

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

Запорізький національний технічний університет  
Факультет інформатики та обчислювальної техніки

Кафедра комп'ютерних  
систем та мереж

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання лабораторних робіт  
з дисципліни “Організація баз даних”  
для студентів спеціальності 123 “Комп'ютерна інженерія”  
усіх форм навчання  
частина VI

**2019**

Методичні вказівки до виконання лабораторних робіт з дисципліни “Організація баз даних” для студентів спеціальності 123 “Комп’ютерна інженерія” усіх форм навчання частина V /Укл.: Т.О. Паромова, Н.В. Щербак- Запоріжжя: ЗНТУ, 2019 – 22 с.

Укладачі: Т.О.Паромова, ст.викладач,

Рецензент: С.Д. Точилін, доцент, к.ф.-м.н.

Відповідальний за випуск: Паромова Т.О., ст.викладач.

Затверджено  
на засіданні кафедри  
“Комп’ютерні системи  
та мережі ”  
Протокол № 7  
від “21” січня 2019.

Рекомендовано до видання  
НМК факультету  
комп’ютерних  
наук і технологій  
Протокол № 6  
від “31” січня 2019.

## ЗМІСТ

|                                                            |    |
|------------------------------------------------------------|----|
| ЛАБОРАТОРНА РОБОТА № 5 .....                               | 4  |
| СТВОРЕННЯ SQL ЗАПИТІВ .....                                | 4  |
| 1. ЗАГАЛЬНІ ВІДОМОСТІ.....                                 | 4  |
| Побудова простих запитів .....                             | 4  |
| Синтаксис оператора SELECT: .....                          | 4  |
| Пропозиція WHERE .....                                     | 5  |
| Пропозиція ORDER BY .....                                  | 8  |
| Пропозиція Group by .....                                  | 9  |
| Застосування) функцій дати та часу.....                    | 9  |
| Застосування агрегатних (узагальнюючих) функцій.....       | 10 |
| Пропозиція HAVING .....                                    | 12 |
| Вкладені запити.....                                       | 13 |
| Використання вкладеного запиту з узагальнюючими функціями  | 14 |
| Використання предиката IN. ....                            | 14 |
| Використання предиката EXIST. ....                         | 15 |
| Використання вкладеного запиту з перевіркою рівності ..... | 15 |
| Багатотабличні запити .....                                | 15 |
| Синтаксис з'єднаних таблиць .....                          | 16 |
| 2. ЛАБОРАТОРНЕ ЗАВДАННЯ.....                               | 17 |
| 3. ЗМІСТ ЗВІТУ.....                                        | 21 |
| 4. КОНТРОЛЬНІ ПИТАННЯ .....                                | 22 |
| 5. РЕКОМЕНДОВАНА ЛІТЕРАТУРА .....                          | 22 |

## ЛАБОРАТОРНА РОБОТА № 5

### СТВОРЕННЯ SQL ЗАПИТІВ

Мета роботи – оволодіти основні способи створення запитів з використанням мови SQL у різних середовищах

### 1. ЗАГАЛЬНІ ВІДОМОСТІ

Запити відносно змісту бази даних та команди та команди зміни бази даних описуються засобами мови SQL.

Оснoву мoви SQL склaдaють oпepaтoри, умoвнo рoзбиті нa кількa гpуп вiдпoвіднo дo викoнувaних фyнкцій:

- Опepaтoри DDL (Data Definition Language) - oпepaтoри визнaчeння oб'єктiв бaзи дaних, нaпpиклaд,
- Опepaтoри DML (Data Manipulation Language) - oпepaтoри мaнiпyлювaння дaними, нaпpиклaд:
- Опepaтoри зaxиcтy і керyвaння дaними, нaпpиклaд:

#### Побудова простих запитів за допомогою оператора вибору **SELECT**

Пpизнaчeння oпepaтoрa **SELECT** склaдaєтьcя y вiдбиpaннi i вiдoбpажeннi дaних oднiєї aбo бiльш тaблиць бaзи дaних. Цей oпepaтoр зaбeзпeчyє викoнaння дiй, еквiвaлeнтних oпepaтoрaм рeляцiоннoй алгeбpи: вiбiркa, oб'єднaння, пpoeкцiя.

Резyльтaтoм викoнaння oпepaтoрa **SELECT** зaвжди є тaблиця.

