Задание семинара 8
Лим В. БПИ225

1. Create Stored procedure.
In this exercise, create a procedure to add a new job into the JOBS table.
a. Create a stored procedure called NEW_JOB to enter a new order into the JOBS table. The procedure should accept three parameters. The first and second parameters supply a job ID and a job title. The third parameter supplies the minimum salary. Use the maximum salary for the new job as twice the minimum salary supplied for the job ID.
b. Invoke the procedure to add a new job with job ID 'SY_ANAL', job title 'System Analyst', and minimum salary 6,000.

Решение:

```
--1.

CREATE OR REPLACE PROCEDURE NEW_JOB(
   p_job_id IN VARCHAR,
   p_job_title IN VARCHAR,
   p_min_salary IN INTEGER
)
LANGUAGE plpgsql
AS
$$
BEGIN
   IF NOT EXISTS (SELECT 1 FROM jobs WHERE job_id = p_job_id) THEN
       INSERT INTO jobs (job_id, job_title, min_salary, max_salary)
       VALUES (p_job_id, p_job_title, p_min_salary, p_min_salary * 2);
   ELSE
       RAISE NOTICE 'Job ID % already exists.', p_job_id;
   END IF;
END;
$$;
```

2. Create Stored Procedure.
In this exercise, create a program to add a new row to the JOB_HISTORY table for an existing employee.
a. Create a stored procedure called ADD_JOB_HIST to add a new row into the JOB_HISTORY table for an employee who is changing his job to the new job ID ('SY_ANAL') that you created in exercise 1b.
The procedure should provide two parameters: one for the employee ID who is changing the job, and the second for the new job ID. Read the employee ID from the EMPLOYEES table and insert it into the JOB_HISTORY table. Make the hire date of this employee as the start date and today's date as the end date for this row in the JOB_HISTORY table.
Change the hire date of this employee in the EMPLOYEES table to today's date. Update the job ID of this employee to the job ID passed as parameter (use the 'SY_ANAL' job ID) and salary equal to the minimum salary for that job ID plus 500.
Note: Include exception handling to handle an attempt to insert a nonexistent employee.
. Disable all triggers on the EMPLOYEES, JOBS, and JOB_HISTORY tables before invoking the ADD_JOB_HIST procedure.
. Execute the procedure with employee ID 106 and job ID 'SY_ANAL' as parameters.
. Query the JOB_HISTORY and EMPLOYEES tables to view your changes for employee 106, and then commit the changes.
. Reenable the triggers on the EMPLOYEES, JOBS, and JOB_HISTORY tables.

Решение:
```
--2.

CREATE OR REPLACE PROCEDURE ADD_JOB_HIST(
   p_employee_id INTEGER,
   p_new_job_id VARCHAR(10)
)
LANGUAGE plpgsql
AS $$
DECLARE
   v_min_salary INTEGER;
BEGIN
   SELECT min_salary INTO v_min_salary
   FROM jobs
   WHERE job_id = p_new_job_id;

   INSERT INTO job_history (employee_id, start_date, end_date, job_id,
department_id)
   VALUES (p_employee_id, (SELECT hire_date FROM employees WHERE
employee_id = p_employee_id), CURRENT_DATE, p_new_job_id,
           (SELECT department_id FROM employees WHERE employee_id =
p_employee_id));

   UPDATE employees
   SET job_id = p_new_job_id,
       salary = v_min_salary + 500,
       hire_date = CURRENT_DATE
   WHERE employee_id = p_employee_id;
```

```
EXCEPTION
   WHEN NO_DATA_FOUND THEN
       RAISE EXCEPTION 'Employee not found';
END;
$$;

CALL ADD_JOB_HIST(106, 'SY_ANAL');
```

3. Create Stored Procedure. In this exercise, create a program to update the minimum and maximum salaries for a job in the JOBS table.
a. Create a stored procedure called UPD_JOBSAL to update the minimum and maximum salaries for a specific job ID in the JOBS table. The procedure should provide three parameters: the job ID, a new minimum salary, and a new maximum salary. Add exception handling to account for an invalid job ID in the JOBS table. Raise an exception if the maximum salary supplied is less than the minimum salary. Provide a message that will be displayed if the row in the JOBS table is locked.
Hint: The resource locked/busy error number is –54.
b. Execute the UPD_JOBSAL procedure by using a job ID of 'SY_ANAL', a minimum salary of 7000, and a maximum salary of 140.
c. Disable triggers on the EMPLOYEES and JOBS tables.
d. Execute the UPD_JOBSAL procedure using a job ID of 'SY_ANAL', a minimum salary of 7000, and a maximum salary of 14000.
e. Query the JOBS table to view your changes, and then commit the changes.
. Enable the triggers on the EMPLOYEES and JOBS tables.

