

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 7

з дисципліни «Професійне використання SQL та PL/SQL для
СУБД Oracle»

Тема: ««Відпрацювання конструкцій мови PL\SQL Oracle – Stored
Procedures; Functions.»»

Виконали:

студент групи

ІМ-13 Первєєв Євгеній

Олексійович

Перевірила:

Ульяницька

К.О.

Київ 2024

Тема: «Відпрацювання конструкцій мови PL\SQL Oracle – Stored Procedures; Functions.»

Мета: Навчитись створювати зберезувані процедури та функції, розуміти різницю та вміти застосовувати дані програмні одиниці на практиці.

Хід роботи

1. Маскування чутливих даних за допомогою зберезуваної процедури.

```
CREATE TABLE Persons (  
    ID INT NOT NULL PRIMARY KEY,  
    FirstName VARCHAR(50) NOT NULL,  
    MiddleName VARCHAR(50),  
    LastName VARCHAR(50) NOT NULL,  
    EmailAddress VARCHAR(100) NOT NULL,  
    PhoneNumber VARCHAR(15) NOT NULL  
);
```

```
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,  
PhoneNumber)  
VALUES (1, 'Іван', 'Петрович', 'Петров', 'ivan@example.com', '+380971234567');
```

```
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,  
PhoneNumber)  
VALUES (2, 'Марія', 'Олексіївна', 'Сидоренко', 'maria@example.com', '+380971234568');
```

```
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,  
PhoneNumber)  
VALUES (3, 'Олександр', 'Ігорович', 'Ковальчук', 'oleksandr@example.com',  
'+380971234569');
```

```
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,  
PhoneNumber)  
VALUES (4, 'Тетяна', 'Сергіївна', 'Павленко', 'tetiana@example.com', '+380971234570');
```

```
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,  
PhoneNumber)  
VALUES (5, 'Андрій', 'Олександрович', 'Мельник', 'andriy@example.com',  
'+380971234571');
```

```
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,  
PhoneNumber)  
VALUES (6, 'Ольга', 'Віталіївна', 'Іваненко', 'olga@example.com', '+380971234572');
```

```
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,
```

```

PhoneNumber)
VALUES (7, 'Володимир', 'Миколайович', 'Бондаренко', 'volodymyr@example.com',
'+380971234573');

```

```

INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,
PhoneNumber)
VALUES (9, 'Ігор', 'Васильович', 'Григоренко', 'igor@example.com', '+380971234575');

```

```

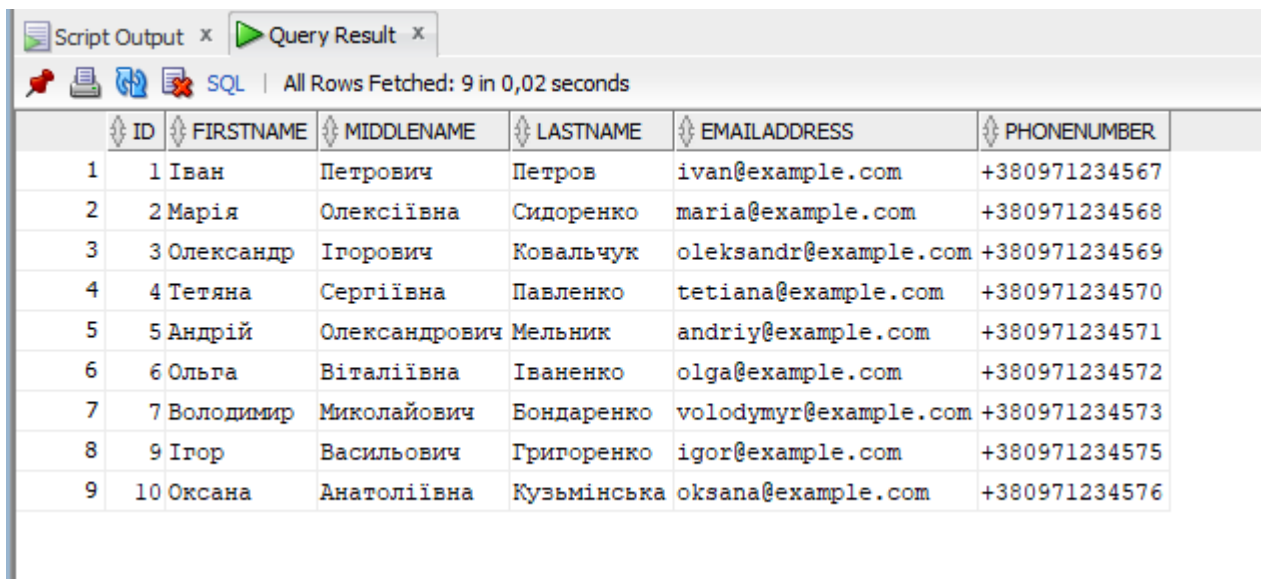
INSERT INTO Persons (ID, FirstName, MiddleName, LastName, EmailAddress,
PhoneNumber)
VALUES (10, 'Оксана', 'Анатоліївна', 'Кузьмінська', 'oksana@example.com',
'+380971234576');