#### ***Синтаксис оператора SELECT:***

Пocлiдoвнiсть ключoвих слiв в oпepaтoрi **SELECT** нe мoжe бyти змiнeний

**SELECT** [ALL | DISTINCT] { \* [список полів [AS нове ім'я]][,.....] } \*

**FROM** ім'я таблиці [alias][,...]

[**WHERE** <пpедикaт-умoвa вiбiркi aбo з'єднaння>]

[**GROUP BY** <Список полів>]

[**HAVING** <пpедикaт-умoвa для гpупи>]

[**ORDER BY** <Список полів, по яких упорядкувати висновок>]

Квaдpатними дyжкaми пoзнaчeні нeoб'язкoвi пpoпoзицiї oпepaтoрy

**SELECT.**

Оператор **SELECT** закінчується символом „;”.

**SELECT** — ключове слово, що повідомляє СУБД, що ця команда - запит. Усі запити починаються с цього слова з наступним пробілом. За ним може зазначатись спосіб вибірки — з видаленням дублікатів (**DISTINCT**) або без видалення (**ALL**, за замовчуванням). Потім вказуються імена полів, мають включатись в результуюче відношення. **SELECT** [**ALL** | **DISTINCT**] { \* [список полів [**AS** нове ім'я]][,.....]\* )

Пропозиція **SELECT** може включати тільки наступні типи елементів:

- імена стовпців;
- агрегатні функції;
- константи;
- вирази, що включають у себе комбінації перерахованих вище елементів.

Символ \* (зірочка) означає, що у результуючий набір включаються всі стовпці з вихідних таблиць запиту.

У пропозиції **FROM** задається перелік вихідних таблиць запиту. **FROM** <Список таблиць> через кому.

**FROM** - ключове слово, подібно **SELECT**, що повинне бути представлене в кожному запиті. Воно супроводжується пробілом і потім іменами таблиць, використовуваних як джерело інформації. У випадку, якщо зазначено більш одного імені таблиці, над перерахованими таблицями здійснюється операція декартова добутку.

***Приклад*** Створити список студентів, із зазначенням їхній ФІО і групи

```
SELECT ФІО, Група
FROM Студенти;
```

### ***Пропозиція WHERE***

Для обмеження набору рядків, що поміщаються в результуючу таблицю запиту використовується пропозиція **WHERE**, де задаються умови добору рядків результату або умови з'єднання рядків вихідних

таблиць, т.е. задаються обмеження набору рядків, що поміщаються в результуючу таблицю.

**WHERE** <Предикат-умова вибірки або з'єднання>

Існує п'ять основних типів умов пошуку (предикатів)

- **Порівняння** – порівнюються результати обчислень двох виразів  
Предикати порівняння { =, <, >, <=, >=, <> }, що мають традиційний сенс

**Приклад** Вибрати всіх студентів, що навчаються в групі РП-410

Select Прізвище

From Студенти

Where курс=4;

- **Діапазон** - перевіряється, чи попадає результат обчислення в заданий діапазон значень

Предикат **Between A and Y** — приймає значення між A и B. Коли порівнюване значення попадає в заданий діапазон, включаючи границі діапазону, тоді відповідний рядок включається в результуючу таблицю. Одночасно вікористовується протилежний предикат **Not Between A and Y**, коли порівнюване значення не попадає в заданий інтервал, включаючи його границі, відповідний рядок не включається у результуючу таблицю.

**Приклад:** Скласти список студентів з другого по четвертий курси.

SELECT Студенти.\*

FROM Студенти

WHERE Студенти.[курс] Between 2 And 4;

- Приналежність до множина - перевіряється, чи належить результат до заданої безлічі значень

Предикат входження в множину **IN (множина) вірний** тоді, коли порівнюване значення входить у множину заданих значень. Одночасно існує протилежний предикат **NOT IN (множина)**, що щира тоді, коли порівнюване значення не входить у задану безліч.

**Приклад:** Скласти список викладачів -доцентів

SELECT Викладачі.\*

FROM Викладачі

WHERE Посада IN ("Доцент");

- Відповідність шаблону – перевіряється, чи відповідає деяке строкове значення заданому шаблону. – предикати порівняння зі зразком **LIKE** і **NOT LIKE**. Предикат **LIKE** вимагає завдання шаблону, з яким порівнюється задане значення, предикат вірний, якщо порівнюване значення відповідає шаблону, і помилковий у протилежному випадку. Предикат **NOT LIKE** має протилежний сенс.

По стандарті в шаблон можуть бути включені спеціальні символи:

- символ підкреслення ( `_` ) — для позначення будь-якого одиночного символу;
- символ відсотка ( `%` ) — для позначення будь-якої довільної послідовності символів;
- інші символи, задані в шаблоні, позначають самих себе.

