ARDBMS UNIT 3

Advantages of Stored Procedures

1. **Improved Performance**
    1. Procedures are compiled once and stored, reducing execution time.
    2. Reduces network traffic as multiple queries are executed in a single call.
2. **Code Reusability**
    1. Once created, a stored procedure can be used by multiple applications.
3. **Security and Access Control**
    1. Users can be granted execution privileges without giving direct access to tables.
    2. Hides business logic from users.
4. **Maintainability**
    1. Centralized business logic makes updates easier without modifying application code.

1. **Encapsulation**
    1. Complex operations can be encapsulated in procedures, improving modularity.
2. **Reduced Network Load**
    1. Instead of sending multiple queries over the network, only procedure calls are sent.

Disadvantages of Stored Procedures

1.Debugging Complexity

1. Debugging stored procedures can be challenging compared to application code.

2.Increased Server Load

1. Since procedures run on the database server, they can increase the load if not optimized.

3.Portability Issues

1. PL/SQL is Oracle-specific; porting stored procedures to other databases requires modification.

4.Dependency Management

1. Changes in table structures may require updating related stored procedures.

1.Version Control Challenges

1. Unlike application code, versioning stored procedures can be more complex.

```
CREATE OR REPLACE PROCEDURE procedure_name (
    param1 IN datatype,      -- Input parameter
    param2 OUT datatype,     -- Output parameter
    param3 IN OUT datatype   -- Input-Output parameter
)
IS
    -- Declare variables if needed
    var1 datatype;
BEGIN
    -- SQL and PL/SQL statements
    SELECT column_name INTO var1 FROM table_name WHERE condition;

    -- Assign values to output parameters
    param2 := var1;

    -- Business logic and calculations
    IF param1 > 100 THEN
        DBMS_OUTPUT.PUT_LINE('High value');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Low value');
    END IF;

    -- Exception Handling
    EXCEPTION
```

Stored Procedure in PL/SQL

Introduction

A **Stored Procedure** is a precompiled collection of SQL statements and procedural logic stored in the database. It allows for efficient execution, code reuse, security, and improved performance by reducing network traffic.

Stored procedures in **PL/SQL (Procedural Language/Structured Query Language)** are used to encapsulate business logic, enforce rules, and perform complex operations on database objects.

Types of Stored Procedures in PL/SQL

Stored procedures in PL/SQL can be categorized based on different criteria:

1. Based on Parameters

- **Procedure without Parameters** – A procedure that does not take any input or return values.
- **Procedure with Parameters** – These are further divided into:
    - **IN Parameter** – Used to pass input values to the procedure.
    - **OUT Parameter** – Used to return values from the procedure.
    - **IN OUT Parameter** – Used to both pass input and return modified output.

2. Based on Execution Type

- **Standalone Procedure** – Created independently in the database schema.
- **Packaged Procedure** – Defined within a PL/SQL package.
- **Nested Procedure** – A procedure declared within another procedure.

3. Based on Functionality

- **DML Procedures** – Perform operations like INSERT, UPDATE, DELETE on database tables.
- **Transactional Procedures** – Handle transactions using COMMIT, ROLLBACK, or SAVEPOINT.
- **Validation Procedures** – Used to enforce business rules and data validation.

Stored Procedure with One Parameter

This procedure fetches an employee’s details based on their **employee ID**.

```
CREATE OR REPLACE PROCEDURE get_employee_details (
   p_emp_id IN NUMBER
)
IS
   v_name VARCHAR2(100);
   v_salary NUMBER;
BEGIN
   SELECT first_name || ' ' || last_name, salary
   INTO v_name, v_salary
   FROM employees
   WHERE employee_id = p_emp_id;

   DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_name);
   DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No employee found with ID: ' || p_emp_id);
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END get_employee_details;
/
```