```

```

CREATE OR REPLACE PROCEDURE MaskSensitiveData AS
BEGIN
    UPDATE Persons
    SET FirstName = 'Xxxxxxx',
        MiddleName = NULL,
        LastName = 'Xxxxxxx',
        EmailAddress = 'xxxxxxxx@xxxx.yyy',
        PhoneNumber = '+380888888888';
END MaskSensitiveData;

```



The screenshot shows a SQL query result window with a table containing 9 rows of data. The table has columns for ID, FIRSTNAME, MIDDLENAME, LASTNAME, EMAILADDRESS, and PHONENUMBER. The data is as follows:

	ID	FIRSTNAME	MIDDLENAME	LASTNAME	EMAILADDRESS	PHONENUMBER
1	1	Іван	Петрович	Петров	ivan@example.com	+380971234567
2	2	Марія	Олексіївна	Сидоренко	maria@example.com	+380971234568
3	3	Олександр	Ігорович	Ковальчук	oleksandr@example.com	+380971234569
4	4	Тетяна	Сергіївна	Павленко	tetiana@example.com	+380971234570
5	5	Андрій	Олександрович	Мельник	andriy@example.com	+380971234571
6	6	Ольга	Віталіївна	Іваненко	olga@example.com	+380971234572
7	7	Володимир	Миколайович	Бондаренко	volodymyr@example.com	+380971234573
8	9	Ігор	Васильович	Григоренко	igor@example.com	+380971234575
9	10	Оксана	Анатоліївна	Кузьмінська	oksana@example.com	+380971234576

	ID	FIRSTNAME	MIDDLENAME	LASTNAME	EMAILADDRESS	PHONENUMBER
1	1	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
2	2	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
3	3	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
4	4	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
5	5	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
6	6	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
7	7	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
8	9	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888
9	10	Xxxxxxx	(null)	Xxxxxxx	xxxxxxxx@xxx.yyy	+380888888888

2. Оновлення даних, використовуючи функцію (FUNCTION).

- Звіт по тестовій бд

```
CREATE OR REPLACE FUNCTION CheckJobHistory(
```

```
    emp_id IN NUMBER
```

```
) RETURN BOOLEAN IS
```

```
    v_count NUMBER;
```

```
BEGIN
```

```
    SELECT COUNT(*)
```

```
    INTO v_count
```

```
    FROM job_history
```

```
    WHERE EMPLOYEE_ID = emp_id;
```

```
    IF v_count > 0 THEN
```

```
        RETURN TRUE;
```

```
    ELSE
```

```
        RETURN FALSE;
```

```
    END IF;
```

```
EXCEPTION
```

```
    WHEN OTHERS THEN
```

```
        RETURN FALSE;
```

END;

CREATE OR REPLACE FUNCTION GetJobStatus(

emp_id IN NUMBER

) RETURN NUMBER IS

v_job_status NUMBER;

hire_date DATE;

BEGIN

IF CheckJobHistory(emp_id) THEN

v_job_status := 2;

ELSE

SELECT e.hire_date

INTO hire_date

FROM employees e

WHERE e.employee_id = emp_id;

IF hire_date > TO_DATE('01-01-2006', 'DD-MM-YYYY') THEN

v_job_status := 1;

ELSE

v_job_status := 0;

