**Лабораторна робота №7 Виконання SQL-запитів на вибірку даних**

**Мета:** Набуття практичних навичок формування з SQL-запитів до БД

Контрольні запитання

1. Яке призначення і склад оператора SELECT.
2. Назвіть вимоги до порядку розміщення стовпців в операторі SELECT.
3. Яка особливість використання символу (\*) в операторі SELECT.
4. Охарактеризуйте призначення пропозиції оператора SELECT – FROM.
5. Яке призначення пропозиції оператора SELECT – WHERE.
6. Яка суть пошуку за шаблоном.
7. Які особливості використання ключових слів AND і ОR.
8. Яке призначення пропозиції оператора SELECT – ORDER BY.
9. Яке призначення і склад оператора INSERT.
10. Яке призначення і структура оператора UPDATE.

***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 лекції №7, 8, теоретичну частину ЛР № 4,5,6 та ПР№6. Ознайомитися з теоретичною частиною цієї ЛР.
2. За результатами виконання ЛР № 4,5,6 та ПР№6 повинна бути сформована та заповнена реляційна БД з таблицями Викладач, Студент, Групи, Предмети, Розклад, Аудиторії, Оцінки. Для цих таблиць виконати завдання:
3. Створити запит для демонстрації модифікатору DISTINCT.
4. Продемонструвати дію спеціальних функції IN, BETWEEN, LIKE, та IS NULL в умовах.
5. Продемонструвати роботу спеціальних функцій з умовою NOT.
6. Продемонструвати використання COUNT(\*)
7. Продемонструвати виконання простих обчислень у запиті.
8. Використати просте обчислення як параметр агрегатної функції.
9. Продемонструвати роботу GROUP BY.
10. Продемонструвати роботу HAVING.

Таблиці та функції обрати самостійно.

1. Результати оформити звітом, в якому надаються сформовані SQL-запити, опис дій, що вони виконують та скріншот виконання на вашій БД.
2. Оформити результати відповідно до стандарту подання лабораторних/ практичних робіт.
3. 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 - 12.10.2023**

**Теоретична частина**

Розглянемо БД, яка складається з 4-х таблиць. В БД надаються страви, які використовуються в різних епізодах серіалу Seinfeld. На рис. 1 показано даталогічну модель БД 

Рисунок 1. Даталогічна модель БД «Food»

Foods є головною таблицею. Кожен запис відповідає окремій страві, її назва реєструється в полі name. Поле Type\_id посилається на таблицю food\_types, яка містить класифікацію продуктів (тобто, фаст-фуд, напої і так далі). Нарешті, таблиця food\_episodes поєднує продукти з великою кількістю епізодів.

Загальний огляд команди select.

Оператор select включає реляційні операції за допомогою серії фраз (речення – clause). Кожна фраза відповідає своїй реляційній операції. У SQLite майже всі фрази не обов’язкові. Користувач SQLite може використовувати тільки ті, операції яких він потребує. Найбільша загальна форма select в SQLite представлена на рис. 2.

***SELECT*** [distinct] списокСтовпців

***FROM*** Ім’яТаблиці

***WHERE*** УмоваПошуку

***GROUP BY*** стовпецьГрупування

***HAVING*** УмоваПошуку

***ORDER BY*** УмоваСортування;

Рисунок 2.Загальна форма команди вибірки даних

Одна із інтерпретацій оператора select – це уявити його як конвеєр, який обробляє відношення. На конвеєрі є необов’язкові процеси, виконання яких можна пропускати. Незалежно від того, використовуються чи ні конкретні операції (процеси), конвеєр завжди працює однаково. На рис.3 можна подивитися порядок виконання. Виконання оператора select починається з фрази from, яка приймає одне або більше відношень і поєднує їх в одне складне відношення і, потім, передає ланцюжку послідовних операцій.



Рисунок 3. Ланцюжок послідовних операцій під час виконання вибірки даних

Фраза from містить список декількох таблиць, представлень і підзапитів (представлених у вигляді змінної tables), розділених комами. Більш ніж одна таблиця (представлення або підзапит) будуть по’єднуватися в одне відношення, яке на рис.3 представляється назвою R1. Різні елементи поєднуються в одне відношення операцією join.

