МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Миколаївський національний університет імені В.О.Сухомлинського
Коледж МНУ імені В.О.Сухомлинського
Циклова комісія технічного напряму підготовки

МЕТОДИЧНІ МАТЕРІАЛИ
для вивчення тем, запланованих
на самостійне опрацювання з навчальної дисципліни
«БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ»

Тема: «Мова структурованих запитів SQL»

для студентів галузі знань 11 «Математика та статистика»

спеціальності 113 «Прикладна математика»

МИКОЛАЇВ – 2019

Методичні вказівки до самостійної роботи з навчальної дисципліни «Бази даних та інформаційні системи» (для студентів IV курсу вищих навчальних закладів I-II рівнів акредитації, які здійснюють підготовку молодших спеціалістів на основі базової загальної середньої освіти) / Коледж МНУ імені В.О.Сухомлинського; уклад.: Цимбалиста М.І. – М., 2019.– 25 с.

Укладач: Цимбалиста Марина Іванівна, викладач вищої категорії циклової комісії технічного напряму підготовки

Розглянуто та схвалено для використання в роботі на засіданні циклової комісії технічного напряму підготовки № __ від _____.2019р.

ЗМІСТ

1. МЕТА ТА ЗАДАЧІ ДИСЦИПЛІНИ, ЇЇ МІСЦЕ В НАВЧАЛЬНОМУ ПРОЦЕСІ

1.1. Мета викладання дисципліни

Метою даного курсу є вивчення мови структурованих запитів SQL, а також надбання практичних навичок розробки та налагодження відповідного програмного забезпечення.

1.2. Задачі вивчення дисципліни

В наслідку вивчення дисципліни студенти повинні:

—вивчити основні підходи та загальні принципи проектування інформаційного забезпечення автоматизованих систем;

- придбати навички розробки та експлуатації сучасних баз даних.

1.3. Загальні вказівки

Дана дисципліна вивчається у 7 та 8 семестрах, а також виконання контрольних завдань. Працювати над дисципліною треба систематично. При цьому необхідно користуватися не тільки основною, а й додатковою літературою.

Під час самостійної опрацювання матеріалу слід також враховувати, що контрольні завдання складені таким чином, щоб результат їхнього виконання був покладений у підвалини виконання лабораторного практикуму.

Підведення підсумків вивчення дисципліни здійснюється у вигляді заліку та іспиту. До складення заліку допускаються студенти, які успішно виконали лабораторний практикум і всі контрольні завдання.

2. ПРОГРАМА ДИСЦИПЛІНИ ТА МЕТОДИЧНІ ВКАЗІВКИ З ЇЇ ВИВЧЕННЯ

2.1. Загальні питання

У якості серверу баз даних використовуються реляційні СКБД, для яких стандартним засобом доступу до даного є високорівнева мова структурованих запитів ***SQL (Structured Query Language)*** [4,5,9,11, 14,18].

У 1992 році Міжнародним комітетом із стандартизації *(ISO -International Standard Organization)* був затверджений стандарт SQL, який часто називають ***SQL/92*** або просто ***SQL2***, вимоги якого враховуються більшістю виробників СКБД.

В 1999 році з'явився новий стандарт - SQL3, який якісно відрізняється від попередніх стандартів своєю об'єктною орієнтацією, а також деякими змінами у використанні.

SQL може використовуватися для маніпуляції з даними (вибірка і модифікація), опрацювання структури бази даних (створення та вилучання таблиць та індексів) і адміністрування бази даних (визначення списку користувачів, призначення прав доступу).

Використання SQL має багато переваг, найбільше важливими з яких є:

- SQL підтримується багатьма постачальниками СКБД. Програми, написані з використанням мови SQL, будуть працювати майже на будь-якому сервері бази даних;
- SQL простий у вивченні, оскільки в ньому для опису дій використовуються прості англійські слова, такі, як ***SELECT, INSERT, UPDATE, DELETE, COMMIT, ROLLBACK.***

Мова SQL може використовуватися двома способами. Перший передбачає ***інтерактивну роботу,*** що міститься у вводі користувачем окремих SQL-операторів і негайного одержання результату.

Другий міститься у ***впровадженні*** SQL-операторів у програми на інших мовах програмування (наприклад, *Ci).* Таке впровадження може зробити програму більш потужною й ефективною. Проте несумісність цих мов програмування зі структурою SQL і властивим йому стилем керування даними потребує внесення ряду розширень в інтерактивний SQL.

2.2. Типи даних

В SQL/92 визначаються різні типи даних, які позначаються ключовими словами: CHARACTER, CHARACTERVARYING, BIT, BITVARYING, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLEPRECISION, DATE, TIME, TIMESTAMP і INTERVAL.

Типи даних CHARACTER і CHARACTERVARYING спільно називаються типами даних символьних рядків; типи даних BIT і BITVARYING – типами даних бітових рядків. Типи даних символьних рядків і типи даних бітових рядків спільно називаються рядковими типами даних, а значення рядкових типів називаються рядками.

Типи даних NUMERIC, DECIMAL, INTEGER і SMALLINT спільно називаються типами даних точних чисел. Типи даних FLOAT, REAL і

DOUBLEPRECISION спільно називаються типами даних приблизних чисел. Типи даних точних чисел і типи даних приблизних чисел спільно називаються числовими типами. Значення числових типів називаються числами.

Типи даних DATE, TIME і TIMESTAMP спільно називаються типами дати-часу. Значення типів дати-часу називаються " дата-час". Тип даних INTERVAL називається інтервальним типом. представляються у формі

2.3. Класифікація команд SQL

Команди *SQL* використовуються для виконання різноманітних дій над реляційними БД. Для зручності роботи вони розділяються на наступні групи (табл. 2.1):

– команди визначення даних (***Data Definition Commands***);
– команди маніпуляції даними (***Data Manipulation Commands***);
– команди вибірки даних (***Data Query Commands***);
– коман ди керування транзакціями (***Transaction Control Commands***);
– команди керування даними (***Data Control Commands***).

