL/SQL? (Nov 24)

Ans. PL/SQL allows writing reusable and maintainable code with high performance. It also supports error handling and tight integration with SQL for advanced data processing.

Q. Define Cursor and its types. (Nov 22), (Nov 24)

Ans. A **cursor** is a database object used to retrieve and manipulate rows returned by a query. Types include **implicit** and **explicit cursors**—explicit ones offer more control over row handling.

Long Questions:

Q. What is a Trigger and What is the use of Trigger and its type? Explain using suitable examples (Nov 24), (Nov 22), (Nov 23)

Ans. A **Trigger** is a special type of stored procedure in a database that automatically executes in response to specific events on a table or view. These events can be **INSERT**, **UPDATE**, or **DELETE** operations. Triggers help enforce business rules, maintain audit trails, and validate data integrity without requiring explicit action from the user.

Uses of Triggers:

- Automatically update related tables.
- Log changes for audit.
- Enforce complex constraints.
- Prevent invalid transactions.

Types of Triggers in SQL Server:

1. **AFTER Triggers (FOR Triggers):** Execute after the triggering event.
2. **INSTEAD OF Triggers:** Execute in place of the triggering event.
3. **DDL Triggers:** Fire on schema-level events like CREATE or DROP.
4. **Logon Triggers:** Fire when a user connects to the database.

Example:

```
CREATE TRIGGER trg_after_update
ON Employee
AFTER UPDATE
AS
BEGIN
  INSERT INTO AuditLog(EmpID, ActionTime)
  SELECT EmpNo, GETDATE() FROM inserted;
END;
```

This trigger logs any update made to the **Employee** table into an **AuditLog** table. Triggers run automatically and enhance automation, consistency, and security.

Q. Write the differences between SQL & PL/SQL. (Nov 22)

Ans. **SQL (Structured Query Language)** and **PL/SQL (Procedural Language/SQL)** are both used to work with databases, but they differ in structure, capabilities, and purpose.

Feature	SQL	PL/SQL
Nature	Declarative	Procedural
Function	Executes a single query	Executes a block of code
Use Case	Used for data manipulation and DDL	Used for writing procedures, triggers
Flow Control	Not supported	Supports loops, conditions, etc.
Compilation	Not compiled, directly executed	Compiled into blocks
Performance	Executes one statement at a time	Executes multiple statements in batch

Example of SQL:

```
SELECT * FROM Employee;
```

Example of PL/SQL:

```
BEGIN
  UPDATE Employee SET Salary = Salary + 500 WHERE Dno = 2;
END;
```

In summary, **SQL** is ideal for basic database operations, while **PL/SQL** is more powerful for writing complex logic with flow control, making it suitable for building full applications within the database.

Q. What are COMMIT, ROLLBACK and SAVEPOINT statements in PL/SQL? (Nov 22)

Ans. In **PL/SQL**, transaction control statements are used to manage changes made by DML operations (like INSERT, UPDATE, DELETE). The key transaction control statements are **COMMIT**, **ROLLBACK**, and **SAVEPOINT**.

1. **COMMIT**: It makes all changes permanent in the database. Once committed, the changes cannot be undone. Example:
   ```
   INSERT INTO Employee VALUES (1, 'Ravi', 2, 3000);
   COMMIT;
   ```
2. **ROLLBACK**: It undoes changes made in the current transaction before the last **COMMIT**. Example:
   ```
   DELETE FROM Employee WHERE EmpNo = 1;
   ROLLBACK;
   ```
3. **SAVEPOINT**: It sets a point within a transaction to which you can **ROLLBACK** partially, without undoing the entire transaction. Example:
   ```
   SAVEPOINT sp1;
   UPDATE Employee SET Salary = 4000 WHERE EmpNo = 2;
   ROLLBACK TO sp1;
   ```

These statements ensure data consistency and allow developers to control the reliability of operations in PL/SQL applications.

Q. What is PL/SQL and what role does it play in Database Management? (Nov 23)

Ans. **PL/SQL (Procedural Language/Structured Query Language)** is Oracle's procedural extension to SQL, designed for writing application logic within the database. It combines the data manipulation power of SQL with procedural features such as loops, conditions, functions, and error handling.

