Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

А. Г. Морозова
Є. С. Меняйлов
К. М. Руккас

ВСТУП ДО SQL БАЗ ДАНИХ

Методичні рекомендації
до виконання індивідуальних (розрахунково-графічних) робіт
для здобувачів вищої освіти факультету математики і інформатики
першого (бакалаврського) рівня вищої освіти

Електронне видання

Харків – 2024

УДК 004.62+ 004.65+004.43
М 80

Рецензенти:

О. Г. Толстолузька – доктор технічних наук, старший науковий співробітник, професор кафедри теоретичної та прикладної системотехніки факультету комп'ютерних наук Харківського національного університету імені В. Н. Каразіна;

А. Г. Чухрай – доктор технічних наук, професор, завідувач кафедри математичного моделювання та штучного інтелекту факультету систем управління літальних апаратів Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків.

Затверджено до розміщення в мережі Інтернет рішенням Науково-методичної ради Харківського національного університету імені В. Н. Каразіна (протокол № 3 від 19 грудня 2023 року)

Морозова А. Г.

М 80 Вступ до SQL баз даних : методичні рекомендації до виконання індивідуальних (розрахунково-графічних) робіт для здобувачів вищої освіти факультету математики і інформатики першого (бакалаврського) рівня вищої освіти [Електронне видання] / А. Г. Морозова, Є. С. Меняйлов, К. М. Руккас. – Харків : ХНУ імені В. Н. Каразіна, 2024. – (PDF 42 с.)

Методичні рекомендації містять теоретичний матеріал, завдання для індивідуальних (розрахунково-графічних) робіт, зразки виконання розрахунково-графічних робіт, питання для самоконтролю, завдання для самостійної роботи, рекомендовану літературу.

Навчально-методичне видання призначається для здобувачів вищої освіти факультету математики і інформатики першого (бакалаврського) рівня вищої освіти, які вивчають дисципліну «Вступ до SQL баз даних».

УДК 004.62+ 004.65+004.43

Електронне навчальне видання комбінованого використання
Можна використовувати в локальному та мережному режимі

Морозова Анастасія Геннадіївна
Меняйлов Євген Сергійович
Руккас Кирило Маркович

ВСТУП ДО SQL БАЗ ДАНИХ

Методичні рекомендації
до виконання індивідуальних (розрахунково-графічних) робіт
для здобувачів вищої освіти факультету математики і інформатики
першого (бакалаврського) рівня вищої освіти

Коректор *Л. Є. Стешенко*
Комп'ютерне верстання *В. В. Савінкова*

Підписано до розміщення 19.12.23. Гарнітура Times New Roman.
Ум. друк. арк. 1,81. Обсяг 0,750 Мб. Зам. № 230/23.

Харківський національний університет імені В. Н. Каразіна,
61022, м. Харків, майдан Свободи, 4.
Свідоцтво суб'єкта видавничої справи ДК № 3367 від 13.01.2009
Видавництво ХНУ імені В. Н. Каразіна

Зміст

Вступ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ до виконання індивідуальних (розрахунково-графічних) робіт з дисципліни «Вступ до SQL баз даних» для здобувачів вищої освіти факультету математики і інформатики першого (бакалаврського) рівня вищої освіти складені відповідно до робочої програми дисципліни «Вступ до SQL баз даних».

Неможливо уявити сучасну інформаційну систему без використання баз даних. Реляційні бази даних відіграють ключову роль у забезпеченні ефективних та надійних структур даних для збереження інформації. Виконання запропонованих у методичних рекомендаціях завдань дозволить студентам на професійному рівні навчитися проектувати схеми реляційних баз даних, використовувати мову SQL для визначення об’єктів бази даних та маніпулювання даними; нормалізувати реляційну базу даних та проводити аналіз на наявність нормальних форм в існуючий базі даних; розробляти програмне забезпечення, що буде використовувати базу даних як сховище.

Перед виконанням індивідуальних робіт студенти зобов'язані проробити основні теоретичні положення та ознайомитися з прикладами розв’язування схожих задач.