Таблиця 2.1 – Команди мови SQL

Команда	Призначення
1	2
Команди визначення даних	
alter table	Змінює структуру таблиці
create index	Створює індекс
create table	Створює таблицю
create view	Створює подання
Drop	Вилучає таблицю, індекс, подання
Команди маніпуляції даними	
Delete	Вилучає запису таблиці
Insert	Добавляє запису в таблицю
Update	Змінює дані таблиці
Команда вибірки даних	
Select	Вибирає дані
Команди керування транзакціями	
commit	Робить зміни, проведені з початку транзакції, постійними
rollback	Відкочує всі проведені зміни до крапки зберігання або до початку транзакції
savepoint	Встановлює контрольну крапку, до котрого згодом можна буде виконати відкат
Команди керування даними	
check database	Перевіряє цілісність бази даних
grant	Надає привілеї
revoke	Скасовує надані раніше привілеї

2.4. Створення баз даних і таблиць

2.4.1. Створення баз даних

До ANSI-стандарт SQL-92 не входить оператор ***CREATE DATABASE***. Замість цього він передбачає команду ***CREATE SCHEMA*** для визначення частини бази даних, якою буде володіти конкретний користувач. Проте в комерційні версії SQL зазвичай включена команда ***CREATE DATABASE***, спрощена форма котрої (тобто без необов'язкових пропозицій і розширень) має вигляд

CREATE DATABASE ім'я_бази_даних.

Створення бази даних не означає автоматичну можливість її використання. У залежності від версії *SQL* для обертання до конкретної бази даних необхідно ввести

USE ім'я_бази_даних

або

DATABASE ім'я_бази_даних

2.4.2. Створення таблиць

За допомогою команд визначення даних ***SQL*** можна створювати, вилучати та маніпулювати таблицями (добавляти, вилучати, перестав-ляти стовпчики та змінювати їхні параметри).

Щоб створити таблицю треба зробити, щонайменше, наступне (наведені приклади зроблено за допомогою СКБД Access).

– задати ім'я таблиці;
– задати імена її стовпчиків;
– визначити тип даних для кожного стовпчика;
– визначити (або використовувати по умовчанню) нульовий статус для кожного стовпчика – припускається або забороняється використання в стовпчику нульових значень.

```
CREATE TABLE ім'я_таблиці
( ім'я_стовпчика тип_даних [NULL | NOT NULL]
[, ім'я_стовпчика тип_даних [NULL | NOT NULL] ] …);
```

Статус стовпчика ***NOT NULL*** означає обов'язкове заповнення відповідного стовпчика. Статус ***NULL*** – означає, що значення стовпчика можуть бути не визначені. Наприклад, відсутність телефону у визначеного клієнта. Деякі системи по умовчанню вставляють деяке значення (пробіл або звичайний нуль), якщо в стовпчик не були введені жодні дані.

Наприклад,

```
create table постачальник (назва varchar(25) not
      null, адреса varchar(35) not null, факс
      char(8) null);
```

2.4.3. Обмеження на множину припустимих значень

Більшість комерційних систем підтримують пропозиції (обмеження)

PRIMARY KEY, UNIQUE, DEFAULT, CHECK, REFERENCES і ***FOREIGN KEY*** у команді ***CREATE TABLE*** у відповідності зі стандартом ***SQL/92.*** Ці елементи забезпечують захист цілісності даних.

– ***PRIMARY KEY*** помічає стовпчик (у якому не можуть бути присутніми нульові значення) у якості первинного ключа таблиці. Кожне значення, що вводиться в цей стовпчик, повинно бути унікальним. Якщо первинний ключ включає декілька стовпчиків (складовий ключ), ***PRIMARY KEY*** визначається на рівні таблиці.

```
create table клієнти
        (код int not null primery key,
        назва char(10) not null, місто char(10));
```

У наступному прикладі обмеження на первинний ключ є обмеженням на всю таблицю, оскільки первинний ключ є складовим.

```
create table іспити
        (дисципліна varchar(35) not null, група char(8) not
      null, дата date, primery key (дисципліна, група));
```

– ***UNIQUE*** – альтернативний ключ. Також гарантує унікальність елементів стовпчика. Це обмеження застосовується до тих полів, що були заявлені ***NOT NULL***.

Стовпчики, що повинні містити унікальні значення, але котрі не є первинними ключами, називаються ***можливими ключами (candidate keys).***

За функціональними можливостями обмеження ***PRIMARY KEY*** й ***UNIQUE*** подібні, за винятком того, що тільки один первинний ключ (що складається з будь-якої кількості стовпчиків) може бути визначений для даної таблиці. Отже, існує розходження між первинним ключем і унікальними стовпчиками в засобах їхнього використання з зовнішніми ключами.

Унікальною можна також зробити й групу полів, вказавши ***UNIQUE*** в якості обмежень таблиці. При цьому комбінації значень зазначених стовпчиків повинні бути унікальними, хоча в самих стовпчиках значення можуть бути й неунікальними. При об'єднанні в групу важливий порядок. Наприклад, комбінації значень стовпчиків ***'a', 'b'*** і ***'b', 'a'*** відрізняються.

Наприклад, потрібно відстежити всі покупки, зроблені за день у кожного продавця.

```
create table покупки
        (ном_продавця int not null, дата date not null, сума
        money, unique (ном_продавця, дата);
```

– ***DEFAULT*** визначає значення, що автоматично вставляється системою, якщо користувач не введе необхідні дані. Наприклад, слово ***невизначений,*** якщо значення ще не визначене.

