**Лабораторна робота№5** Використання операторів мови DМL – оператори INSERT, UPDATE, DELETE, COMMIT

**Мета:** Набуття практичних навичок використання операторів DМL – оператори INSERT, UPDATE, DELETE, COMMIT

**Хід роботи**

**Постановка задачі**. Продовжуємо працювати з постановкою задачі з минулих лабораторних та практичних робіт по створенню БД гіпотетичного навчального закладу.

***Методичні рекомендації***

*При вивченні теми слід звернути увагу на створення та збереження структури бази даних(БД), редагування структури, визначення типів полів, зв’язування таблиць у БД.*

1. Повторити лекцію №7. Ознайомитися з теоретичною частиною.
2. Для раніш створених таблиць Викладач, Студент, Група, Предмети, Розклад та Аудиторії написати SQL-запит для введення даних до них (5-10 записів) та дозаповнити таблиці. Написати використані SQL-запити. По кожній таблиці виконати запит: SELECT \* FROM <ім’я таблиці>

Зробити скрінщот отриманих результатів, який разом з SQL-запитом включити до звіту по ЛР.

1. Обновити 2 – 3 записи в існуючих таблицях за допомогою команди UPDATE. Записати використані SQL-запити до звіту по ЛР.
2. Вилучити 2 – 3 записи в існуючих таблицях за допомогою команди DELETE Записати використані SQL-запити до звіту по ЛР.
3. Сформувати звіт з наданням виконаних команд та відповідними поясненнями.
4. Оформити результати відповідно до стандарту подання лабораторних/ практичних робіт.
5. Результати надсилати на електронну адресу викладача [t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com)

Файл повинен мати назву в такому форматі:

**DB<Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **DB3101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт -"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-31, 32 - 29.10.2023**

***Контрольні запитання***

1. Дайте визначення SQL?
2. Чи є SQL мовою програмування?
3. Перелічите основні завдання й функції SQL
4. DDL дайте визначення й назвіть характерні риси
5. DML призначення й принципи
6. Які різновиди DML мови ви знаєте?
7. Які є об'єкти в БД?
8. Назвіть основні функції операторів Create, Drop й Alter
9. Приведіть приклад синтаксису операторів Create, Drop й Alter
10. Коротка характеристика команди SELECT, її особливості.
11. Коротка характеристика команди UPDATE, її особливості.
12. Коротка характеристика команди INSERT, її особливості.

**ТЕОРЕТИЧНА ЧАСТИНА**

У SQL для виконання операцій введення даних в таблицю, їх зміни і видалення призначені три команди мови маніпулювання даними (DML). Це команди – INSERT (вставити), UPDATE (оновити), DELETE (видалити). Для фіксації транзакції використовується команда COMMIT.

**Добавлення даних в таблицю бази даних**

Команда INSERT додає нові рядки в таблицю:

- додати рядок за допомогою положення стовпців в таблицю (INSERT VALUES);

- додати рядок за допомогою назв стовпців (INSERT VALUES);

- додати рядки з однієї таблиці в іншу (INSERT SELECT).

Важливі параметри команди INSERT:

* при додаванні рядків за допомогою положення стовпців в таблицю ви додаєте значення в новий рядок у тій же послідовності, в який вони з’являються в таблиці. При вставці рядка в стовпчик ви задаєте назву стовпця, до якого додаєте значення для нового рядка. Потрібно завжди додавати рядки за допомогою назв стовпців. При цьому ваш запит буде працювати і у випадку, якщо хтось змінить порядок стовпців у таблиці або додасть нові стовпці;
* за допомогою команди INSERT VALUES ви вказуєте точні значення, які повинні бути вставлені в таблицю. За допомогою команди INSERT SELECT ви вибираєте рядки з іншої таблиці, які хочете внести до поточної; INSERT VALUES додає в таблицю один рядок, а INSERT SELECT – будь-яка кількість рядків;
* кожне додане значення повинне бути того ж типу (або можливість для конвертації), що і інші дані в стовпці (для перетворення використовується функція CAST()). Приклад: *insert into groups values(cast('3' as int),'IPZ-33');*
* щоб зберегти систему посилань, доданий зовнішній ключ повинен містити або NULL, або значення існуючого ключа з первинного або унікального посилання вторинного ключа;
* додане значення не може відмінити обмеження (наприклад, CHECK);
* жоден вираз не повинен приводити до арифметичної помилки (наприклад, переповнення або ділення на нуль);
* порядок рядків в таблиці не має значення, ви не можете керувати розташуванням рядків, тому нові рядки можуть з’являтися в будь-якому місті таблиці.

