Лабораторна робота 7

Тема: Проектування вкладених DQL, DML-запитів

Завдання

За кожним етапом створити файл-скрипт *N.sql*, де *N* – номер етапу, в який під час виконання завдань вказувати:

1) умова завдання у вигляді багаторядкового коментаря

2) *SQL*-команда

3) рядки із відповіддю на запит (для *SELECT*-команд) або реакція СУБД (для помилки) у вигляді багаторядкового коментаря

Для отримання рядків із відповіддю на *SQL*-команда зручно використовувати *SQLPlus*.

Етап 1. DQL із вкладеними запитами.

1. Виконання простих однорядкових підзапитів із екві-з'єднанням або тета-з'єднанням.

На рисунку 1 показані приклади простих однорядкових підзапитів з екві-з'єднанням або тета-з'єднанням.

```
-- Приклад 1. Отримати список прізвищ співробітників,
-- зарплата яких більше зарплати співробітника з номером 7566
SELECT ENAME
FROM   EMP
WHERE  SAL >
            (
            SELECT SAL
              FROM EMP WHERE EMPNO = 7566
            );

-- Приклад 2 Виконання однорядкових підзапитів
     SELECT   ename, job  FROM emp
       WHERE    job =
            (SELECT job FROM emp WHERE empno = 7369)
       AND      sal >
            (SELECT sal
            FROM emp WHERE  empno = 7876);
```

Рис. 1 – Приклади простих однорядкових підзапитів з екві-з'єднанням або тета-з'єднанням

Створіть схожий запит, використовуючи одну або дві таблиці вашої бази даних.

2. Використання агрегатних функцій у підзапитах.

На рисунку 2 показаний приклад використання агрегатних функцій у підзапитах.

```
-- Приклад 3 Використання агрегатних функцій у підзапитах
-- Отримати список співробітників, зарплата яких мінімальна
SELECT ENAME, JOB, SAL
FROM EMP
WHERE SAL =
          (SELECT MIN(SAL) FROM EMP);
```

Рис. 2 – Приклад використання агрегатних функцій у підзапитах

Створіть схожий запит, використовуючи одну або дві таблиці вашої бази даних.

3. Пропозиція HAVING із підзапитами.

На рисунку 3 показаний приклад використання пропозиції HAVING у підзапиті

```
-- Приклад 4 Використання пропозиції HAVING у підзапиті
-- Отримати список підрозділів, у яких середня зарплата співробітників
-- більше середньої зарплати співробітників у всій компанії
SELECT DEPTNO, AVG(SAL)
FROM EMP
GROUP BY DEPTNO
HAVING AVG(SAL) >
            (SELECT AVG(SAL) FROM EMP);
```

Рис. 3 – Приклад використання пропозиції HAVING у підзапиті

Створіть схожий запит, використовуючи одну або дві таблиці вашої бази даних.

4. Виконання багаторядкових підзапитів

На рисунку 4 показаний приклад використання операторів ALL, ANY у багаторядкових підзапитах.

```
-- Приклад 7 Використання оператора ANY у багаторядкових підзапитах
-- Отримати список співробітників, які не є клерками,
-- а зарплата яких менша за будь-яку зарплату, яку отримують клерки
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE   SAL < ANY
                (SELECT SAL FROM EMP
                    WHERE JOB = 'CLERK')
        AND JOB <> 'CLERK';

-- Приклад 8 Інший запит з урахуванням того,
-- що "< ANY" означає "менше максимального"
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE   SAL <
                (SELECT MAX(SAL) FROM EMP
                    WHERE JOB = 'CLERK')
        AND JOB <> 'CLERK';

-- Приклад 9 Використання оператора ALL у багаторядкових підзапитах
-- Отримати список співробітників, зарплата яких більша
-- максимальної середньої зарплати у різних підрозділах
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE SAL > ALL
                (SELECT AVG(SAL)
                    FROM EMP GROUP BY DEPTNO);

-- Приклад 10 Інший запит з урахуванням того,
-- що ">ALL" означає "більше максимального"
SELECT  EMPNO, ENAME, JOB
FROM EMP
WHERE SAL >
            (SELECT MAX(AVG(SAL))
                FROM EMP GROUP BY DEPTNO);
```

Рис. 4 – Приклад використання операторів ALL, ANY у багаторядкових підзапитах

Створіть схожий запит, використовуючи одну або дві таблиці вашої бази даних.