END IF;

END IF;

RETURN v_job_status;

END;

```

CREATE OR REPLACE FUNCTION GetRegionStatus(emp_id NUMBER)
RETURN INT
AS
    region_status INT;
    region VARCHAR2(50);
    salary INT;
BEGIN
    SELECT r.region_name
    INTO region
    FROM regions r
    JOIN countries c ON r.region_id = c.region_id
    JOIN locations l ON c.country_id = l.country_id
    JOIN departments d ON l.location_id = d.location_id
    JOIN employees e ON d.department_id = e.department_id
    WHERE e.employee_id = emp_id;

    SELECT e.salary
    INTO salary
    FROM employees e
    WHERE e.employee_id = emp_id;

    IF region = 'Europe' AND salary < 6000 THEN
        region_status := 1;
    ELSIF region = 'Asia' AND salary > 10000 THEN
        region_status := 2;
    ELSE
        region_status := 3;
    END IF;

```

```

    RETURN region_status;
END;

CREATE OR REPLACE FUNCTION GenerateEmployeeReport RETURN
SYS_REFCURSOR IS
    rep_cursor SYS_REFCURSOR;
BEGIN
    OPEN rep_cursor FOR
        SELECT
            e.first_name || ' ' || e.last_name AS EmployeeName,
            e.email AS Email,
            e.phone_number AS PhoneNumber,
            d.department_name AS Department,
            GetJobStatus(e.employee_id) AS JobStatus,
            GetRegionStatus(e.employee_id) AS RegionStatus
        FROM
            employees e
        JOIN
            departments d ON e.department_id = d.department_id;

    RETURN rep_cursor;
END;

```

Hello, world!

Employee: Jennifer Whalen, Email: JWHALEN, Phone: 515.123.4444, Department: Administration, JobStatus: 2, RegionStatus: 3
Employee: Michael Hartstein, Email: MHARTSTE, Phone: 515.123.5555, Department: Marketing, JobStatus: 2, RegionStatus: 3
Employee: Pat Fay, Email: PFAY, Phone: 603.123.6666, Department: Marketing, JobStatus: 0, RegionStatus: 3
Employee: Den Raphaely, Email: DRAPHEAL, Phone: 515.127.4561, Department: Purchasing, JobStatus: 2, RegionStatus: 3
Employee: Alexander Khoo, Email: AKHOO, Phone: 515.127.4562, Department: Purchasing, JobStatus: 0, RegionStatus: 3
Employee: Shelli Baida, Email: SBAIDA, Phone: 515.127.4563, Department: Purchasing, JobStatus: 0, RegionStatus: 3
Employee: Sigal Tobias, Email: STOBIAS, Phone: 515.127.4564, Department: Purchasing, JobStatus: 0, RegionStatus: 3
Employee: Guy Himuro, Email: GHIMURO, Phone: 515.127.4565, Department: Purchasing, JobStatus: 1, RegionStatus: 3
Employee: Karen Colmenares, Email: KCOLMENA, Phone: 515.127.4566, Department: Purchasing, JobStatus: 1, RegionStatus: 3
Employee: Susan Mavris, Email: SMAVRIS, Phone: 515.123.7777, Department: Human Resources, JobStatus: 0, RegionStatus: 3
Employee: Matthew Weiss, Email: MWEISS, Phone: 650.123.1234, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Adam Fripp, Email: AFRIPP, Phone: 650.123.2234, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Payam Kaufling, Email: PKAUFLIN, Phone: 650.123.3234, Department: Shipping, JobStatus: 2, RegionStatus: 3
Employee: Shanta Vollman, Email: SVOLLMAN, Phone: 650.123.4234, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Kevin Mourgos, Email: KMOURGOS, Phone: 650.123.5234, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Julia Nayer, Email: JNAYER, Phone: 650.124.1214, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Irene Mikilineni, Email: IMIKKILI, Phone: 650.124.1224, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: James Landry, Email: JLANDRY, Phone: 650.124.1334, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Steven Markle, Email: SMARKLE, Phone: 650.124.1434, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Laura Bissot, Email: LBISSOT, Phone: 650.124.5234, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Mozhe Atkinson, Email: MATKINSO, Phone: 650.124.6234, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: James Marlow, Email: JAMLOW, Phone: 650.124.7234, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: TJ Olson, Email: TJOLSON, Phone: 650.124.8234, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Jason Mallin, Email: JMALLIN, Phone: 650.127.1934, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Michael Rogers, Email: MROGERS, Phone: 650.127.1834, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Ki Gee, Email: KGEE, Phone: 650.127.1734, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Hazel Philtanker, Email: HPHILTAN, Phone: 650.127.1634, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Renske Ladwig, Email: RLADWIG, Phone: 650.121.1234, Department: Shipping, JobStatus: 0, RegionStatus: 3