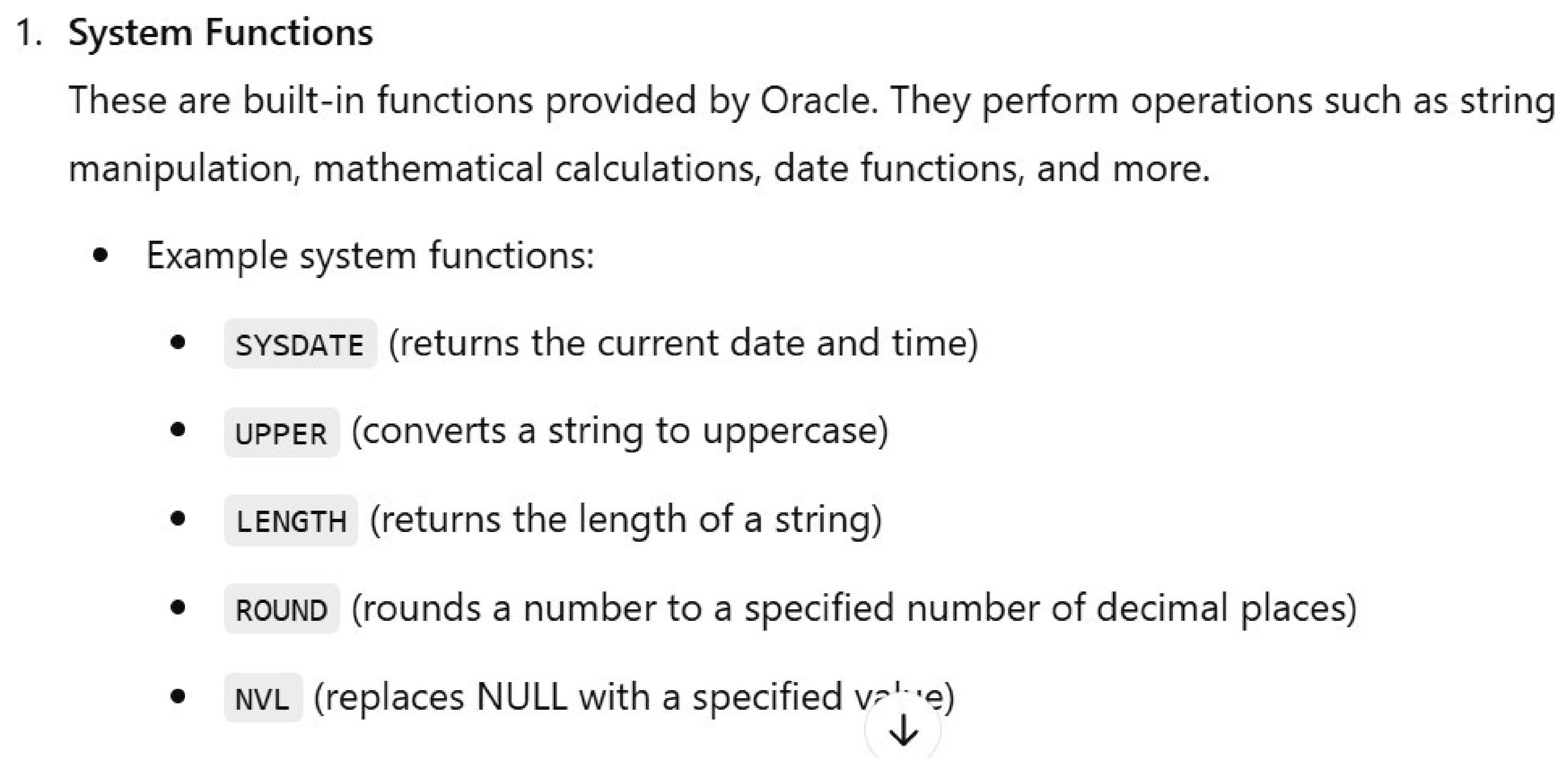
PL/SQL stored procedure with multiple parameters:

```
CREATE OR REPLACE PROCEDURE add_employee (
    p_emp_id      IN NUMBER,
    p_first_name IN VARCHAR2,
    p_last_name  IN VARCHAR2,
    p_salary      IN NUMBER,
    p_department IN NUMBER
)
IS
BEGIN
    INSERT INTO employees (employee_id, first_name, last_name, salary, department_id)
    VALUES (p_emp_id, p_first_name, p_last_name, p_salary, p_department);

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Employee ' || p_first_name || ' ' || p_last_name || ' added
successfully.');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Error: Employee ID ' || p_emp_id || ' already exists.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END add_employee;
/
```

1. System Functions in PL/SQL
PL/SQL provides built-in **system functions** that perform operations like string manipulation, mathematical calculations, date handling, and more.



1. **System Functions**

   These are built-in functions provided by Oracle. They perform operations such as string manipulation, mathematical calculations, date functions, and more.

   - Example system functions:
     - `SYSDATE` (returns the current date and time)
     - `UPPER` (converts a string to uppercase)
     - `LENGTH` (returns the length of a string)
     - `ROUND` (rounds a number to a specified number of decimal places)
     - `NVL` (replaces NULL with a specified v[illegible]e)

Example:

```sql
SELECT SYSDATE FROM DUAL;
SELECT UPPER('hello') FROM DUAL;
SELECT ROUND(15.678, 2) FROM DUAL;
```

User-Defined Function in PL/SQL

A **User-Defined Function (UDF)** in PL/SQL is a custom function created by the user to perform specific tasks and return a value. It is used to encapsulate logic that can be reused across various SQL queries or PL/SQL blocks.

```
CREATE OR REPLACE FUNCTION get_employee_salary (emp_id
NUMBER) RETURN NUMBER IS emp_salary NUMBER; BEGIN
SELECT salary INTO emp_salary FROM employees WHERE
employee_id = emp_id; RETURN emp_salary; END
get_employee_salary; /
```

Difference between Function and Procedure:

S.NO	Function	Procedure
1.	Functions always return a value after the execution of queries.	The procedure can return a value using “IN OUT” and “OUT” arguments.
2.	In SQL, those functions having a DML statement can not be called from SQL statements. But autonomous transaction functions can be called from SQL queries.	A procedure can not be called using SQL queries.
3.	Each and every time functions are compiled they provide output according to the given input.	Procedures are compiled only once but they can be called many times as needed without being compiled each time.
4.	A Function can not return multiple result sets.	A procedure is able to return multiple result sets.
5.	The function can be called using Stored Procedure.	While procedures cannot be called from function.
6.	A function used only to read data.	A procedure can be used to read and modify data.
7.	The return statement of a function returns the control and function’s result value to the calling program.	While the return statement of the procedure returns control to the calling program, it can not return the result value.
8.	The function does not support try-catch blocks.	Procedure supports try-catch blocks for error handling.