**Команда INSERT** здійснює вставку в таблицю нового рядка. У найпростішому випадку вона має наступний вигляд:

***INSERT INTO*** <ім’я таблиці> ***VALUES*** (<значення>, <значення>, ...);

При такому записі зазначені в дужках після ключового слова VALUES значення вводяться в поля доданого в таблицю нового рядка в тому порядку, в якому відповідні стовпці вказані при створенні таблиці.

Наприклад, введення нового рядка в таблицю Authors може бути здійснений таким чином

INSERT INTO Authors VALUES (101, ‘Semenyuk’, ‘Оleksandr’, ‘Оleksandrovich’, ‘Kyiv’, 15);

Щоб така команда могла бути виконана, таблиця з вказаними в ній ім’ям (Authors) повинна бути попередньо визначена (створена) командою CREATE TABLE. Якщо в будь-поле необхідно вставити NULL-значення, то воно вводиться як звичайне значення:

INSERT INTO Authors VALUES (101, ‘Semenyuk’, ‘Оleksandr’, NULL, ‘Kyiv’, 15);

У випадках, коли необхідно ввести значення полів в порядку, відмінному від порядку стовпців, заданого командою CREATE TABLE, або якщо потрібно ввести значення не у всі стовпці, то слід використовувати наступну форму команди INSERT:

INSERT INTO Authors (AUID, Address, SURNAME, NAME) VALUES (101, ‘Kyiv’, ‘Semenyuk’, ‘Оleksandr’);

Стовпцям, найменування яких не зазначені в наведеному в дужках списку, автоматично присвоюється значення за замовчуванням, якщо воно призначене при описі таблиці (команда CREATE TABLE), або значення NULL.

За допомогою команди INSERT можна витягти значення з однієї таблиці і розмістити його в іншій. Наприклад:

INSERT INTO Authors1 SELECT \* FROM Authors WHERE Address = ‘Kyiv’;

При цьому таблиця Authors1 повинна бути попередньо створена командою CREATE TABLE і мати структуру, ідентичну таблиці Authors.

**Додавання рядка за допомогою назв стовпчиків**

Формат оператору

INSERT INTO *table*

(*column1*, *column2*, …, *columnN*)

VALUES(*value1*, *value2*, …, *valueN*);

*table* – це назва таблиці, в яку додається рядок;

*column1*, *column2*, …,*columnN* – список назв стовпчиків в *table*;

*value1*, *value2*, …, *valueN* – список літерних позначок або виразів, які задають значення для вказаних стовпчиків в новому рядку. Кількість значень повинно відповідати кількості стовпчиків в списку, а значення повинні бути вказані в тій же послідовності, що і назви стовпчиків.

СУБД вставляє кожне значення в стовпчик, використовуючи відповідні значення в списку. Значення *value1* додається в стовпчик *column1* нового рядка, значення *value2* – в стовпчик *column2* и т.д. Пропущеному стовпчику присвоюється значення по замовчуванню абр NULL.

Приклади використання команди INSERT

1) INSERT INTO authors VALUES(

'A08', 'Michael', 'Polk', '512-953-1231', '4028 Guadalupe St', 'Austin', 'TX', '78701');

2) INSERT INTO authors(

au\_id,au\_fname,au\_lname,phone,address,city,state,zip)

VALUES('A09','Irene','Bell','415-225-4689','810 Throckmorton Ave','Mill Valley','CA',

'94941');

3) INSERT INTO authors(

zip,

phone,

address,

au\_lname,

au\_fname,

state,

au\_id,

city)

VALUES(

'60614',

'312-998-0020',

'1937 N. Clark St',

'Weston',

'Dianne',

'IL',

'A10',

'Chicago');

4) INSERT INTO authors(

au\_id,

au\_fname,

au\_lname,

phone)

VALUES(

'A11',

'Max',

'Allard',

'212-502-0955');

5) INSERT INTO publishers

SELECT

pub\_id,

pub\_name,

city,

state,

country

FROM new\_publishers

WHERE city = 'Los Angeles';

6) INSERT INTO publishers(

pub\_id,

pub\_name,

city,

state,

country)

SELECT

pub\_id,

pub\_name,

city,

state,

country

FROM new\_publishers

WHERE country <> 'USA';

7) INSERT INTO publishers(

pub\_id,

pub\_name,

city,

state,

country)

SELECT \*

FROM new\_publishers

WHERE pub\_name = 'XXX';

**Видалення даних з таблиці БД**

Команда **DELETE** видаляє рядки з таблиці. Формат команди:

***DELETE FROM*** *table* [***WHERE*** *search\_condition*];

table – це назва таблиці, з якої видаляються рядки.