– ***CHECK*** визначає, які дані можуть вводитися у визначений стовпчик, тобто дозволяє визначити область припустимих значень стовпчика. При цьому можна задати список, діапазон або шаблон припустимих значень. Наприклад, задати щоб у стовпчик уводилися значення, що складаються з двох букв і чотирьох цифр. Якщо користувач спробує порушити це

обмеження, то команда виконуватися не буде. Ці обмеження іноді називають ***правилами (rules)***.

Обмеження з перевіркою також можуть застосовуватися до всієї таблиці, якщо вони використовують більш одного стовпчика. Наприклад, у таблиці ***Продажі*** є стовпчики для опису кількості замовлених і відправлених товарів. У цьому випадку можна використовувати табличне обмеження, що перевіряє, щоб кількість відправлених книг не перевищувала кількості замовлених.

```
create table продажі (назва_id char(6) not null primery
key, відправлено int not null,
замовлено int null, дата_замовлення date null, check
(відправлено <= замовлено));
```

2.4.4. Підтримка посилальної цілісності

Таблиці бази даних можуть бути певним чином зв'язані між собою (рис. 2.1). Такі зв'язки між таблицями встановлюються шляхом завдання посилань (***references***) між окремими полями таблиць.

Коли поле однієї таблиці посилається на поле в іншій таблиці, воно називається ***зовнішнім ключем*** (***foreign key***); поле, на яке воно посилається, називається його ***батьківським ключем***. Імена зовнішнього та батьківського ключів можуть бути неоднаковими. Наприклад, таблиці ***Покупці*** та ***Продавці*** зв'язуються між собою полем ***ном_прод***.

Покупці

ном	***прізвище***	***ном_прод***
101	Дорошенко	1
102	Петренко	2
103	Сидоренко	1
104	Іваненко	3
105	Степаненко	4
106	Романенко	2

Продавці

ном_прод	***прізвище***	***сума***
1	Сахно	1000
2	Рябуха	2000
3	Кравчук	1500
4	Заремба	5000

Рисунок 2.1 – Приклад використання зовнішнього ключа

Як видно з прикладу, кожне значення зовнішнього ключа повинно бути подане в батьківському ключі один і тільки один раз. У той же час на одне значення батьківського ключа можуть посилатися декілька полів зовнішнього ключа.

```
create table покупці
(ном integer not null primary key,
прізвище char(15),
ном_прод integer,
foreign key (ном_прод) references продавці(ном_прод));
```

Обмеження ***REFERENCES*** і ***FOREIGN KEY*** зв'язують разом первинні та зовнішні ключі. Якщо вводиться значення в стовпчик зовнішнього ключа, визначеного за допомогою речення ***REFERENCES***, цей стовпчик і стовпчик,

на який він посилається, повинні існувати в таблицях, інакше команда модифікації даних буде відхилена.

Зовнішній ключ, як і первинний, може бути визначений на будь-якій кількості полів, які разом розглядаються як єдине ціле (складовий ключ). Зовнішній і батьківський ключі повинні бути визначені на однаковій множині полів (за кількістю, типам і порядку слідування):

FOREIGN KEY (список стовпчиків) REFERENCES ім'я таблиці (список стовпчиків)

2.5. Проектування запитів

2.5.1. Загальний формат команди SELECT

Для створення запитів використовується конструкція ***SELECT***. Результатом виконання запиту є таблиця, що зберігається в тимчасовому буфері серверу бази даних. Обрані дані можна використовувати для перегляду, друку звітів, формування графіків або, наприклад, додати їх до постійних (базових) таблиць бази даних.

Найбільш проста конструкція ***SELECT*** наступна:

SELECT список полів, що вибираються
FROM список таблиць;

Результатом запиту є інша таблиця, що утворюється з заданих у запиті таблиць. Наприклад:

select код, назва, ціна from товари;

Якщо необхідно вивести всі стовпчики таблиці, то використовується символ «*»:

Select * from товари;

2.5.2. Вибір рядків: речення WHERE

Речення ***WHERE*** задає предикат, умову, що може бути вірною або помилковою для кожного рядку таблиці. В результаті вибираються тільки ті рядки з таблиці, для яких предикат має значення «істина». У ***SQL*** є ряд операторів і ключових слів для завдання умов.

Оператори порівняння

Ці оператори використовуються в такий спосіб:

WHERE вираз оператор_порівняння вираз

У якості виразів можуть використовуватися константи, імена стовпчиків, функції, підзапити або їхні комбінації, що пов'язані арифметичними операторами.

Зазвичай оператори порівняння застосовуються до чисельних значень. У ***SQL*** вони також можуть застосовуватися до даного з типами ***char*** і ***varchar*** (< означає *раніш за абеткою*, > означає *пізніше*) і до дат (< означає *раніш у хронологічному порядку*, > означає *пізніше*). При цьому символьні значення та дати містяться у лапки.

```
select код, назва, місто from
Постачальники where місто ="Київ";
```

select прізвище, посада
from Посадовці
where прізвище > "Іваненко";
select *
from Продавці where рейтинг > 100;
select *
from titles
where advance*2 > price + sales;

<u>Комбінації умовних і логічних операцій</u>

Логічні оператори виконуються в суворо визначеній послідовності (рис. 2.2). Якщо у виразі присутні обидва типи операторів, першими здійснюються арифметичні оператори.