Employee: Girard Geoni, Email: GGEONI, Phone: 650.507.9879, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Nandita Sarchand, Email: NSARCHAN, Phone: 650.509.1876, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Alexis Bull, Email: ABULL, Phone: 650.509.2876, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Julia Dellinger, Email: JDELLING, Phone: 650.509.3876, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Anthony Cabrio, Email: ACABRIO, Phone: 650.509.4876, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Kelly Chung, Email: KCHUNG, Phone: 650.505.1876, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Jennifer Dilly, Email: JDILLY, Phone: 650.505.2876, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Timothy Gates, Email: TGATES, Phone: 650.505.3876, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Randall Perkins, Email: RPERKINS, Phone: 650.505.4876, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Sarah Bell, Email: SBELL, Phone: 650.501.1876, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Britney Everett, Email: BEVERETT, Phone: 650.501.2876, Department: Shipping, JobStatus: 0, RegionStatus: 3
Employee: Samuel McCain, Email: SMCCAIN, Phone: 650.501.3876, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Vance Jones, Email: VJONES, Phone: 650.501.4876, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Alana Walsh, Email: AWALSH, Phone: 650.507.9811, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Kevin Feeney, Email: KFEENEY, Phone: 650.507.9822, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Donald OConnell, Email: DOCONNEL, Phone: 650.507.9833, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Douglas Grant, Email: DGRANT, Phone: 650.507.9844, Department: Shipping, JobStatus: 1, RegionStatus: 3
Employee: Alexander Hunold, Email: AHUNOLD, Phone: 590.423.4567, Department: IT, JobStatus: 1, RegionStatus: 3
Employee: Bruce Ernst, Email: BERNST, Phone: 590.423.4568, Department: IT, JobStatus: 1, RegionStatus: 3
Employee: David Austin, Email: DAUSTIN, Phone: 590.423.4569, Department: IT, JobStatus: 0, RegionStatus: 3
Employee: Valli Pataballa, Email: VPATABAL, Phone: 590.423.4560, Department: IT, JobStatus: 1, RegionStatus: 3
Employee: Diana Lorentz, Email: DLORENTZ, Phone: 590.423.5567, Department: IT, JobStatus: 1, RegionStatus: 3
Employee: Hermann Baer, Email: HBAER, Phone: 515.123.8888, Department: Public Relations, JobStatus: 0, RegionStatus: 3
Employee: John Russell, Email: JRUSSEL, Phone: 011.44.1344.429268, Department: Sales, JobStatus: 0, RegionStatus: 3
Employee: Karen Partners, Email: KPARTNER, Phone: 011.44.1344.467268, Department: Sales, JobStatus: 0, RegionStatus: 3
Employee: Alberto Errazuriz, Email: AERRAZUR, Phone: 011.44.1344.429278, Department: Sales, JobStatus: 0, RegionStatus: 3
Employee: Gerald Cambraut, Email: GCAMBRAU, Phone: 011.44.1344.619268, Department: Sales, JobStatus: 1, RegionStatus: 3
Employee: Eleni Zlotkey, Email: EZLOTKEY, Phone: 011.44.1344.429018, Department: Sales, JobStatus: 1, RegionStatus: 3
Employee: Peter Tucker, Email: PTUCKER, Phone: 011.44.1344.129268, Department: Sales, JobStatus: 0, RegionStatus: 3