*search\_condition* задає умови, які повинні виконуватися для рядків, що видаляються. Це умови порівняння LIKE, BETWEEN, IN та IS NULL, або умови запиту (оператори порівняння IN, ALL, ANY и EXISTS, які будемо розглядати при вивченні запитів) в комбінації з AND, OR и NOT. Якщо вираз WHERE упускається, то вилучаються всі рядки з таблиці.

Наступний вираз видаляє всі рядки таблиці Authors1.

DELETE FROM Authors1;

В результаті таблиця стає порожньою.

Для видалення з таблиці відразу декількох рядків, що задовольняють деякій умові можна скористатися пропозицією WHERE, наприклад,

DELETE FROM Authors1 WHERE AUID = 103;

Можна видалити групу рядків

DELETE FROM Authors1 WHERE Address = ‘Kyiv’;

Важливі параметри команди DELETE:

􀂄 на відміну від команд INSERT и UPDATE, ця команда не потребує введення назв стовпців, оскільки вилучає рядки цілком;

􀂄 видаляє рядки з таблиці, а не саму таблицю, яку можна вилучити (разом з даними, індексами та ін.) командою DROP TABLE;

􀂄 використовує (опціонально) вираз WHERE для визначення які саме рядки потрібно вилучити. Якщо не вказати умову WHERE, команда DELETE вилучить всі рядки в таблиці;

􀂄 може бути небезпечною, оскільки ви можете випадково пропустити вираз WHERE (та вилучити всі рядки) або неправильно вказати умову пошуку для WHERE (та вилучити не ті рядки). Перед запуском команди DELETE рекомендується запустити команду SELECTіз такою же умовою WHERE. Команда SELECT відобразить всі рядки, що будуть вилучені СУБД при запуску команди DELETE. Щоб побачити тільки кількість рядків, що будуть вилучені використовуйте команду SELECT COUNT (\*);

􀂄 для збереження цілісності посилань СУБД дозволяє вказати дію, яка виконується автоматично за допомогою команди DELETE при видаленні рядків, на які вказує вторинний ключ (FOREIGN KEY);

􀂄 жодний вираз не повинен приводити до арифметичної помилки (наприклад, переповнення або ділення на нуль);

􀂄 порядок рядків в таблиці не має значення, ви не можете управляти розташування м рядків, тому їх видалення може довільним чином змінити розташування інших рядків в таблиці.

**Оновлення даних в таблиці БД**

Команда UPDATE змінює значення в існуючих рядках таблиці. Використовується для зміни:

􀂄 всі рядки в таблиці;

􀂄 окремі рядки в таблиці.

Для оновлення рядка потрібно вказати:

􀂄 яку таблицю змінювати;

􀂄 назву стовпців, які потрібно змінити, а також нові значення;

􀂄 умову пошуку для находження рядків для оновлення (опціонально).

Важливі параметри команди UPDATE:

􀂄 використовує вираз WHERE, в якому вказується, які рядки потрібно змінити Без виразу WHERE команда UPDATE змінить всі рядки таблиці;

􀂄 може бути небезпечна, тому що ви можете випадково пропустити вираз WHERE (та змінити всі рядки) або неправильно вказати умову пошуку для WHERE (змінити не ті рядки). Перед запуском команди UPDATE рекомендується запустити команду SELECT з тим же виразом WHERE, що і для оновлення рядків. Команда SELECT відобразить всі рядки, які будуть змінені СУБД при запуску команди UPDATE. Щоб відобразити тільки кількість таких рядків використовуються командою SELECT COUNT (\*);

􀂄 кожне змінене значення повинно бути того ж типу (або мати можливість для конвертації), що і інші дані в стовпці;

􀂄 для збереження цілісності посилання, СУБД дозволяє вказати дію, яка буде виконуватися автоматично за допомогою команди UPDATE при зміні значення, на які вказує вторинний ключ (FOREIGN KEY);

􀂄 змінене значення не може відмінити обмеження, накладене на стовпець (CHECK);

􀂄 жодний вираз не повинен приводити до арифметичної помилки (наприклад, переповнення або ділення на нуль);

􀂄 порядок рядків в таблиці не має значення і ви не можете керувати розташуванням рядків, тому нові рядки можуть з’явитися в будь-якому місті таблиці.