5. **Використання оператора WITH для структуризації запиту**

На рисунку 5 показаний приклад використання підзапитів в операторі WITH.

```
-- Приклад 12.2 Використання підзапитів в операторі WITH.
-- Вибрати співробітників з більшою зарплатою,
-- ніж середня зарплата у співробітників у їх підрозділах
WITH
AVG_SAL AS
    (
        SELECT DEPTNO, AVG(SAL) SALAVG
        FROM EMP
        GROUP BY DEPTNO
    )
SELECT A.ENAME, A.SAL, A.DEPTNO, B.SALAVG
FROM EMP A, AVG_SAL B
    WHERE   A.DEPTNO = B.DEPTNO
            AND A.SAL > B.SALAVG;
```

Рис. 5 – приклад використання підзапитів в операторі WITH

Створіть схожий запит, використовуючи одну або дві таблиці вашої бази даних.

6. **Використання кореляційних підзапитів**

На рисунку 6 показаний приклад використання кореляційних підзапитів з оператором EXISTS.

```
-- Приклад 14 Отримати список підроділів, в яких працюють співробітники.
-- Використання оператора EXISTS
SELECT D.DNAME
    FROM DEPT D
    WHERE EXISTS (
            SELECT E.DEPTNO
            FROM EMP E
            WHERE E.DEPTNO = D.DEPTNO);

-- Приклад 15 Отримати список співробітників, які не мають підлеглих
SELECT EMPLOYEE.ENAME
FROM EMP EMPLOYEE
WHERE NOT EXISTS
            (SELECT MANAGER.MGR
            FROM EMP MANAGER
            WHERE MGR = EMPLOYEE.EMPNO);
```

Рис. 6 – Приклад використання кореляційних підзапитів з оператором EXISTS

Етап 2. DML із вкладеними запитами.

1. Багатотабличне внесення даних

На рисунку 7 показаний приклад використання багатотабличного внесення даних.

```
-- Приклад 1 Використовуючи один INSERT-оператор, зареєструвати нового співробітника
-- у новому підрозділі, який також має бути зареєстрований.
-- Назва підрозділу = 'Odessa Office' і розташований в тому самому місті,
-- що і підрозділ його керівника.
-- Співробітник: ім'я = 'IVANOV', посада = 'STUDENT', дата зарахування = поточна дата,
-- зарплата = 0, премія = 0, керівником є співробітник з ім'ям KING.

CREATE SEQUENCE DEPTNO START WITH 50 INCREMENT BY 10;
CREATE SEQUENCE EMPNO START WITH 8000;
INSERT ALL
    INTO DEPT (DEPTNO, DNAME, LOCNO)
        VALUES (DEPTNO.NEXTVAL, 'ODESSA OFFICE', LOCNO)
    INTO EMP (EMPNO, ENAME, JOB,
            MGR, HIREDATE, SAL, COMM, DEPTNO)
        VALUES (EMPNO.NEXTVAL, 'IVANOV', 'STUDENT',
            EMPNO, SYSDATE, 0, 0, DEPTNO.CURRVAL)
SELECT E.EMPNO, D.LOCNO
FROM EMP E, DEPT D
WHERE E.ENAME = 'KING'
        AND E.DEPTNO = D.DEPTNO;
```

Рис. 7 –Приклад використання багатотабличного внесення даних

Створіть один схожий запит, виконавши одночасне внесення до двох таблиць вашої БД.

2. Використання багатостовпцевих підзапитів при зміні даних

На рисунку 8 показаний приклад використання багатостовпцевих підзапитів при зміні даних.

```
-- Приклад 3 Зміна посади та підрозділу співробітника
-- з номером = 7698 на такі ж, як у співробітника з номером = 7499.
UPDATE  emp
  SET   (job, deptno) =
                    (SELECT job, deptno
                            FROM    emp
                            WHERE   empno = 7499)
  WHERE    empno = 7698;
```

Рис. 8 – Приклад використання багатостовпцевих підзапитів при зміні даних

Створіть один схожий запит на зміну двох колонок однієї таблиці вашої БД, використовуючи багатостовпцевий підзапит.

3. Видалення рядків із використанням кореляційних підзапитів.

На рисунку 9 показаний приклад використання кореляційних підзапитів під час видалення рядків.

```
-- Приклад 6 Другий варіант видалення всіх співробітників із підрозділу SALES
-- Використання тимчасової таблиці як підзапит
DELETE FROM (
            SELECT * FROM EMP
            WHERE DEPTNO =
                (SELECT DEPTNO FROM DEPT
                    WHERE DNAME ='SALES')
            );
```

Рис. 9 – приклад використання кореляційних підзапитів під час видалення рядків

Створіть один схожий запит на видалення рядків таблиці за допомогою EXISTS або NOT EXISTS.