-Звіт по бд за варіантом

CREATE OR REPLACE TYPE MedicineReport AS OBJECT (

MedicineName VARCHAR2(20),


```
ApplicationMethod VARCHAR2(20),  
PreparationMethod VARCHAR2(20),  
RecipeName VARCHAR2(20),  
RecipeIngredients VARCHAR2(20),  
PreparationTime NUMBER  
);
```

```
ALTER TYPE MedicineReport MODIFY ATTRIBUTE (MedicineName  
VARCHAR2(100)) CASCADE;
```

```
ALTER TYPE MedicineReport MODIFY ATTRIBUTE (ApplicationMethod  
VARCHAR2(100)) CASCADE;
```

```
ALTER TYPE MedicineReport MODIFY ATTRIBUTE (PreparationMethod  
VARCHAR2(100)) CASCADE;
```

```
ALTER TYPE MedicineReport MODIFY ATTRIBUTE (RecipeName  
VARCHAR2(100)) CASCADE;
```

```
ALTER TYPE MedicineReport MODIFY ATTRIBUTE (RecipeIngredients  
VARCHAR2(100)) CASCADE;
```

```
CREATE OR REPLACE TYPE MedicineReportTable AS TABLE OF  
MedicineReport;
```

```
CREATE OR REPLACE FUNCTION GenerateReport RETURN  
MedicineReportTable IS
```

```
    rep_table MedicineReportTable := MedicineReportTable();  
BEGIN  
    FOR rec IN (SELECT  
        fm.Name AS MedicineName,  
        am.Name AS ApplicationMethod,  
        pm.Name AS PreparationMethod,
```

```

        r.Name AS RecipeName,
        r.Ingredients AS RecipeIngredients,
        um.P_Time AS PreparationTime
FROM
    Finished_Medicines fm
JOIN
    Application_Methods am ON fm.A_Method_ID = am.ID
LEFT JOIN
    Unfinished_Medicines um ON fm.ID = um.ID
LEFT JOIN
    Preparation_Methods pm ON um.P_Method_ID = pm.ID
LEFT JOIN
    Recipes r ON um.Recipe_ID = r.ID)
LOOP
    rep_table.EXTEND;

    rep_table(rep_table.LAST) := MedicineReport(rec.MedicineName,
rec.ApplicationMethod, rec.PreparationMethod, rec.RecipeName,
rec.RecipeIngredients, rec.PreparationTime);

    END LOOP;





    RETURN rep_table;
END;

SELECT * FROM TABLE(GenerateReport());

```

Script Output x

Query Result x

    SQL | All Rows Fetched: 11 in 0,054 seconds

	MEDICINENAME	APPLICATIONMETHOD	PREPARATIONMETHOD	RECIPENAME	RECIPEINGREDIENTS	PREPARATIONTIME
1	Ношпа	Зовнішнє заст.	Змішування	Рецепт1	Інгредієнт1	15
2	Мазь Вишневського	Внутрішнє заст.	Зміш-відст-фільтр	Рецепт2	Інгредієнт2	10
3	Аспірин	Зовнішнє заст.	Зміш-відст-фільтр	Рецепт3	Інгредієнт3	30
4	Trenbolon	Зовнішнє заст.	(null)	(null)	(null)	(null)
5	Sustanon	Внутрішнє заст.	(null)	(null)	(null)	(null)
6	Детралекс	Зовнішнє заст.	(null)	(null)	(null)	(null)
7	null	Для змішування	(null)	(null)	(null)	(null)
8	Вазелін	Внутрішнє заст.	(null)	(null)	(null)	(null)
9	Аспаркам	Зовнішнє заст.	(null)	(null)	(null)	(null)
10	null	Внутрішнє заст.	(null)	(null)	(null)	(null)
11	(null)	Зовнішнє заст.	(null)	(null)	(null)	(null)

Висновок

Під час виконання лабораторної роботи з теми "Відпрацювання конструкцій мови PL/SQL Oracle – Stored Procedures; Functions" я отримав цінний досвід у створенні та використанні збережуваних процедур та функцій у середовищі бази даних Oracle. Ретельне вивчення концепцій цих програмних одиниць дозволило мені краще розуміти різницю між ними, а саме, що процедури використовуються для виконання дій без повернення значень, тоді як функції повертають результати своєї роботи. Цей досвід дозволить мені в майбутньому створювати більш ефективні програмні рішення, які використовують базу даних Oracle, та впроваджувати їх у практичні сценарії