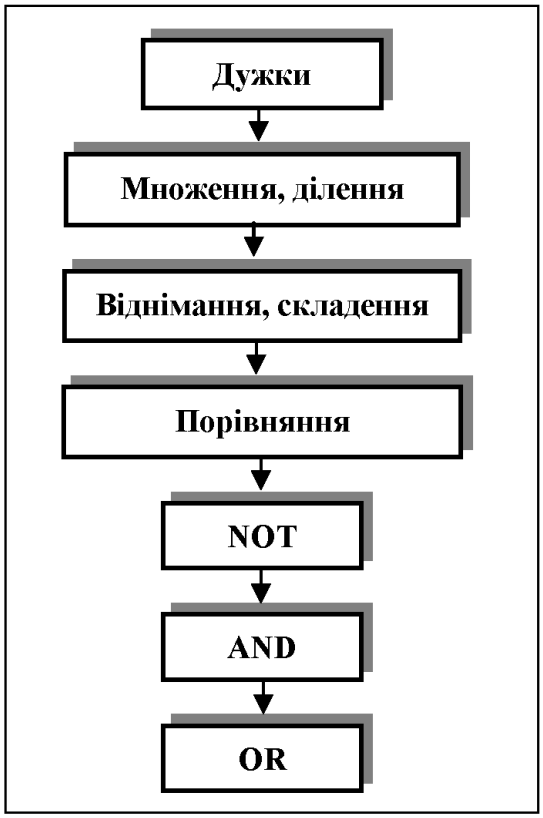

Рисунок 2.2 – Послідовність виконання операцій

Роздивимося пошук моніторів в таблиці ***Товари***, незалежно від ціни, а також принтерів із ціною більше 700 грн.

Select модель, тип, ціна
From Товари
Where тип="монітор" or тип="принтер" and ціна >700;

Обмеження на ціну застосовується тільки до принтерів, тому що оператор **AND** виконується перед оператором **OR.**

Щоб першим виконувався оператор **OR**, у запиті треба використовувати дужки. Наприклад, якщо потрібно вибрати всі принтери та сканери з ціною вище 1000 грн:

Select модель, тип, ціна

From Товари

Where (тип="монітор " or тип="принтер ") and ціна >1000;

Роздивимося запит, що здійснює пошук книг, витрати на який не окупилися, тобто прибуток від продажу котрих менше подвоєної суми, що сплачена авторам. Крім того, введемо контрольну дату.

select назва, тип, ціна, гонорар, продано, дата from Книги where ціна*продано < 2* гонорар and дата < 45/09/04';

Діапазони (BETWEEN і NOT BETWEEN)

Діапазон можна визначити двома способами:

- за допомогою операторів порівняння < >;
- за допомогою ключового слова (оператора) ***BETWEEN***

Ключове слово ***BETWEEN*** можна використовувати для створення включаючого діапазону, якщо в результуючі значення повинні включатися й межі діапазону. Наприклад:

select модель, кількість
from Товари
where кількість between 100 and 150;

При використанні конструкції ***not between*** знаходяться рядки, значення яких лежать поза вказаним діапазоном. Наприклад:

select модель, кількість
from Товари
where кількість not between 100 and 150;

Аналогічний результат дає використання операторів порівняння:

select модель, кількість
from Товари
where кількість < 100 and кількість > 150;

Треба бути уважними при завданні діапазону символьних значень. Наприклад, необхідно провести вибірку покупців, прізвища яких потрапляють у заданий алфавітний діапазон:

select *
from Автори
where прізвище between 'А' and 'Л'

При цьому прізвища авторів, які починаються на ***Л,*** виявляться пропущеними, хоча оператор ***BETWEEN*** є включаючим. У даному випадку порівнюються рядки різної довжини: рядок 'Л' і, наприклад, рядок 'Лупенко'. Том у ***BETWEEN*** доповнює рядок 'Л' пробілами. У більшості кодів знак пробілу передує символам кирилиці. Тому Лупенко не був вибраним. Для правильної вибірки потрібно вказати наступну букву алфавіту або приписати літеру 'я' (декілька літер 'я', якщо це необхідно).

Списки (IN, NOT IN)

Ключове слово ***IN*** дозволяє вибрати значення, що збігаються зі значеннями з заданого списку. Наприклад:

select name, royalty
from authors

where name in ('Довлатов', 'Бродський', 'Шмельов');

що еквівалентно запиту:

select name, royalty
from authors
where name='Довлатов'or name='Бродський'
or name='Шмельов';

<u>Вибірка нульових (невідомих) значень</u>

Значення ***NULL*** використовується для подання невідомої інформації. При цьому ***NULL***-значення не є звичайним нулем або порожньою позицією. Його не можна порівнювати з будь-яким іншим значенням. Проте, якщо за логікою запиту необхідно вибрати саме таку інформацію, то потрібно використовувати наступну конструкцію:

select код, картина, художник from Живопис
where художник is null;

<u>Пошук по підрядках (LIKE і NOT LIKE)</u>

При необхідності вибору інформації за неповними даними використовується наступна конструкція:

WHERE ім'я_стовпчика [NOT] LIKE 'зразок' [ESCAPE ключовий символ]

Ключове слово ***LIKE*** може бути застосовано до символьних полів і в деяких системах до полів дати, проте для чисельних полів воно не може бути застосовано.

Зразок (в лапках) може містити один або декілька шаблонів – символів, які заміщають у зразку пропущені букви або рядки. Ключове слово ***ESCAPE*** використовується, якщо в зразку міститься один із шаблонів, але його потрібно розглядати як просту літеру.

У ***ANSI SQL*** використовуються два символи шаблону разом із ключовим словом ***LIKE:*** знак відсотка (%) і підкреслення (_).

% – будь-який рядок з будь-якою кількістю символів; **_** – будь-який одиночний символ.

Наприклад:

select автор, назва
from Книги
where назва like "%Fox Pro%";
select *
from Покупці
where прізвище like 'А%';
select *
from Автори
where прізвище like 'Mc%' or прізвище like 'Mac%;
select *
from Терміни
where термін like «*"&[Введіть назву терміна]&"*";

Щоб знайти в рядку символ підкреслення або відсотка, у предикаті ***LIKE*** будь-який символ можна визначити як ***ESCAPE-символ*** (керуючий символ). Він використовується в предикаті безпосередньо перед символом шаблона й означає, що наступний за ним символ інтерпретується саме як звичайний символ, а не символ шаблона.

```
select notes from table
    where notes like "%/%% escape /";
```

У результаті будуть обрані всі значення, що містять символ «%».