*Приклад:* Список усіх співробітників, які мешкають на вулиці Горького

```
SELECT Викладачі.*
```

```
FROM Викладачі
```

```
WHERE Адреса LIKE "%Горького%";
```

— адреса, що містить набір символів Горького (при цьому припустимо використання символу `*` для позначення будь-якої кількості символів)

- Значення NULL- перевіряється, чи містить заданий стовпець визначник NULL

Предикат порівняння з невизначеним значенням IS NULL.

Якщо в даному кортежі (у даному рядку) зазначений атрибут має невизначене значення, то предикат IS NULL приймає значення «Істина» (TRUE), а предикат IS NOT NULL — «Неправда» (FALSE), у протилежному випадку предикат IS NULL приймає значення «Неправда», а предикат IS NOT NULL приймає значення «Істина».

*Приклад:* Список усіх співробітників, що мають телефони

```
SELECT Викладачі.*
```

```
FROM Викладачі
```

```
WHERE Телефон IS NOT NULL;
```

- Предикати існування **EXIST** і неіснування **NOT EXIST**. Ці предикати відносяться до убудованих подзапросам,
- Для побудови більш складних запитів предикати можуть бути побудовані з використанням логічних операторів AND, OR, NOT.
- Обчислення у виражах виконуються за наступними правилами:
  - вираз обчислюється ліворуч праворуч;
  - першими обчислюються вираження в дужках;
  - оператори NOT обчислюються до виконання операторів AND або OR;
  - оператори AND обчислюються до виконання операторів OR;
  - для усунення яких-небудь неоднозначностей рекомендується використовувати дужки.

**Приклад:** Перелічити усіх викладачів – доцентів кафедри Фізики  
 SELECT Викладачі.\*

FROM Викладачі

WHERE Кафедра= «Фізики» and Посада = «Доцент»;

### **Пропозиція ORDER BY**

Сортування результатів **ORDER BY** <Список полів> використовується у раз необхідності упорядковування рядків результуючої таблиці, тому що у загальному випадку результати запитів не упорядковані. За замовчуванням сортування встановлене по зростанню **ASC** для сортування по зменшенню **DESC**.

**Приклад:** Упорядкування результатів запиту по декількох полях зі зростанням або убубанням (ключові слова ASC, DESC):

У результаті одержимо таблицю, у якій рядки йдуть у порядку зростання за значення поля DNUM, а рядки, з однаковим значенням DNUM йдуть у порядку убубання значення поля VOLUME

SELECT PD.PNUM, PD.DNUM, PD.VOLUME

FROM PD

ORDER BY DNUM ASC, VOLUME DESC;

### **Пропозиція Group by**

**GROUP BY** <Список полів>

При відображенні таблиці групуються рядки, що мають однакове значення заданого стовпця. Це пропозиція найчастіше використовується з агрегатними функціями.

#### **Приклад:**

```
SELECT Студенти.Група, Count(Студенти.Прізвище) AS Прі-
звище
FROM Студенти
GROUP BY Студенти.Група;
```

### **Застосування) функцій дати та часу**

Для використання в умовах добору операцій з датами передбачені функції (Таблиця 1.2).

Таблиця 1.1 – Функції дати

| Імя функції | Повертає:                                              |
|-------------|--------------------------------------------------------|
| Date()      | поточну дату                                           |
| DatePart()  | зазначена частина часу                                 |
| DateDiff()  | тривалість інтервалу часу між двома зазначеними датами |
| DateAdd()   | дату, до якої додано зазначений інтервал часу          |
| Day()       | день місяця, який відповідає зазначеній даті           |
| Weekday()   | номер дня тижня, який відповідає зазначеній даті       |
| Month()     | Номер місяцю, який відповідає зазначеній даті          |
| Year()      | рік, який відповідає зазначеній даті                   |

У зазначених функціях використовуються наступні позначення параметрів:

| Параметр | опис      |
|----------|-----------|
| уууу     | рік       |
| q        | квартал   |
| m        | місяць    |
| y        | день року |
| d        | день      |

|    |            |
|----|------------|
| W  | ДЕНЬ ТИЖНЯ |
| WW | ТИЖДЕНЬ    |

Приклади використання функцій дати в умовах добору

#12/12/19960# - введення символу # необов'язково

або 12.12.1978 відбираються записи, які відповідають зазначеній у полі даті.