Індивідуальна робота № 1
ПРОЕКТУВАННЯ ТА СТВОРЕННЯ СХЕМИ БД. ОДНОТАБЛИЧНІ ТА БАГАТОТАБЛИЧНІ ЗАПИТИ ДО БД

Теоретичний матеріал

Інструкція для створення бази даних

```
CREATE DATABASE database_name
```

Приклад:

```
CREATE DATABASE Films
GO
```

Інструкція для видалення бази даних:

```
DROP DATABASE Films
GO
```

Створення таблиць, первинних та зовнішніх ключів

Для створення таблиць у SQL Server використовується DDL оператор CREATE TABLE.

Синтаксис оператора:

```
CREATE TABLE [[database.] owner.] table_name
( Column_name    datatype     [not null | null ]
                              IDENTITY [(seed, increment)]
[constraint]
[, column_name datatype    [not null   |   null ]
                              IDENTITY[(seed,
increment)]]].[constraint]...)
```

Приклади:

Створення таблиці ArchivType

```
CREATE TABLE ArchivType(
    archivType_id  INT NOT NULL IDENTITY(1,1),
    archiv_name    CHAR(20)
);
```

Створення таблиці ArchivType та первинного ключа archivType_id_pk за стовпцем archivType_id

```
CREATE TABLE ArchivType(
    archivType_id  INT NOT NULL IDENTITY(1,1),
    archiv_name    CHAR(20),
  CONSTRAINT archivType_id_pk PRIMARY KEY (archivType_id)
 );
```

Створення таблиці Archiv з первинним ключем archiv_id_pk та зовнішнім ключем archiv_fk

```
CREATE TABLE Archiv(
    archiv_id  INT IDENTITY(1,1) PRIMARY KEY CLUSTERED not
    null,
    is_present BIT,
    archivType INT,
    filmName   INT,

    CONSTRAINT archiv_fk FOREIGN KEY (archivType) REFERENCES
    ArchivType(ArchivType_id),
    CONSTRAINT film_fk FOREIGN KEY (filmName) REFERENCES
    Film(film_id)
);
```

Для додавання зовнішнього, первинного та стовпців до таблиці використовується оператор **ALTER TABLE.**

Додавання зовнішнього ключа:

```
ALTER TABLE table_name
ADD  CONSTRAINT constraint_name
FOREIGN KEY (column_name) REFERENCES ref_table(ref_column)
```

Додавання первинного ключа:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
PRIMARY KEY [CLUSTERED] (column_name)
```

Видалення зовнішнього (первинного) ключа:

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name
```

Додавання стовпця до таблиці:

```
ALTER TABLE table_name
ADD column_name  data_type
```

Видалення стовпця таблиці:

```
ALTER TABLE table_name
DROP COLUMN column_name
```

Приклад створення первинного ключа archivType_id_pk

```
ALTER TABLE ArchivType ADD CONSTRAINT archivtype_id_pk
PRIMARY KEY CLUSTERED (archivType_id)
```

Приклад створення зовнішнього ключа archiv_fk

```
ALTER TABLE Archiv ADD CONSTRAINT archiv_fk
FOREIGN KEY (archivType)
REFERENCES ArchivType (archivType_id)
```

Вставка даних (додавання нового запису до таблиці)

Для вставки даних у таблицю використовується SQL-оператор INSERT. Цей оператор має дві форми, залежно від того, чи всім стовпцям таблиці надаються значення.

Загальний вигляд інструкції INSERT:

```
INSERT INTO table_name [ (column_name, column_name, ...) ]
VALUES (value, value, ..) [, (value, value, …)…]
```

1. **Перша форма оператора INSERT**

Додавання запису із зазначенням всіх стовпців (у тому порядку, в якому вони були вказані в інструкції CREATE TABLE).

```
INSERT INTO ARTIST VALUES (3, 'Miro', 'Spanish', 1870, 1950);
```

Зверніть увагу, що значення типу INTEGER не беруться у лапки, на відміну від CHAR та VARCHAR.