Фраза where відбирає необхідні записи з R1. За ключовим словом where йде предикат (логічний вираз), який визначає критерій відбору записів з R1, які повинні бути включені до наступного відношення. Обрані записи утворюють нове відношення R2.

Як показує рис. 4, фраза select в SQLite по’єднує усі дані, розглянуті у фразі from, відбирає записи (обмежує) їх у фразі where і відбирає поля (проектує) у фразі select.



Рисунок 4**.** Етапи виконання команди select

Згадаємо, що Інструкція ***Select*** має такі розділи:

**SELECT** *перелік полів або виразів*

**INTO** *ім’я нової таблиці*

**FROM** *перелік таблиць, з яких вибрані поля*

**WHERE** *умови відбору записів*

**GROUP BY** *поля або вирази для угруповання даних*

**HAVING** *умови відбору записів після угруповання*

**UNION**

**ORDER BY** *перелік полів для сортування*

Обов’язковими є тільки розділи SELECT та FROM. Інші розділи краще використовувати в тому порядку, в якому вони тут записані.

***Розділ  SELECT***

Він визначає потрібні для запиту поля та вирази. Полю можна надати нове ім’я – *псевдонім* **(*alias*)**, яке буде виводиться як його назва. Можна створювати обчислювальні поля. Синтаксис розділу:

SELECT [ ALL | DISTINCT | DISTINCTROW] <*список полів* >

* ALL – використовується за замовчанням і дозволяє включати до запиту однакові рядки;
* DISTINCT – аналізується результат вибірки, і з однакових значень виводиться лише одне, тобто всі рядки вибірки різні.
* DISTINCTROW – кожен рядок вибірки буде створений з унікальної комбінації ***записів*** базових таблиць, але в залежності від того, які поля вибрані, може статися так, що ***вибірка***матиме однакові рядки. Параметр DISTINCTROW впливає на результат вибірки тільки тоді, коли використовувалась операція JOIN та до вибірки були включені не всі стовпці таблиць (рядки-копії виключаються).

### *****Розділ  FROM*****

В цьому розділі через кому записуються імена або псевдоніми таблиць та представлень, з яких вибираються дані.

Якщо до запиту входить декілька таблиць, то треба вказувати типи їх з'єднання. Синтаксис:

*Головна таблиця* **JOINTYPE** *дочірня таблиця* **ON**  *умови зв’язку таблиць*

**ON *умови зв’язку таблиць*** вказує які поля таблиць треба порівнювати і яким чином. Найчастіше використовується оператор =.

**JOINTYPE** – тип з'єднання таблиць. Можливі такі типи:

1) INNER JOIN – встановлюється за замовчанням. До запиту включаються лише ті записи з обох таблиць, в яких співпадають значення відповідних полів.

2) LEFT [OUTER] JOIN – до запиту включаються всі записи головної (лівої) таблиці, незалежно від того чи існують для них відповідні записи в дочірній (правій) таблиці. Якщо в правій таблиці немає даних для вибірки, в стовпці запиту їхнє місце вільне.

3) RIGHT [OUTER] JOIN – до запиту включаються всі записи правої таблиці, незалежно від того чи існують для них відповідні записи в лівій таблиці.

4) FULL [OUTER] JOIN – в MS Access відсутній. В SQL Server до запиту ввійдуть усі записи з двох таблиць: ті, що задовольняють умові з’єднання, і ті, що не задовольняють.

***Розділ  WHERE***

Тут вказується логічна умова, від виконання якої залежить чи ввійде запис до вибірки. Можна вказати декілька умов, об’єднаних логічними операторами AND, OR або іншими. WHERE в невеликих запитах може замінити конструкцію INNER JOIN, але це суттєво зменшує швидкість виконання запиту.

***GROUP BY***

Він використовується для підсумкових запитів В списку полів розділу SELECT без змін дозволяється використовувати імена тільки тих полів, які входять до розділу GROUP BY. Усі інші поля повинні бути використані як аргументи підсумкових функцій AVG(), MAX(), MIN(), SUM(), COUNT() або інших.

***HAVING***

Він дуже схожий на розділ WHERE. В них однакові правила запису логічних умов. Але умови, задані в розділі WHERE, перевіряються для окремих записів перед угрупованням, а умови розділу HAVING перевіряються для груп записів після угруповання стовпців, вказаних в розділі GROUP BY і звичайно містять підсумкові функції.