4. Поєднаний INSERT/UPDATE запит – оператор MERGE

На рисунку 10 показаний приклад використання оператора MERGE.

```
-- 8.1 Створити копію таблиці emp
CREATE TABLE EMP_ALL AS
    SELECT * FROM EMP;

-- 8.2 Змінити коди підрозділів усіх співробітників та
UPDATE emp SET deptno = 20;

-- 8.3 Видалити співробітників, перша літера імені яких починається з A
DELETE FROM emp
    WHERE ename like 'A%';

-- 8.4 Відновити вихідні коди підрозділів роботи невидалених співробітників,
-- а також відновити видалених співробітників
MERGE INTO EMP A
    USING EMP_ALL B
        ON (A.EMPNO = B.EMPNO)
    WHEN MATCHED THEN
        UPDATE SET A.DEPTNO = B.DEPTNO
    WHEN NOT MATCHED THEN
        INSERT (EMPNO, ENAME, JOB, MGR,
                HIREDATE, SAL, COMM, DEPTNO)
            VALUES (B.EMPNO, B.ENAME, B.JOB, B.MGR,
                B.HIREDATE, B.SAL, B.COMM, B.DEPTNO);
```

Рис. 10 – Приклад використання оператора MERGE

Створіть один схожий запит на видалення, використовуючи одну або дві таблиці вашої бази даних.

Етап 3. Ієрархічні та рекурсивні запити

1. Виберіть таблицю вашої БД, до якої потрібно додати нову колонку, яка стане FK-колонкою для PK-колонки цієї таблиці та буде використана для зберігання ієрархії.

Використовується команда ALTER TABLE таблиця ADD колонка тип_даних;

Заповніть дані для створеної колонки, виконавши серію команд UPDATE.

2. Використовуючи створену колонку, отримайте дані з таблиці через ієрархічний зв'язок типу «зверху-вниз».

3. Згенеруйте унікальну послідовність чисел, використовуючи рекурсивний запит, в діапазоні від 1 до 100. На основі отриманого результату створіть запит, що виводить на екран список ще не внесених значень однієї з PK-колонок однієї з таблиць БД за прикладом на рисунку 11.

```
-- 14 Вибрати номери підрозділів, які пропущені у таблиці dept.
SELECT SQ.RN
FROM (SELECT ROWNUM AS RN
        FROM DUAL CONNECT BY LEVEL <=
                    (SELECT MAX(DEPTNO) FROM DEPT)
    ) SQ
WHERE SQ.RN NOT IN (SELECT DEPTNO FROM DEPT)
ORDER BY RN;
```

Рис. 11 – Приклад запиту отримання пропущених значень

Етап 4. Документування результатів роботи на Веб-сервісі *GitHub*

4.1 Розпочинаючи роботу над документуванням рішень лабораторної роботи, необхідно у вашому *GitHub*-репозиторії створити *Issue* з назвою «*tasks-of-laboratory-work-7*».

1) створити *Issue* з назвою «*tasks-of-laboratory-work-7*»;

2) підключити до *Issue* ваш *GitHub-project* (правий розділ «*Projects*» сторінки з *Issue*);

3) змінити статус *Issue* з «*Todo*» на «*In progress*», автоматично перевівши *Scrum*-картку з цим *Issue* на *Scrum*-дошку «*In progress*»;

4) створити нову *Git*-гілку з назвою, яка відповідає назві *Issue*, наприклад, «*tasks-of-laboratory-work-7*» (використовується посилання «*Create a branch*» у правому розділі «*Development*» сторінки з *Issue*).

4.2 Після створення *Git*-гілки перейти до цієї гілки для створення оновлень файлів *Git*-репозиторію.

4.3 У новій гілці Git-репозиторію створити каталог з назвою «7-Nested-Hierar» (кнопка «Add file» - «Create new file»), при створенні якого одночасно створити файл README.md з першим рядком «7 Проектування вкладених DQL, DML-запитів» зі стилем «Заголовок 3-го рівня» мови розмітки Markdown (три символи решітка ###).

4.4 Розмістити в каталозі «7-Nested-Hierar» GitHub-репозиторія файли 1.sql, 2.sql, 3.sql з рішеннями завдань відповідних етапів.

4.5 Виконати запит *Pull Request*, розпочавши процес *Code Review*.

Під час створення *Pull Request* необхідно вказати:

- *Reviewers = Oleksandr Blazhko, Maria Glava*;
- *Labels = enhancement (New feature or request)*;
- *Projects* = посилання на *GitHub-project*.

Завершення процесу *Code Review* відбудеться до початку нового заняття, після чого викладач закриє *Issue*, завершуючи процес виконання завдань з лабораторної роботи.

Під час консультації в понеділок ви зможете отримати більше коментарів щодо ваших рішень.