*Year ([ім'я поля ])>1960* - відбираються записи, для яких рік у відповідному полі більший за 1960.

*Format(Now(); "ww")* - функція *Format()* використовується для відображення номера тижня року для поточної дати, де *ww* є номер тижня у діапазоні от 1 до 53.

*DateAdd("y"; -10; [Дата])* - функція *DateAdd()* використовується для відображення дати, яка на 10 днів передуює даті, що зазначена у полі «Дата».

*DatePart("q"; [ДатаРазмещения]) = 4* - відбирання записів, які у полі «ДатаРазмещения» відповідають 4 кварталу.

Для більш докладного ознайомлення з правилами складання умов скористайтесь підручниками з мови SQL.

**Приклад:** Підрахувати кількість студентів які народились у березні

Select count (\*)

From Студенти

Where month([День народження]) = 3;

### **Застосування агрегатних (узагальнюючих) функцій**

У SQL додані додаткові функції, що дозволяють обчислювати узагальнені групові значення. Для застосування агрегатних функцій передбачається попередня операція угруповання.. Можна застосовувати агрегатні функції також і без операції попереднього угруповання, у цьому випадку усе відношення розглядається як одна група і для цієї групи можна обчислити одне значення на групу. Агрегатні функції повертають **одне** значення для всієї групи.

Стандарт ISO передбачає 5 узагальнюючих (агрегатних) функцій:

**COUNT** - підрахунок кількості рядків або непорожніх значень полів, що вибрав запит. Застосовується для числових і нечислових полів.

**Приклад:** Підрахувати кількість найменувань товару по кожній групі

```
SELECT P1.Найменування, Count(*) AS Кількість
FROM P1
GROUP BY P1.Група;
```

**SUM** – визначається сума всіх обраних значень даного поля, застосовується тільки для числових полів

*Приклад:* Визначити кількість кожного товару на складі

```
SELECT Найменування, sum(Кількість) as Всього
FROM склад
GROUP BY Найменування;
```

Всього – у данному випадку це і'мя поля, у якому відображатиметься сума значень поля Кількість для кожної групи у полі Найменування.

**AVG** - середнеарифметичне значення всіх обраних значень даного поля, застосовується тільки для числових полів.

*Приклад:* Визначити середню ціну кожного товару на складі

```
SELECT Найменування, avg(ціна) as ср_ціна
FROM склад
GROUP BY Найменування;
```

**MIN** – визначається найменше з всіх обраних значень даного поля, застосовується для числових і нечислових полів.

*Приклад:* Визначити найменування товару, який має мінімальну ціну

```
SELECT Найменування, min(ціна) as min_ціна
FROM склад;
```

**MAX** - найбільше з всіх обраних значень даного поля, застосовується для числових і нечислових полів

Крім функції COUNT (\*) при обчисленні результату спочатку виключаються порожні рядки, після чого зазначена операція застосовується тільки до конкретних значень стовпця, що залишилися. Варіант COUNT (\*) є особливим випадком функції COUNT ( ) – у цьому випадку підраховуються всі рядки в результуючій таблиці, незалежно від того є чи порожні, дублюючі або інші значення.

Якщо до застосування узагальнюючої функції необхідно виключо-

чити дублюючі значення, впливає перед ім'ям стовпця у визначенні функції використовувати ключове слово **DISTING**. Це слово не може бути використане з функціями **MAX** і **MIN**. Однак його використання перед функціями **SUM** і **AVG** може привести до зміни результату. Крім того ключове поле **DISTING** може бути використане в запиті тільки один раз.

Узагальнюючі функції можуть бути використані тільки в списку пропозиції **SELECT** або **HAVING**.

Якщо список у **SELECT** містить узагальнюючі функції, і в тексті запиту немає пропозиції **GROUP BY**, то жоден з елементів списку пропозиції **SELECT** не може містити посилань на стовпці, крім випадку, коли це посилання є аргументів узагальнюючої функції.

**Приклад:** Визначити кількість студентів, що успішно склали сесію  
**SELECT** COUNT(\*) as Кількість  
**FROM** Сесія  
**WHERE** Оцінка > 2;

**Приклад:** Визначити кількість студентів, що складали сесію і середній бал на сесії  
**SELECT** COUNT(DISTING, Код\_Студента) as Кількість, AVG(Оцінка)  
 as CP\_бал  
**FROM** Сесія  
**WHERE** Оцінка IS NOT NULL;

Агрегатні функції не можна використовувати в пропозиції **WHERE**, тому що предикати оцінюються в термінах одиночного рядка, а агрегатні функції — у термінах груп рядків.