Приклади:

like "27%" - рядок, що починається з 27;

like "27@%" - 27%;

like "_n" - an, in, on тощо;

like "@_n" - _n.

2.5.3. Агрегатні функції

Агрегатні функції (табл. 2.2) використовуються для одержання узагальнюючих значень. Вони дають єдине значення для цілої групи рядків таблиці. Агрегатні функції завжди мають аргументи, що є виразами та містяться у дужках. У якості аргументів зазвичай використовуються назви стовпчиків, але також припускаються константи, функції та будь-які їхні комбінації з арифметичними операторами.

Таблиці 2.2 – Агрегатні функції

Функція	Результат
COUNT	Визначає кількість рядків або значень поля, обраних за допомогою запиту, що ***не є NULL-значеннями***
SUM	Обчислює арифметичну суму всіх обраних значень даного поля
AVG	Обчислює середнє значення для всіх обраних значень даного поля
MAX	Обчислює найбільше зі всіх обраних значень даного поля
MIN	Обчислює найменше зі всіх обраних значень даного поля

Для ***SUM*** і ***AVG*** можуть використовуватися тільки цифрові поля. Для ***COUNT***, ***MAX*** і ***MIN*** – цифрові та символьні поля. Стосовно до символьних полів ***MAX*** і ***MIN*** визначають значення відповідно коду ***ASCII*** згідно алфавіту.

Наприклад, середня зарплата у випадку індексації

```
select avg(оклад*індекс)
from відомість;
```

Функції ***COUNT(DISTINCT)*** і ***COUNT(*)*** відрізняються тим, що перша рахує кількість непорожніх і неповторюваних рядків, а друга – загальну

кількість рядків. Багато реалізацій підтримують аргумент ALL (не підтримуваний ANSI), що при використанні у функції ***COUNT*** дозволяє підраховувати і повторювані, але не порожні значення. Наприклад:

select count (*) from Покупці;
select count (all рейтинг) from Покупці;
select count (distinct назва) from Поставки;

В агрегатних функціях для обмеження кількості рядків, що беруть участь в обчисленнях, можна використовувати конструкцію ***WHERE.*** Наприклад:

select назва, sum(сума)
from Продажі
where назва = "Монітор"

2.5.4. Групування даних

Речення ***GROUP BY*** розділяє таблицю на набори, а агрегатна функція обчислює для кожного з них підсумкове значення. Ці значення називаються ***агрегатним вектором.***

```
SELECT список вибору
  FROM список таблиць
  [WHERE умови]
  [GROUP BY список_угруповання];
```

Наприклад:

select художник, count(картина) from Музеї
group by художник;

До списку вибору включаються стовпчик, за яким проводиться угруповання, та агрегатна функція.

Шляхом сортування одночасно за декількома елементами можна створювати групи всередині груп.

select музей, художник, count(картина) from Музеї
group by музей, художник;

При відсутності агрегатної функції конструкція ***GROUP BY*** в запиті еквівалентна реченню ***DISTINCT***:

select художник
from Музеї
group by художник;

select distinct художник
from Музеї;

Спільна робота речень ***WHERE*** і ***GROUP BY*** відбувається в такий спосіб. Спочатку знаходяться всі рядки, що відповідають умові ***WHERE***. Потім речення ***GROUP BY*** поділяє відібрані рядки на групи. Рядки, що не задовольняють умовам речення ***WHERE***, не включаються до жодної групи.

select музей, count(картина) from Музеї
where painter = "Рубенс" group by музей;

<u>Речення HAVING</u>

У самому загальному сенсі речення ***H AVING*** працює аналогічно реченню

WHERE, але застосовується до груп. ***WHERE*** накладає обмеження на рядки, а ***HAVING*** – на групи. Як правило, речення ***HAVING*** використовується разом із реченням ***GROUP BY.***

Якщо в списку вибору є агрегатні функції, речення ***WHERE*** виконується перед ними, тоді як речення ***HAVING*** застосовується до всього запиту в цілому, після розбивки на групи та обчислення значень функцій.

З погляду синтаксису умовного виразу речення ***WHERE*** і ***HAVING*** ідентичні. Відмінність полягає у тому, що в умові речення ***WHERE*** не можуть знаходитися агрегатні функції. Крім того, у більшості систем елементи речення ***H AVING*** повинні включатися до списку вибору. На речення ***WHERE*** це обмеження не поширюється. У реченні ***H AVING*** може міститися будь-яка кількість умов.

Пропозиція ***H AVING*** працює наступним чином: спочатку ***GROUP BY*** розділяє рядки на набори (за типом), потім на отримані групи накладаються умови речення ***HAVING.***

```
select тип, count(*)        select автор
from Товари                 from Книги
group by тип                group by автор
having count(*) > 1;        having автор like "П%";
```

Якщо в реченні ***H AVING*** є декілька умов, вони об'єднуються за допомогою операторів ***AND, OR*** або ***NOT***.

```
select pub id, sum(advance), avg(price) from tit _ es
  group by pub id having sum(advance) > $15000 and
  avg(price) < $20 and pub_id > '0800';
```

2.5.5. Сортування результатів запиту

Для поліпшення сприйняття результатів виконання запиту можна використовувати конструкцію ***ORDER BY,*** що дозволяє упорядкувати вихідні дані. Дані можуть сортуватися як по убуванню ***(DESC***), так і по зростанню ***(ASC***). По умовчанню прийнята зростаюча послідовність сортування.

При цьому задати сортування можна у такі способи:

- зазначити стовпчик або стовпчики, за якими буде виконуватися сортування;
- зазначити номер (позицію) виразу в списку вибору;
- зазначити ім'я виразу.