PL/SQL plays a vital role in **Database Management** by enabling the development of complex and secure applications within the database itself. It allows grouping of multiple SQL statements into a single block, which improves performance and maintainability. Developers use PL/SQL to write stored procedures, functions, packages, triggers, and anonymous blocks that execute business logic automatically.

Key Roles of PL/SQL:

- Automating repetitive database tasks like billing or payroll.
- Enhancing data security by validating inputs before updating records.
- Improving performance by reducing client-server communication.

- Creating database triggers for auditing and enforcing rules.
- Handling exceptions and errors gracefully within applications.

PL/SQL ensures that business logic resides close to the data, improving performance and data consistency. It is widely used in enterprise applications for creating reusable and reliable database programs.

Q. a) Explain the basic structure followed by PL/SQL. List various data types used in PL/SQL. (Nov 24)
b) Write a program in PL/SQL to show the uses of implicit cursor without using any attribute.

Ans. **a.** PL/SQL programs follow a **block structure** that is easy to read, manage, and execute. Each PL/SQL block consists of three main sections:

1. **Declaration Section (optional)** – Variables, constants, and cursors are declared here.
2. **Begin Section (mandatory)** – Contains executable statements like SQL queries or control structures.
3. **Exception Section (optional)** – Handles errors or exceptions that may occur during execution.

Basic Syntax:

```
DECLARE
  salary NUMBER;
BEGIN
  SELECT Salary INTO salary FROM Employee WHERE EmpID = 1;
  DBMS_OUTPUT.PUT_LINE('Salary is ' || salary);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employee found.');
END;
```

Common Data Types in PL/SQL:

- **NUMBER** – for numeric values
- **VARCHAR2(n)** – for variable-length strings
- **DATE** – for date and time values
- **BOOLEAN** – for logical values (TRUE, FALSE)
- **CHAR(n)** – fixed-length strings
- **BLOB/CLOB** – for large objects (binary/text)

PL/SQL also supports user-defined types, %TYPE and %ROWTYPE, which reference table columns and entire rows, enhancing reusability and consistency.

b. In PL/SQL, **implicit cursors** are automatically created by the system for single-row SQL statements like SELECT INTO, INSERT, UPDATE, or DELETE. Here's an example using an implicit cursDECLARE

```
  emp_name VARCHAR2(50);
  emp_salary NUMBER;
BEGIN
  SELECT Name, Salary INTO emp_name, emp_salary
  FROM Employee
  WHERE EmpNo = 101;

  DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
  DBMS_OUTPUT.PUT_LINE('Salary: ' || emp_salary);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Some error occurred.');
END;
```

Explanation:

- The SELECT INTO statement uses an implicit cursor created by PL/SQL.
- No cursor attributes (like %FOUND, %NOTFOUND) are used here.

- The program fetches the employee name and salary and displays them.
- It handles exceptions gracefully if no record is found.

This example demonstrates how simple and effective implicit cursors can be for single-record operations in PL/SQL.

Unit 4: Backup and Restore

Q. What do you mean by database recovery? (Nov 22)

Ans. Database recovery is the process of restoring a database to a correct state after a failure. It ensures data integrity using backups and transaction logs.

Q. What is attaching a database? (Nov 22)

Ans. Attaching a database means connecting a detached .MDF (data file) and .LDF (log file) to SQL Server. It allows access to databases moved from another server or system.

Q. What is a database restore? (Nov 24)

Ans. Database restore is the process of recovering a database from a backup file. It is used to retrieve lost or corrupted data after crashes or errors.

Q. Detach SQL Server Database. (Nov 23)

Ans. Detaching a database removes it from SQL Server while keeping the data and log files intact. It is often used before moving the database to another server or location.

Long Questions:

Q. How many types of SQL Server database backups are there? Write a note on full backups and differential backups. (Nov 22)

Ans. In **SQL Server**, there are **five main types of database backups**:

1. **Full Backup**
2. **Differential Backup**
3. **Transaction Log Backup**
4. **File/Filegroup Backup**
5. **Copy-Only Backup**

1. Full Backup: A full backup captures the **entire database**, including all data, objects, system tables, and transaction logs (at the time of backup). It serves as the **foundation** for all other types of backups.