### **Пропозиція HAVING**

У розділі **HAVING** задаються обмеження на виконання групування.  
**HAVING** <Предикат-умова для групи>.

Пропозиція **HAVING** використовується разом із пропозицією **GROUP BY** для завдання обмежень на добір груп, що будуть поміщені в результуючу таблицю. Пропозиція **HAVING** використовується для фільтрації груп, на відміну від пропозиції **WHERE**, що використовується для фільтрації рядків.

У відповідності зі стандартом **ISO** усі імена стовпців пропозиції

**HAVING** мають зазначатись у пропозиції **GROUP BY** або застосовуватися в узагальнюючій функції.

*Приклад:* визначити номери деталей, якщо сумарний розмір поставки перевищує 400.

```
SELECT DetNum, SUM(Det.Volume) AS SUM
From Det
GROUP BY DetNum
HAVING SUM(Det.Volume)>400;
```

### Вкладені запити

Мова **SQL** дозволяє використовувати один запит у якості допоміжного для визначення результатів другого запиту.

Зовнішній оператор **SELECT** використовує результат внутрішнього запиту для визначення змісту остаточного результату всієї операції. Внутрішні запити можуть бути поміщені пропозиції **WHERE** і **HAVING** зовнішнього оператора **SELECT**, крім того внутрішні запити можуть використовуватися в операторах **INSERT**, **UPDATE**, **DELETE**

Існує три типи внутрішніх запитів:

- скалярний підзапрос повертає єдине значення. Скалярний запит може використовуватися скрізь, де необхідно одержати єдине значення
- рядковий підзапрос повертає значення декількох стовпців таблиці, але у виді єдиного рядка. Рядковий підзапит може використовуватися скрізь, де застосовується конструктор рядкових значень.
- табличний підзапрос повертає значення одного або більш стовпців таблиці, розміщені більш ніж в одному рядку. Табличний підзапрос може використовуватися скрізь, де допускається вказувати таблицю.

Вкладений запит являє собою інструмент створення тимчасової таблиці, зміст якої витягається й обробляється зовнішнім оператором. Вкладений запит може вказуватися безпосередньо після оператора порівняння (=, <, > і т.д.) у пропозиціях **WHERE** і **HAVING**. Текст підзапроса має розміщуватись у дужках.

*Приклад:* Визначити список постачальників, статус яких менше максимального статусу в таблиці постачальників (порівняння з підзапро-

com):

```
SELECT *
FROM P
WHERE P.STATYS <
      (SELECT MAX(P.STATUS)
FROM P);
```

Т.к. поле P.STATUS порівнюється з результатом підзапроса, те підзапрос повинний бути сформульований так, щоб повертати єдине значення..

### ***Використання вкладеного запиту з узагальнюючими функціями***

**Приклад** Скласти список усіх співробітників, що одержують зарплату вище середньої по підприємству з указівкою, на скількох цих зарплат більше середньої.

```
SELECT
      Кодработника, ФИО, Платня, (SELECT avg(зарплата) FROM
Штат;)
FROM Штат
WHERE Платня >
      (SELECT avg(зарплата)
FROM Штат);
```

У цьому випадку необхідно використовувати вкладений запит, тому що не можна використовувати узагальнюючі функції в пропозиції WHERE. Вкладений запит повертає єдине значення.

### ***Використання предиката IN.***

Використовується у випадку, коли вкладений запит повертає таблицю

**Приклад:** Одержати список постачальників, що поставляють деталь номер 2:

```
SELECT *
FROM P
WHERE P.PNUM IN
      (SELECT DISTINCT PD.PNUM
FROM PD
WHERE PD.DNUM = 2);
```

У даному випадку вкладений вкладений запит може повертати таблицю, що містить кілька рядків.

### **Використання предиката *EXIST*.**

**Приклад:** Одержати список постачальників, що поставляють деталь номер 2:

```
SELECT *
FROM P
WHERE EXIST
    (SELECT *
     FROM PD
     WHERE
     PD.PNUM = P.PNUM AND PD.DNUM = 2);
```

Вкладений запит повертає таблицю значень.

### **Використання вложеного запиту з перевіркою рівності**

**Приклад:** Скласти список персоналу, що працює у філії, розташованій за адресою вул.Горького, 5

```
SELECT *
FROM штат
WHERE Кодфилиала=
    (SELECT Кодфилиала
     FROM Філія
     WHERE адреса ='вул.Горького,5');
```

В внутрішньому запиті визначається номер філії, яка розташована за зазначеною адресою. В зовнішньому запиті визначається список працівників даної філії.