Наприклад,

```
select *
from Посадовці
order by прізвище asc;
```

При використанні більш одного стовпчика в пропозиції ***ORDER BY*** виконується ***вкладене сортування***, тобто спочатку виконується сортування по першому полю, а потім по наступним. Кількість рівнів сортування не обмежується. Стандарт ***ANSI***, а також більшість комерційних систем, потребують, щоб стовпчики, за якими проводиться сортування, були присутні в списку вибору.

Порядок указівки стовпчиків у реченні ***ORDER BY*** не зобов'я-заний збігатися з порядком перерахування стовпчиків і виразів в операторі ***SELECT.***

```
select товар, ціна
from Поставлено
order by ціна dsc, товар asc;
```

У реченні ***ORDER BY*** замість імені стовпчика можна зазначити його позицію в списку вибору.

```
select телефон, прізвище from Абоненти
order by1 asc;
```

Стовпчики, отримані за допомогою агрегатних функцій, а також вирази зі списку вибору ***SELECT*** можна використовувати також у реченні ***ORDER BY,*** якщо на них посилаються за номером. Речення ***ORDER BY*** завжди виконується останнім.

Наприклад, треба знайти середню вартість книг за кожним типом, витрати на які більше 5000 грн, і впорядкувати результати за ціною.

```
select тип, avg(ціна) from Книги
where витрати > 5000 group by тип order
by 2 desc;
```

Якщо в списку вибору вираз визначений із заголовком, по ньому можна виконувати сортування.

```
select видавництво, ціна*уцінка as т_ціна, назва from Книги
    order by видавництво, т_ціна desc;
```

2.5.6. З'єднання таблиць

За допомогою операції з'єднання в одному операторі ***SELECT*** можна маніпулювати даними з декількох таблиць.

Кожне з'єднання задається для двох таблиць, хоча в одному операторі ***SELECT*** може виконуватися декілька з'єднань. При цьому в якості стовпчиків, за якими проводиться з'єднання, використовуються первинний (або потенційний) та зовнішній ключі.

Існують два варіанти вказівки стовпчиків з'єднання: в реченні ***FROM*** з використанням операції ***INNER JOIN*** або в реченні ***WHERE:***

```
SELECT список_вибору
        FROM таблиця_1 INNER JOIN таблиця_2 ON
                [таблиця_1.]стовпчик з'єднання = [таблиця_2.]
                стовпчик з'єднання]
```

або

```
SELECT список_вибору
        FROM таблиця_1, таблиця_2 [, таблиця_3]…
            WHERE [ таблиця_1.]стовпчик з'єднання =
                    [ таблиця_2.] стовпчик з'єднання
```

Нехай в БД є таблиці ***Товари(код, тип, назва, ціна)*** і ***Замовлення (ном_замовлення, дата, код_товару, ціна_одиниці, кількість, вартість,***

між яким існує зв’язок ***1:N.*** Необхідно визначити, коли та в якій кількості замовлявся конкретний товар.

select Товари.назва, Товари.тип, Замовлення.кількість,
Замовлення.дата from Товари Inner Join Замовлення
On Товари. код= Замовлення. код_товару where Товари.назва =
[Введіть назву товару];

або

select Товари.назва, Товари.тип, Замовлення.кількість,
Замовлення.дата from Товари, Замовлення
On Товари. код= Замовлення. код_товару where (Товари.назва =
[Введіть назву товару]) AND
(Товари. код= Замовлення. код_товару);

В результаті виконання запиту з’єднуються ті рядки обох таблиць, в яких значення стовпчиків з’єднання співпадають.

Щоб прискорити ввід запитів і зробити їх більш зрозумілими, у списку таблиць можна визначити псевдоніми таблиць (аліаси).

select T.назва, T.тип, З.кількість, З.дата from товари as T, замовлено as З where T.код =З.код_товару

2.5.7 Запити на об'єднання

Результати виконання декількох запитів можна об'єднати за допомогою речення ***Union***. Наприклад, з таблиць ***Викладачі***, ***Співробітники*** та ***Аспіранти*** необхідно вибрати інформацію про дні народження:

Select Прізвище, дата From
Викладачі Union
Select Прізвище, дата From Співробітники Union
Select Прізвище, дата
From Аспіранти Order by дата;

Операція ***Union*** може об’єднувати результати декількох запитів, таблиць або інструкцій SQL. Наприклад,

TABLE [Замовлення] Union
TABLE [Замовлення_архів];

Примітки:

1. Для об'єднання двох і більш запитів необхідно, щоб їхні стовпчики, що входять у набір вихідних даних, були ***сумісні по об'єднанню.***
2. Якщо NULL-значення заборонені для будь-якого стовпчика в об'єднанні, то вони повинні бути заборонені для усіх відповідних стовпчиків в інших запитах об'єднання.
3. У реченні сортування ***Order by*** назва атрибута повинна братися з першого запиту.

Оператор Union можна використати як різновид оператора IF для відображення різних значень для одного поля, залежно від значень в інших полях. Без оператора Union для одержання аналогічного результату потрібне виконання декількох запитів.

3. ІНДИВІДУАЛЬНІ ЗАВДАННЯ

3.1 Загальні положення

В процесі вивчення дисципліни студенти заочної форми навчання повинні виконати контрольну роботу. Основна ціль її виконання – закріплення теоретичного матеріалу та оволодіння навичками його практичного використання.

Контрольна робота передбачає виконання 15 завдань, кожне з яких складається з двох частин: створення таблиць бази даних і відповідних запитів.

Приступати до виконання контрольних робіт слід лише після детального вивчення теоретичного матеріалу та виконання лабораторних робіт згідно з відповідними методичними вказівками.