Example:

```
BACKUP DATABASE dbname TO DISK = 'C:\Backup\dbname_full.bak';
```

Full backups are ideal for restoring the database completely and are typically scheduled weekly or daily, depending on data sensitivity.

2. Differential Backup: A differential backup stores **only the changes made since the last full backup**, making it smaller and faster. It depends on the last full backup for restoration.

Example:

```
BACKUP DATABASE dbname TO DISK = 'C:\Backup\dbname_diff.bak' WITH DIFFERENTIAL;
```

To restore using differential backup, you first restore the full backup, then apply the latest differential backup.

These backups ensure **data recovery, minimize downtime**, and protect against data loss in case of hardware failure, user error, or corruption.

Q. What is the purpose of taking a database backup? (Nov 23)

Ans. The primary **purpose of taking a database backup** in SQL Server is to ensure **data protection and recovery** in case of unexpected failures. Backups act as a safety net to restore data after events like system crashes, hardware failures, accidental deletions, or malicious attacks (e.g., ransomware).

Key purposes include:

1. **Disaster Recovery**: A recent backup allows you to recover data after system failure or corruption.
2. **Data Safety**: Prevents permanent data loss by keeping a copy of critical information.
3. **Versioning**: Maintains historical copies that help revert databases to a previous state.
4. **Testing and Development**: Developers and testers use backups to restore data in test environments.
5. **Legal Compliance**: Some organizations must maintain data backups for regulatory reasons.

SQL Server supports various types of backups such as **full, differential, and transaction log backups**, each serving different recovery needs. For example, full backups restore the entire database, while differential and log backups help reduce backup size and recovery time.

In summary, **database backups are essential** for maintaining business continuity, reducing downtime, and ensuring the integrity and availability of enterprise data.

Q. Write a note on different types of database backups. (Nov 24)

Ans. SQL Server provides multiple types of backups to help protect data and ensure effective recovery in case of failures. The main types are:

1. **Full Backup:** This is the most comprehensive type, capturing the **entire database**, including data, objects, and part of the transaction log. It serves as the base for all other backup types.
2. **Differential Backup:** This backup contains only the **data changed since the last full backup**. It's faster and smaller than a full backup, but must be restored along with the last full backup.
3. **Transaction Log Backup:** Captures all **transaction log records** since the last log backup. It allows **point-in-time recovery** and is used in conjunction with full backups for minimal data loss.
4. **File or Filegroup Backup:** Backs up individual files or filegroups. Useful for very large databases where full backups may be time-consuming.
5. **Copy-Only Backup:** A standalone backup that doesn't affect the existing backup chain. It's used for temporary purposes like testing or migration.

Each type serves a specific role in a complete **backup strategy**. Combining them ensures efficient storage use, quicker recovery times, and maximum data protection.

Q. Explain the importance of database backup and restore procedures in maintaining data integrity. (Nov 23)

Ans. **Database backup and restore procedures** are critical for maintaining **data integrity, security, and availability** in any system that stores important or sensitive information. They protect organizations from data loss due to hardware failures, software bugs, cyberattacks, or human errors.

Importance of Backups:

- **Disaster Recovery:** In case of system crashes or corruption, backups allow the database to be restored to its last healthy state, ensuring business continuity.
- **Data Integrity:** Regular and tested backups ensure that restored data is **accurate, consistent, and uncorrupted**.
- **Protection Against Errors:** Accidental deletion or updates can be reversed using recent backups.
- **Support for Audits and Compliance:** Backups help meet legal and regulatory requirements regarding data retention.

Restore Procedures:

- A well-defined **restore strategy** ensures the correct use of **full, differential, and transaction log backups** in sequence.
- **Point-in-time recovery** allows restoration to a specific moment, minimizing data loss.
- Proper documentation and testing of restore procedures reduce downtime and prevent mistakes during emergencies.

In conclusion, regular database backups and efficient restore strategies are essential for safeguarding critical data, ensuring operational reliability, and maintaining the trust of users and stakeholders.