Разом з внутрішнім запитом можна використовувати предикат EXISTS, що повертає істину, якщо висновок підзапаса не порожній.

### **Багатотабличні запити**

У мові **SQL** операція з'єднання використовується для об'єднання інформації з двох таблиць шляхом утворення двох зв'язаних рядків, обраних з кожної таблиці.

Для виконання з'єднання досить у пропозиції **FROM** вказати імена декількох таблиць через кому, а в пропозиції **WHERE** вказати стовпці, використовувані для з'єднання таблиць. При цьому якщо у схемі бази даних встановлений зв'язок між таблицями, які використовуються у запиті, то при з'єднанні здійснюється один з типів з'єднання таблиць, що передбачено у мові **SQL**. У **MS Access** за замовчуванням

здійснюється внутрішнє з'єднання. Якщо у схемі бази даних зв'язок не встановлений, то здійснюється декартове множення таблиць за з'єднаними рядками.

Найчастіше багатотабличні запити застосовуються для отримання інформації з таблиць, що мають зв'язок 1:M.

### ***Синтаксис з'єднаних таблиць***

При виконанні операції з'єднання дані з двох таблиць комбінуються з утворенням зв'язаних рядків, у яких значення порівнянних

Синтаксис з'єднаної таблиці має вигляд:

Таблиця А [тип з'єднання] JOIN Таблиця В

**JOIN ... ON** - З'єднання за допомогою предиката з'єднує рядки таблиць А і В за допомогою зазначеного предиката.

**OUTER** - Ключове слово OUTER (зовнішній) не є обов'язковим, воно не використовується ні в яких операціях з даними.

**INNER** - Тип з'єднання "внутрішнє". Внутрішній тип з'єднання використовується за замовчуванням, коли тип явно не заданий. У таблицях А і В з'єднуються тільки ті рядки, для яких знайдено збіг.

**Приклад:** Створити список студентів, що мешкають у гуртожитку, із указівкою всіх даних студентів

```
SELECT Студенти.*, Гуртожиток.Гуртожиток, Гуртожиток.Кімната
FROM Студенти INNER JOIN Гуртожиток ON Студенти.Код_студента = Гуртожиток.[Код студента];
```

**LEFT (OUTER)** - Тип з'єднання "ліве (зовнішнє)". Ліве з'єднання таблиць А і В містить у собі всі рядки з лівої таблиці А і ті рядки з правої таблиці В, для яких виявлений збіг. Для рядків з таблиці А, для яких не знайдено відповідності в таблиці В, у стовпці, що витягаються з таблиці В, заносяться значення NULL.

**Приклад:** Перелічити філії компанії й об'єкти, що здаються в оренду, що розташовані в тому самому місті, а також інші філії компанії

```
SELECT b.*, p.*
```

```
FROM філія b LEFT JOIN Оренда a on b.city=a.city
```

У результуючу таблицю попадають не тільки ті рядки, що знайшли собі пари в таблицях b і p, але і ті рядки з лівої таблиці, що з'єднується, що не знайшли відповідності в другій таблиці. У цьому рядку всі поля правої таблиці заповнюються значеннями NULL.

**RIGHT (OUTER)** - Тип з'єднання "праве (зовнішнє)". Праве з'єднання таблиць А і В містить у собі всі рядки з правої таблиці В і ті рядки з лівої таблиці А, для яких виявлений збіг. Для рядків з таблиці В, для яких не знайдено відповідності в таблиці А, у стовпці, що витягаються з таблиці А заносяться значення NULL.

**FULL (OUTER)** - Тип з'єднання "повне (зовнішнє)". Це комбінація лівого і правого з'єднань. У повне з'єднання включаються всі рядки з обох таблиць. Для співпадаючих рядків полючи заповнюються реальними значеннями, для незбіжних рядків полючи заповнюються відповідно до правил лівого і правого з'єднань.

**UNION JOIN** - З'єднання об'єднання є зворотним стосовно внутрішнього з'єднання. Воно включає тільки ті рядки з таблиць А і В, для яких не знайдено збігів. У них використовуються значення NULL для стовпців, отриманих з іншої таблиці. Якщо взяти повне зовнішнє з'єднання і видалити з нього рядка, отримані в результаті внутрішнього з'єднання, то вийде з'єднання об'єднання.