Результати виконання кожної інфдивідуальної роботи оформлюються у вигляді звіту, який виконується комп'ютерним способом і повинен містити завдання та тексти відповідних SQL-запитів.

Робота повинна бути датована та підписана виконавцем.

Виконані інфдивідуальні роботи роботи повинні бути представлені на перевірку в суворій відповідності до графіку навчального процесу. При наявності зауважень викладача слід виконати всі виправлення та доповнення.

Захист індивідуального завдання відбувається безпосередньо на комп'ютері.

3.2 Варіанти завдань

Варіант № 1

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Пацієнт: номер пацієнта; ім'я пацієнта; адреса пацієнта; дата операції; діагноз; препарат, призначений після операції; дата народження, стать пацієнта.

Хірург: код хірурга, ім'я, дата народження, стать, стаж роботи хірургом, категорія, адреса, телефон.

Відділення: код відділення, назва.

Зв'язати таблиці, додавши таблицю ***Операції***.

Написати на мові SQL наступні запити:

1)список прооперованих пацієнтів кожного хірурга із вказівкою діагнозу і віку;

2)список пацієнтів клініки з вказівкою прізвища хірурга і його категорії;

3)список пацієнтів (із вказівкою імені хірурга, його категорії і назви відділення); прооперованих 14 травня 2011 року з діагнозом "виразка шлунка";

4)список пацієнтів чоловічої статі з усіх відділень.

Варіант № 2

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Навчальний заклад: назва навчального закладу; рівень акредитації,

місто, в якому розташований навчальний заклад, кількість студентів очної форми навчання, кількість студентів заочної форми навчання, кількість викладачів.

Міністерства (у веденні яких знаходяться навчальні заклади)*:* повна назва міністерства, адреса, телефон.

Написати на мові SQL наступні запити:

1) назви навчальних закладів, розташованих у м. Києві;
2) для кожного міністерства список навчальних закладів і їхньої категорії;
3) список ВНЗ (по містах) із вказівкою назви міністерства, міста та телефону;
4) список навчальних закладів (із вказівкою назв), розташованих в Києві, що мають кількість студентів денної форми навчання більше 1000.

Варіант № 3

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Кафедра: код кафедри, назва кафедри, аудиторія, де розташована кафедра, телефон, прізвище завідувача. ***Викладачі:*** код, прізвище.

Дисципліни: код, назва.

Розклад іспитів: предмет; дата; час; лектор; група; аудиторія; кількість студентів.

Написати на мові SQL наступні запити:

1) повна інформація про іспит (по предмету);
2) коли, в яких аудиторіях і в яких групах лектор Зайцев А.И. приймає іспити;
3) вивести для кожної кафедри (у заголовку її повна назва і телефон) список груп, дати іспитів і прізвища викладачів;
4) список груп (за датою іспиту) із вказівкою назви предмета і кафедри, на якій працює викладач.

Варіант № 4

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Розклад руху потягів: номер потяга; пункт призначення; час відправлення; час прибуття; кількість годин в шляху; відстань у км.

Потяги: номер потяга, кількість вагонів спальних, купейних, пасажирських, дата останнього техогляду.

Написати на мові SQL наступні запити:

1) список потягів, які відправляються до вказаного пункту призначення;
2) список всіх потягів, які відправляються в указаний період часу;
3) список потягів, пункти призначення яких знаходяться більше, ніж 300 км від станції відправлення;
4) список потягів, які проходили техогляд більше ніж тиждень тому.

Варіант № 5

Створити за допомогою мови SQL базу даних, концептуальна модель

якої складається з наступних сутностей:

Товар: назва, група товару, ціна.

Замовлення: номер, замовник, дата, загальна сума.

Зв'язати таблиці, та додати інформацію про ціну та кількість товару в кожному замовленні.

Написати на мові SQL наступні запити:

1) список і кількість товарів, що замовлялися за вказаний період часу;
2) склад конкретного замовлення с вказівкою вартості окремих товарів;
3) список замовників, які замовляли конкретний товар за вказаний період.

Варіант № 6

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Театри міста: назва театру, адреса, телефон, прізвище директора, кількість місць у театрі.

Спектакль: код спектаклю, назва спектаклю; режисера; автор п'єси.

Театральна афіша: назва театру; адреса, спектакль, дата; час.

Написати на мові SQL наступні запити:

1) для кожного театру (із вказівкою в заголовку адреси, телефону) список спектаклів із прізвищем режисера та автора п'єси;
2) список театрів, в яких йдуть спектаклі за А.П. Чеховим;
3) розклад показів конкретного спектаклю;
4) театральні спектаклі на поточну дату.

Варіант № 7

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Кінотеатри: назва кінотеатру, район, адреса, кількість місць, категорія, телефон.

***Кінофільми*:** назва, режисер, кіностудія, країна.

Кіноафіша: назва кінотеатру; назва фільму, дата. Враховувати, що у кожного фільму може бути декілька сеансів.

Написати на мові SQL наступні запити:

1) список кінотеатрів за районами міста з зазначенням адреси та телефону, де демонструється конкретний фільм;
2) список кінофільмів із вказівкою прізвища режисера, кінотеатру, його категорії, адреси та телефону;
3) список кінотеатрів (із вказівкою назви кінотеатру і району, в якому він розташований), в яких йдуть фільми, що вироблені у Франції;
4) список фільмів, які демонструються в указаному районі.

Варіант № 8

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Читачі: номер читацького квитка, прізвище, адреса, телефон, місце роботи.

Книги: назва; автор; видавництво; рік випуску; ціна книги.

Автори: код, прізвище, ім'я, по батькові.

Зв'язати таблиці, додавши таблицю ***Видача книг,*** яка додатково включає поля ***дата видачі*** та ***дата повернення***.