Формат команди

**UPDATE** *table*

**SET** *column* = *expr*

[**WHERE** *search\_condition*];

*table* –назва таблиці, яка буде обновлюватися;

*column* – назва стовбця (з даними для змін) в *table*;

*expr* – літерний вираз або запит, який зчитує одно значення. Значення, прочитане *expr*, замінить існуюче значення в *column*. Щоб змінити значення в декількох стовпцях, введіть в пункті SET список виразів (*column* = *expr*), розділених комами. Ви можете задавати список полів для оновлення в будь-якому порядку.

Умова *search\_condition* задає умови, які повинні виконуватися для рядків, що змінюються. Такими умовами можуть бути умови WHERE (оператори порівняння, LIKE, BETWEEN, IN и IS NULL) або умови запиту (оператори порівняння, IN, ALL, ANY и EXISTS) в комбінації з AND, OR и NOT. Якщо ви пропустите вираз WHERE, будуть змінені всі рядки в таблиці.

Команда UPDATE дозволяє змінювати, тобто оновлювати, значення деяких або всіх полів в існуючому рядку або рядках таблиці. Наприклад, щоб для всіх продажів, відомості про яких містяться в таблиці BookSales, змінити значення поля Sales на 200, можна використовувати конструкцію:

UPDATE BookSales SET Sales = 200;

Для задання конкретних рядків таблиці, значення полів яких повинні бути змінені, в команді UPDATE можна використовувати предикат, що вказується в пропозиції WHERE.

UPDATE BookSales SET Sales = 200 WHERE ISBN =’978-5-699-79339-6’;

В результаті виконання цього запиту буде змінено поле Sales тільки для книг з ISBN =’978-5-699-79339-6’.

Команда UPDATE дозволяє змінювати не тільки один, але й декілька стовпців:

UPDATE BookSales SET Sales = 200, Units=1000 WHERE ISBN = ‘978-5-699-79339-6’;

У пропозиції SET команди UPDATE можна використовувати скалярні вирази, що вказують спосіб зміни значень поля, в які можуть входити значення змінюваного та інших полів.

Наприклад, для збільшення в таблиці BookSales значення поля Sales в два рази для книг з ISBN = ‘978-5-699-79339-6’ можна використовувати запит:

UPDATE BookSales SET Sales = Sales\*2 WHERE ISBN=’978-5-699-79339-6’;

Вираз SET не є предикатом, тому в ньому можна вказати значення NULL наступним чином:

UPDATE BookSales SET Sales = NULL WHERE ISBN=’978-5-699-79339-6’;

**Приклади.**

UPDATE titles

SET contract = 0;

UPDATE titles

SET price = price \* 2.0

UPDATE titles

SET type = 'self help',

pages = NULL

WHERE type = 'psychology';

UPDATE titles

SET sales = sales \* 0.5

WHERE sales >

(SELECT AVG(sales)

FROM titles);

UPDATE titles

SET pubdate = DATE '2003-01-01'

WHERE title\_id IN

(SELECT title\_id

FROM title\_authors

WHERE au\_id IN

(SELECT au\_id

FROM authors

WHERE au\_fname = 'Sarah'

AND au\_lname = 'Buchman'));

UPDATE titles

SET pub\_id =

(SELECT pub\_id

FROM publishers

WHERE pub\_name = 'Abatis Publishers')

WHERE pub\_id =

(SELECT pub\_id

FROM publishers

WHERE pub\_name = 'Tenterhooks Press');

СУБД буде розраховувати вираз в SET або WHERE з використанням значення, які знаходилися в стовпці до початку змін.

UPDATE mytable

SET col1 = col1 \* 2,

col2 = col1 \* 4,

col3 = col2 \* 8,

WHERE col1 = 1

AND col2 = 2;

СУБД задає col1 рівним 2, col2 – рівним 4 (1×4, а не 2×4), а col3 – рівним 16 (2×8, а не 4×8).