## 2. ЛАБОРАТОРНЕ ЗАВДАННЯ

Виконання завдань можливо:

1. У середовищі СУБД Access – режим SQL:

- завантажити СУБД Access
- закладинка Запрос- Конструктор Запросов – пкм Режим SQL- у вікні ввести потрібний код запиту.

2. У середовищі HeidiSQL – графічної оболонки СУБД MySQL

3. У командному рядку **MySQL 5.7 Command Line Client**

- Запустити **MySQL 5.7 Command Line Client**
- Ввести пароль
- Ввести команду set names cp866;- для коректного відображення кирилиці у робочій базі даних.

Середовище виконання лабораторної роботи студент обирає самостійно.

Всі створені запити зберегти за імям відповідного завдання, наприклад, запит у завданні 2.1.1 зберегти за ім'ям 2\_1\_1.

### Завдання 2. 1 Створення простого запиту на мові SQL

2.1.1. Визначити номери домашніх телефонів кожного викладача, у списку мають вказуватись й викладачі, у які не мають домашніх телефонів.

2.1.2. Створити список викладачів кафедри Радіотехніки, які не мають домашнього телефону, список відсортувати за алфавітом..

2.1.3. Створити список студентів групи РП 312, список має бути відсортований за алфавітом.

## **Завдання 2. 2 Створення SQL запитів з використанням агрегатних функцій**

2.2.1. Визначити кількість студентів 1984 року народження.

2.2.2. Визначити кількість викладачів, які народились між 1940 та 1950 роками.

2.2.3. Визначити кількість студентів, яким нарахована стипендія у кожному місяці та її середній розмір.

2.2.4. Підрахувати кількість студентів, які мешкають у кожному гуртожитку.

2.2.5. Визначити кількість студентів, що мешкають у кожній кімнаті кожного гуртожитку.

## **Завдання 2. 3 Створення багатотабличного запиту**

2.3.1. Створити список мешканців кожної кімнати у гуртожитку 1.

2.3.2. Визначити список студентів, які отримали хоча б одну двійку на іспитах.

2.3.3. Визначити середній бал з дисципліни Архітектура ЕОМ.

2.3.4. Змінити запит за п. 2.3.3 таким чином, що б в результаті виконання запиту визначався середній бал з кожної дисципліни.

2.3.5. Визначити середній розмір стипендії у кожній групі по місяцях.

## **Завдання 2. 4 Використання вкладених запитів**

2.4.1. Визначити список студентів, які мають оцінку з Архітектури ЕОМ вище середнього балу з предмету.

2.4.2. Визначити список студентів, які не мешкають у гуртожитку

2.4.3. Визначити, які студенти, що мешкають у гуртожитку, склали іспити на 2

## **Завдання 2. 5 Створення запиту на оновлення**

2.5.1. Змінити назву дисципліни Архітектура ЕОМ на Архитектура.

2.5.2. Змінити назву групи РП-311 на власну.

2.5.3. Змінити прізвище викладача Аверкова на Андрієнко

## **Завдання 2. 6 Завдання для самостійної розробки запитів**

### **Варіант 1.**

- 1.1. Знайти, на якій кафедрі не має жодного професора.
- 1.2. У якому гуртожитку проживають студенти з кожної групи?
- 1.3. Підрахувати кількість доцентів на усіх кафедрах.
- 1.4. Скільки доцентів приймало іспити протягом сесії?
- 1.5. Знайти викладача, якій поставив найбільшу кількість 2.
- 1.6. Знайти, в який день на сесії складали іспит більше двох

груп.

1.7. Знайти, в який день на сесії всі студенти отримали найменший середній бал.

1.8. Знайти групу в якій стипендію будуть одержувати найбільша кількість студентів.

1.9. У якому гуртожитку не буду отримувати стипендію найбільша кількість студентів?

### **Варіант 2.**

- 2.1. Знайти, на якій кафедрі не має жодного доцента.
- 2.2. У якому гуртожитку проживають студенти не з усіх груп?
- 2.3. Підрахувати кількість професорів на усіх кафедрах.
- 2.4. Підрахувати кількість доцентів, які приймали іспити на сесії.

2.5. Знайти викладача, якій поставив найбільшу кількість 5.

2.6. Знайти в який день на сесії складала іспит тільки одна група.

2.7. Знайти, в який день на сесії складали іспит найбільша кількість студентів.

2.8. Знайти групу в якій стипендію будуть одержувати найменша кількість студентів.

2.9. У якому гуртожитку не будуть отримувати стипендію найменша більша кількість студентів?