***Розділ  ORDER BY***

В цьому розділі вказуються імена полів або їх номери в списку полів даного запиту (нумерація починається з 1). За замовчанням встановлений порядок сортування за зростанням (ASC). Для сортування за зменшенням використовується службове слово DESC.

***Оператор  UNION***

Запит з використанням UNION можна створити в MS Access тільки в режимі SQL. В результаті роботи створюється вибірка, яка містить рядки, що вибирає як перша, так і друга інструкція *Select*. Синтаксис:

*Інструкція*  SELECT

UNION [ALL]

*Інструкція*  SELECT

[ORDER BY **…**]

Кількість стовпців в обох інструкціях SELECT повинна бути однаковою. Типи даних в стовпцях повинні бути ідентичними. Імена стовпців беруться з першої інструкції SELECT. Якщо вказаний предикат ALL, то будуть виведені всі рядки обох запитів, інакше співпадаючі рядки виводитися не будуть. Інструкцій SELECT може бути декілька, тоді треба вживати круглі дужки. Упорядкування ORDER BY можна використати єдиний раз після останньої інструкції, воно діє на всю вибірку. В ньому треба використовувати номери стовпців або їхні імена з першої вибірки.

***Розділ  INTO***

Синтаксис: INTO  *нова таблиця*

Результати запиту будуть записані в нову таблицю з вказаним іменем і потрібною структурою, яка створюється автоматично. Якщо таблиця з таким іменем вже існує, то вона буде вилучена.

Examples for our test base

SELECT Sub\_code, ID\_T from Subject,Teachers where Sub\_code>11 and ID\_T=51

SELECT Sub\_name, Name\_T from Subject,Teachers where Sub\_code>11 and ID\_T=51

CREATE TABLE TeachSubj

(

id\_ts INTEGER PRIMARY KEY AUTOINCREMENT,

Sub\_code INTEGER,

ID\_T INTEGER,

FOREIGN KEY (Sub\_code) REFERENCES Subject (Sub\_code )

FOREIGN KEY (ID\_T) REFERENCES Teachers (ID\_T )

);

insert INTO TeachSubj SELECT DISTINCT NULL, Sub\_code, ID\_T from Subject,Teachers

SELECT name NM FROM Studebts

UNION

SELECT Name\_T NM FROM Teachers

ORDER by NM

insert INTO TeachSubj SELECT DISTINCT NULL, Sub\_code, ID\_T from Subject,Teachers WHERE Sub\_code >12 and Sub\_code <16 AND Name\_T not in

(SELECT name NM FROM Studebts

UNION

SELECT Name\_T NM FROM Teachers)

insert INTO TeachSubj SELECT DISTINCT NULL, Sub\_code, ID\_T from Subject,Teachers

WHERE Sub\_code >12 and Sub\_code <16 AND Name\_T in

(SELECT name NM FROM Studebts

UNION

SELECT Name\_T NM FROM Teachers)

SELECT count(\*) FROM TeachSubj

select DISTINCT Sub\_code, ID\_T from TeachSubj

SELECT DISTINCT Name\_T, Sub\_name from Teachers, Subject, TeachSubj

WHERE TeachSubj.ID\_T=Teachers.ID\_T and TeachSubj.Sub\_code=TeachSubj.Sub\_code

order by Name\_T, Sub\_name

SELECT DISTINCT Name\_T, Sub\_name from Teachers, Subject, TeachSubj

WHERE TeachSubj.ID\_T=Teachers.ID\_T and TeachSubj.Sub\_code=Subject.Sub\_code and Teachers.ID\_T<12 and Subject.Sub\_code > 11

order by Name\_T, Sub\_name

delete from TeachSubj

WHERE TeachSubj.ID\_T<12 and TeachSubj.Sub\_code > 11

SELECT DISTINCT Name\_T, Sub\_name from Teachers, Subject, TeachSubj

WHERE TeachSubj.ID\_T=Teachers.ID\_T and TeachSubj.Sub\_code=Subject.Sub\_code

order by Name\_T, Sub\_name

SELECT count(\* ) as Cout\_Tech, Sub\_name from Teachers, Subject, TeachSubj

WHERE TeachSubj.ID\_T=Teachers.ID\_T and TeachSubj.Sub\_code=Subject.Sub\_code

GROUP BY TeachSubj.Sub\_code

order by Name\_T, Sub\_name