Написати на мові SQL наступні запити:

1) список боржників (термін утримання книги більше місяця) з вказівкою прізвища, телефону та місця роботи абонента;
2) список книг, які знаходяться на руках у вказаного читача;
3) список читачів, які обслуговувалися за вказаний період часу.

Варіант № 9

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Викладачі: особистий номер, ФИО, посада, вчений ступінь, вчене звання, кафедра, дата народження..

Кафедри: код кафедри, назва кафедри, аудиторія, телефон, факультет, кількість співробітників.

Написати на мові SQL наступні запити:

1) список кафедр (з повною характеристикою) указаного факультету;
2) список викладачів (за абеткою) указаної кафедри (прізвище, посада, оклад);
3) список викладачів (за абеткою) вказаного факультету, що мають вказаний вчений ступінь;
4) список днів народження (впорядкованих у межах року) викладачів указаної кафедри.

Варіант № 10

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Бібліографічна картка: номер картки; назва книги; видавництво; рік випуску; ціна книги, тематика.

Видавництво: назва, місто, адреса, телефон.

Автори: код, прізвище, ім'я, по батькові.

Написати на мові SQL наступні запити:

1) список книг з авторами та повним бібліографічним описом конкретного видавництва за вказаною тематикою;
2) список книг, які випущені конкретним видавництвом за вказаний період;
3) список книг вказаного автора;
4) список видавництв вказаного міста.

Варіант № 11

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Співробітник: табельний номер, підрозділ, посада, дата прийому на роботу, адреса, телефон.

Посадові оклади: посада, оклад.

Підрозділи: назва, прізвище керівника, телефон.

Написати на мові SQL наступні запити:

1) список співробітників (прізвище, посада, оклад) указаного підрозділу;
2) список службовців із вказівкою підрозділу, старше 40 років, які працюють на посаді інженера;
3) список службовців із вказівкою підрозділу та посади, що мають стаж роботи в даній організації більше, ніж 10 років;
4) підвищити всі оклади на 20%.

Варіант № 12

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Будівельні організації: код, назва організації, адреса, телефон.

Побудовані об'єкти: назва об'єкта, адреса розташування об'єкта (вулиця, номер будинку), дата здачі, оцінка, організація-здавач;

Написати на мові SQL наступні запити:

1) для кожної будівельної організації список об'єктів з адресою і датою здачі;
2) за вказаним районом визначити перелік об'єктів, їх адрес, оцінки здачі, назви організації-здавача та її адреси;
3) перелік об'єктів, які були споруджені за конкретний період і відповідних будівельних організацій;
4) список об'єктів (із вказівкою назви об'єкта, оцінки), зданих конкретною будівельною організацією в першому кварталі поточного року.

Варіант № 13

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Програмісти: прізвище програміста, відділ, в якому він працює.

Програми: ім'я програми, бібліотека, в якій зберігається, кількість операторів, мова, на якій написана, призначення, дата створення.

Зв'язати таблиці.

Написати на мові SQL наступні запити:

1) одержати список програм, написаних мовою Асемблер, обсягом більш 150 операторів;
2) одержати для кожного відділу список програм, створених співробітниками відділу із вказівкою мови програмування, на якій створена програма;
3) вивести для кожної бібліотеки програм її склад із вказівкою автора програми та відділу, в якому він працює;
4) вивести список програм, довжина яких перевищує 300 операторів і які були створені в поточному році.

Варіант № 14

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Платіжна відомість: особистий номер; ФИО; підрозділ; стаж роботи, сума; дата виплати; тип виплати.

Підрозділі код підрозділу, назва підрозділу, начальник, телефон.

Написати на мові SQL наступні запити:

1) список співробітників вказаного підрозділу з вказівкою виплат по кожному співробітнику за конкретне дату;
2) одержати список співробітників, що одержали премію 15 травня 2006 р.;
3) список виплат вказаному працівнику за певний період;
4) список співробітників по стажу роботи із вказівкою назви підрозділу

Варіант № 15

Створити за допомогою мови SQL базу даних, концептуальна модель якої складається з наступних сутностей:

Групи: номер, факультет, спеціальність. Склад групи: номер групи, номер залікової книжки, прізвище студента.

Факультети: код; назва; інститут, до складу якого входить.

Дисципліни: код, назва, лектор.

Зв'язати таблиці, використавши таблицю ***Сесія***.

Написати на мові SQL наступні запити:

1) результати здачі іспиту з конкретної дисципліни в конкретній групі;
2) список груп з повними назвами відповідних факультетів і інститутів;
3) результати сесії в конкретній групі за дисциплінами.

4. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Основна

1. Корнієнко С. К. Системи баз даних: організація та проектування: Навч. Посібник / С.К Корнієнко . – Запоріжжя: ЗНТУ, 2006. – 252 с.
2. Кригель П. SQL. Библия пользователя / П. Кригель. – К.: Диалектика/Вильямс, 2010. – 752 с.
3. Ку з нец о в С.Д. Базы данных: Языки и модели / М.: Бином, 2008. -720 с.
4. Грабер М. Введение в SQL / М.Грабер. – М.: Лори, 2008. – 303 с.

Додаткова

1. Бейли Д. Изучаем SQL / Д. Бейли. - СПб.: Питер, 2011. - 592 с.
2. Боуман Д, Практическое руководство по SQL / Д. Боуман, С.Эмерсон, М.Дарновски. – М.,СПб.: Вильямс, 2002. – 352 с.
3. Васвани Д. MySQL: Использование и администрирование / Д. Васвани. - СПб.: Питер, 2011. - 368 с.
4. Організація баз даних і знань частина 2. Методичні вказівки до лабораторних робіт для студентів напрямку підготовки 6.050103 “Програмна інженерія” усіх форм навчання / Ук л а д . С.К.Корнієнко, О.І.Качан. – Запоріжжя: ЗНТУ, 2012. – 17 с.