### **Варіант 3.**

- 3.1. Знайти, на якій кафедрі не має жодного старшого викладача.

3.2. У якому гуртожитку проживає найбільша кількість студентів?

3.3. Підрахувати кількість старших викладачів на усіх кафедрах.

3.4. Підрахувати кількість професорів, які приймали іспити протягом сесії.

3.5. Знайти викладача, якій на сесії поставив 4 більше за всіх.

3.6. Знайти, в який день на сесії складала іспит найменша кількість студентів.

3.7. Знайти, в який групі студенти отримали найбільшу кількість відмінних оцінок.

3.8. Знайти групу в якій середня стипендія за вересень буде найбільша.

3.9. У якому гуртожитку студенти будуть отримувати найменшу середню стипендію за жовтень?

#### **Варіант 4.**

4.1. Знайти, на якій кафедрі є викладачі професори.

4.2. У якому гуртожитку проживає найменша кількість студентів?

4.3. Підрахувати кількість викладачів пенсійного віку на усіх кафедрах.

4.4. Підрахувати кількість старших викладачів, які приймали іспити протягом сесії.

4.5. Знайти викладача, якій на сесії поставив найбільше 3.

4.6. Знайти, в який день на сесії всі студенти отримали найменшу кількість 5.

4.7. Знайти, в який групі студенти отримали найбільший середній бал.

4.8. Знайти групу в якій середня стипендія за жовтень буде найменша.

4.9. У якому гуртожитку студенти будуть отримувати найбільшу середню меншу стипендію за жовтень?

#### **Варіант 5.**

5.1. Знайти, на яких кафедрах є викладачі професори.

5.2. У якому гуртожитку проживає найменша кількість студентів, які отримали на сесії хоча б одну 2?

5.3. Підрахувати на якій кафедрі найменша кількість викладачів.

5.4. Підрахувати кількість іспитів, які склали студенти протягом сесії.

5.5. Знайти викладача, якій на сесії не поставив жодної 2.

5.6. Знайти, в який день на сесії студенти отримали найбільшу кількість 2.

5.7. Знайти, в якій групі студенти отримали найбільший середній бал.

5.8. Знайти групу в якій найбільша кількість студентів, що отримують стипендію протягом всього семестру.

5.9. У якому гуртожитку проживають студенти що не будуть отримувати стипендію протягом семестру?

### **Варіант 6.**

6.1. Знайти, скільки професорів на всіх кафедрах.

6.2. У якому гуртожитку проживає найменша кількість студентів, які отримали на сесії тільки 4 та 5?

6.3. Підрахувати на якій кафедрі найбільша кількість викладачів.

6.4. Підрахувати кількість іспитів, які прийняли викладачі протягом сесії.

6.5. Знайти викладача, якій на сесії не поставив жодної 5.

6.6. Знайти, в який день на сесії всі студенти отримали найбільший середній бал.

6.7. Знайти, в який групі найбільшу кількість студентів, які склали іспити тільки на 4 та 5.

6.8. Знайти групу в якій найменша кількість студентів, що отримують стипендію протягом всього семестру.

6.9. Підрахувати кількість студентів у кожному гуртожитку, що отримувати стипендію більшу за середню у вересні.

## **3. ЗМІСТ ЗВІТУ**

3.1.Тема та мета лабораторної роботи.

3.2.Тексти всіх SQL запитів та результати їх виконання.

3.3.Відповіді на контрольні питання.

#### **4. КОНТРОЛЬНІ ПИТАННЯ**

- 4.1. Основні пропозиції оператора SELECT ?
- 4.2. Як у мові SQL визначаються умови добору рядків?
- 4.3. Як у мові SQL визначається сортування даних?
- 4.4. Як створити обчислювальне поле у SQL запиті?
- 4.5. Призначення пропозиції Having у операаторі SELECT
- 4.6. Які умови можна використовувати у пропозиції WHERE?
- 4.7. Як з'єднати дані двох таблиць, які мають однаковий стовпчик?
- 4.8. Що таке вкладений запит? У яких випадках використовується вкладений запит?

#### **5. РЕКОМЕНДОВАНА ЛІТЕРАТУРА**

- 5.1. <http://kafinfo.ru/petrsu.ru/mysql/index.htm>
- 5.2. <http://www.site-do.ru/db/sql3.php>
- 5.3. <http://phpclub.ru/mysql/doc>
- 5.4. <http://www.sql.ru/>
- 5.5. <http://www.cyberforum.ru>