Stored Procedures

DR. MOHAMMED A. MOHAMMED

UNIVERSITY OF SULAIMANI | COLLEGE OF SCIENCE | COMPUTER DEPT.

Stored Procedures

A stored procedure is a precompiled, reusable database object that contains a set of SQL statements and procedural logic to perform a specific task or operation.

Stored procedures are stored within the database server and can be called from an application or other parts of the database like triggers, functions, or other stored procedures.

Stored procedures can accept input parameters and return output parameters or result sets. Stored procedures can be used for many purposes, such as inserting, updating, or deleting data, generating reports, or performing complex calculations.

Stored Procedures

One of the main advantages of using stored procedures is that they can improve performance by reducing network traffic between the application and the database server. This is because the SQL statements are executed on the server, rather than sending them back and forth between the client and the server. Additionally, stored procedures can be optimized and cached, leading to faster execution times.

Another advantage of using stored procedures is that they provide a layer of security. Instead of granting direct access to the tables in the database, you can grant permissions to execute specific stored procedures, which can help protect the data and prevent unauthorized access.

Overall, stored procedures can provide a more efficient and secure way to perform database operations and are a common tool used by database developers and administrators.

Creating Stored Procedure

```
CREATE OR REPLACE PROCEDURE add_employee (
   p_first_name  IN  employees.first_name%TYPE,
   p_last_name   IN  employees.last_name%TYPE,
   p_email       IN  employees.email%TYPE,
   p_phone_number IN employees.phone_number%TYPE,
   p_hire_date   IN  employees.hire_date%TYPE,
   p_job_id      IN  employees.job_id%TYPE,
   p_salary      IN  employees.salary%TYPE,
   p_commission_pct IN employees.commission_pct%TYPE,
   p_manager_id  IN  employees.manager_id%TYPE,
   p_department_id IN employees.department_id%TYPE
)
```

→ continue next slide

Creating Stored Procedure (Cont…)

```
AS
BEGIN
    INSERT INTO hr.employees (
        employee_id,
        first_name,
        last_name,
        email,
        phone_number,
        hire_date,
        job_id,
        salary,
        commission_pct,
        manager_id,
        department_id
    ) → continue next slide
```

Creating Stored Procedure(Cont…)

```
VALUES (
        hr.employees_seq.NEXTVAL,
        p_first_name,
        p_last_name,
        p_email,
        p_phone_number,
        p_hire_date,
        p_job_id,
        p_salary,
        p_commission_pct,
        p_manager_id,
        p_department_id
    );
    COMMIT;
END;
```

Execute the stored procedure

```
BEGIN
  add_employee('John', 'Doe', 'johndoe@email.com', '1234567890', SYSDATE, 'IT_PROG', 5000,
0.1, 100, 60);
END;
```

IN and OUT Parameters

```
CREATE OR REPLACE PROCEDURE calculate_area_perimeter (
   p_length IN NUMBER,
   p_width IN NUMBER,
   p_area OUT NUMBER,
   p_perimeter OUT NUMBER
)
IS
BEGIN
   p_area := p_length * p_width;
   p_perimeter := 2 * (p_length + p_width);
END;
```

IN and OUT Parameters (Cont…)

```
DECLARE
   v_length NUMBER := 5;
   v_width NUMBER := 10;
   v_area NUMBER;
   v_perimeter NUMBER;
BEGIN
   calculate_area_perimeter(v_length, v_width, v_area, v_perimeter);
   DBMS_OUTPUT.PUT_LINE('Area: ' || v_area);
   DBMS_OUTPUT.PUT_LINE('Perimeter: ' || v_perimeter);
END;
```

Conditional execution

```
CREATE OR REPLACE PROCEDURE add_employee (
   p_first_name  IN  employees.first_name%TYPE,
…
   p_is_manager  OUT INTEGER
)
AS
BEGIN
   IF (p_job_id = 'IT_PROG' OR p_job_id = 'ST_CLERK') THEN
      p_is_manager := 0;
   ELSE
      p_is_manager := 1;
   END IF;

   INSERT INTO hr.employees (
      employee_id,
…
   ) VALUES (
      hr.employees_seq.NEXTVAL,
      p_first_name,
… );   COMMIT; END;
```

Conditional execution

```
DECLARE
   is_manager INTEGER;
BEGIN
   add_employee('John', 'Doe', 'johndoe@example.com', '123-456-7890', SYSDATE, 'IT_PROG',
5000, NULL, 100, 50, is_manager);
   DBMS_OUTPUT.PUT_LINE('Is Manager: ' || is_manager);
END;
```

Update statement

```
CREATE OR REPLACE PROCEDURE update_employee_salary (
    p_employee_id  IN  employees.employee_id%TYPE,
    p_new_salary   IN  employees.salary%TYPE
)
AS
BEGIN
    UPDATE employees
    SET salary = p_new_salary
    WHERE employee_id = p_employee_id;

    COMMIT;
END;
```

Update statement (Cont.)

```
DECLARE
  v_employee_id employees.employee_id%TYPE := 100;
  v_new_salary employees.salary%TYPE := 6000;
BEGIN
  update_employee_salary(v_employee_id, v_new_salary);
END;
```

Delete statement

```
CREATE OR REPLACE PROCEDURE delete_employee(
  p_employee_id IN NUMBER
)
AS
BEGIN
  DELETE FROM employees
  WHERE employee_id = p_employee_id;
  COMMIT;
END;
```

Delete statement (Cont.)

```
DECLARE
  v_employee_id NUMBER := 100;
BEGIN
  delete_employee(v_employee_id);
END;
```

```
CREATE OR REPLACE PROCEDURE get_employee_years_of_service (
    p_employee_id IN employees.employee_id%TYPE,
    p_years_of_service OUT INTEGER
)
AS
    v_employee employees%ROWTYPE;
    v_hire_date DATE;
BEGIN
    -- Get the employee record
    SELECT *
    INTO v_employee
    FROM employees
    WHERE employee_id = p_employee_id;

    -- Calculate the years of service
    v_hire_date := to_date(v_employee.hire_date, 'DD-MON-RR');
    p_years_of_service := round((sysdate - v_hire_date) / 365.25, 2);

    -- Print the employee information and years of service
    dbms_output.put_line('Employee ID: ' || v_employee.employee_id);
    dbms_output.put_line('Employee Name: ' || v_employee.first_name || ' ' || v_employee.last_name);
    dbms_output.put_line('Years of Service: ' || p_years_of_service);
END;
```

```
DECLARE
    v_years_of_service INTEGER;
BEGIN
    get_employee_years_of_service(101, v_years_of_service);
END;
```

End of Stored Procedures