Решение:

```
--3.

CREATE OR REPLACE PROCEDURE UPD_JOBSAL(
   p_job_id VARCHAR(10),
   p_new_min_salary INTEGER,
   p_new_max_salary INTEGER
)
LANGUAGE plpgsql
AS $$
BEGIN
   IF p_new_max_salary < p_new_min_salary THEN
       RAISE EXCEPTION 'Maximum salary cannot be less than minimum
salary';
   END IF;

   UPDATE jobs
   SET min_salary = p_new_min_salary,
       max_salary = p_new_max_salary
   WHERE job_id = p_job_id;

EXCEPTION
   WHEN OTHERS THEN
       RAISE NOTICE 'Error while updating the salary for job_id: %',
p_job_id;
END;
$$;

CALL UPD_JOBSAL('SY_ANAL', 7000, 14000);
```

4. Create Stored Function. Create a subprogram to retrieve the number of years of service for a specific employee.
a. Create a stored function called GET_YEARS_SERVICE to retrieve the total number of years of service for a specific employee. The function should accept the employee ID as a parameter and return the number of years of service. Add error handling to account for an invalid employee ID.
b. Invoke the GET_YEARS_SERVICE function in a call to DBMS_OUTPUT.PUT_LINE for an employee with ID 999.
c. Display the number of years of service for employee 106 with DBMS_OUTPUT.PUT_LINEinvoking the GET_YEARS_SERVICE function.
d. Query the JOB_HISTORY and EMPLOYEES tables for the specified employee to verify that the modifications are accurate.Note: The values represented in the results on this page may differ from those you get when you run these queries.

Решение:

```
--4.

CREATE OR REPLACE FUNCTION GET_YEARS_SERVICE(p_employee_id INT)
RETURNS INT AS
$$
DECLARE
   v_years_service INT;
BEGIN
   SELECT EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
hire_date)
   INTO v_years_service
   FROM employees
   WHERE employee_id = p_employee_id;

   IF v_years_service IS NULL THEN
       RAISE EXCEPTION 'Employee with ID % does not exist',
p_employee_id;
   END IF;

   RETURN v_years_service;
END;
$$ LANGUAGE plpgsql;
```

5. Create Stored Function. In this exercise, create a program to retrieve the number of different jobs that an employee worked on during his or her service.
a. Create a stored function called GET_JOB_COUNT to retrieve the total number of different jobs on which an employee worked.
The function should accept the employee ID in a parameter, and return the number of different jobs that the employee worked on until now, including the present job. Add exception handling to account for an invalid employee ID.Hint: Use the distinct job IDs from the JOB_HISTORY table, and exclude the current job ID, if it is one of the job IDs on which the employee has already worked. Write a UNION of two queries and count the rows retrieved into a PL/SQL table. Use a FETCH with BULK COLLECT INTO to obtain the unique jobs for the employee.
b. Invoke the function for an employee with ID 176.

Решение:

```
--5.

CREATE OR REPLACE FUNCTION GET_JOB_COUNT(p_employee_id INTEGER)
RETURNS INTEGER AS $$
DECLARE
   v_job_count INTEGER;
BEGIN
   SELECT COUNT(DISTINCT job_id) INTO v_job_count
   FROM job_history
   WHERE employee_id = p_employee_id
   UNION
   SELECT COUNT(DISTINCT job_id)
   FROM employees
   WHERE employee_id = p_employee_id;

   RETURN v_job_count;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
       RAISE EXCEPTION 'Employee not found';
END;

$$ LANGUAGE plpgsql;

SELECT GET_JOB_COUNT(176);
```

6. Create Trigger.
In this exercise, create a trigger to ensure that the minimum and maximum salaries of a job are never modified such that the salary of an existing employee with that job ID is outside the new range specified for the job.
a. Create a trigger called CHECK_SAL_RANGE that is fired before every row that is updated in the MIN_SALARY and MAX_SALARY columns in the JOBS table. For any minimum or maximum salary value that is changed, check whether the salary of any existing employee with that job ID in the EMPLOYEES table falls within the new range of salaries specified for this job ID. Include exception handling to cover a salary range change that affects the record of any existing employee.
b. Test the trigger using the SY_ANAL job, setting the new minimum salary to 5000 and the new maximum salary to 7000. Before you make the change, write a query to display the current salary range for the SY_ANAL job ID, and another query to display the employee ID, last name, and salary for the same job ID. After the update, query the change (if any) to the JOBS table for the specified job ID.
c. Using the job SY_ANAL, set the new minimum salary to 7000 and the new maximum salary to 18000. Explain the results.

Решение:

```
--6.

CREATE OR REPLACE FUNCTION check_salary_range()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.min_salary > NEW.max_salary THEN
       RAISE EXCEPTION 'Minimum salary cannot be greater than maximum
salary';
   END IF;

   IF EXISTS (SELECT 1 FROM employees WHERE job_id = OLD.job_id AND
(salary < NEW.min_salary OR salary > NEW.max_salary)) THEN
       RAISE EXCEPTION 'Employee salary falls outside new salary
range';
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER CHECK_SAL_RANGE
BEFORE UPDATE ON jobs
FOR EACH ROW
EXECUTE FUNCTION check_salary_range();

UPDATE jobs
SET min_salary = 5000, max_salary = 7000
WHERE job_id = 'SY_ANAL';
```