Якщо для стовпців дані відсутні, можна використовувати ключове слово NULL для значень, що пропускаються:

```
INSERT INTO ARTIST VALUES (15, 'Matisse', 'French', NULL, NULL);
```

ВАЖЛИВО!!!! Ключове слово NULL пишеться без лапок!!!!

2. **Друга форма оператора INSERT**

Друга форма допускає пропуск значень деяких стовпців або зміна порядку проходження стовпців у таблиці при додаванні, передбачає перерахування імен стовпців, яким будуть присвоєні значення. Така форма використовується також для додавання рядків до таблиці, де є поля, що автоматично визначаються (IDENTITY). Наприклад, наступний оператор додає до таблиці ARTIST рядок, у якому стовпцям ArtistID, Name і Nationality присвоєно значення, а стовпці BirthDate і DeceasedDate залишені порожніми:

```
INSERT INTO ARTIST (Name, Nationality, ArtistID) VALUES
('Malevich', 'Ukrainian', 20);
```

Якщо для будь-якого стовпця при створенні таблиці було визначено початкове значення, тоді незважаючи на те, що в операторі INSERT значення цього стовпця може і не надаватися, СУБД все одно встановить його значення за замовчуванням (DEFAULT).

Існує кілька додаткових зауважень, які слід зробити з приводу другої форми оператора INSERT.

- **По-перше**, порядок, у якому перераховуються значення для стовпців (value), повинен відповідати порядку слідування їх імен. У попередньому прикладі імена стовпців йдуть у порядку (Name, Nationality, ArtistID), тому спочатку має бути вказано значення стовпця Name, потім Nationality і, нарешті, ArtistID. Слід зазначити, що стовпці таблиці йдуть в іншому порядку.
- **По-друге**, щоб вставка була виконана, необхідно встановити значення всіх ОБОВ'ЯЗКОВИХ стовпців (NOT NULL).

3. Копіювання даних з іншої таблиці

Якщо потрібно скопіювати велику кількість даних із іншої таблиці, їх значення можна отримати за допомогою оператора SELECT.

Наприклад, наступний оператор копіює значення стовпців ArtistID, Name і Nationality з таблиці ARTIST до таблиці ARTIST_NEW (яка має бути створена заздалегідь):

```
INSERT INTO ARTIST_NEW (ArtistID, Name, Nationality) SELECT
ArtistID, Name, Nationality FROM ARTIST;
```

4. Додавання декілька рядків операторі INSERT

Якщо потрібно додати відразу кілька рядків, тоді і в першій і другій формі значення другого і наступних рядків вказується через кому, наприклад:

```
INSERT INTO ARTIST VALUES
(15, 'Matisse', 'French', NULL, NULL),
(4, 'Murashko', 'Ukrainian', 1854, 1900),
(5, 'Frings', 'US', 1700, 1800)
```

Зміна (Редагування) даних

Значення існуючих даних можна змінити за допомогою SQL-оператора UPDATE.

Загальний вигляд інструкції UPDATE:

```
UPDATE table_name
SET column_name = new_value
[, column_name = new_value …]
[WHERE condition]
```

Потрібно брати до уваги, що це потужна команда, і її слід використовувати з обережністю. Розглянемо наступний приклад:

```
UPDATE WORK
SET Copy = '99/100'
WHERE WorkID = 506;
```

Цей оператор встановлює значення стовпця `Copy` у рядку твору з номером 506 рівним '99/100'. Тепер подивимося, що мало на увазі, коли йшлося про обе-

режність. Припустимо, що збираючись зробити цю зміну, ви зробили помилку і забули про пропозицію WHERE. Таким чином, було передано на виконання СУБД наступний оператор

```
UPDATE WORK
SET Copy = '99/100'
```

В результаті, якщо не порушуються обмеження стовпця Copy (наприклад, обмеження унікальності), то у всій таблиці WORK у стовпця Copy буде одне значення – '99/100'.

Підсумок такий: оператор UPDATE дуже потужний і простий у використанні, але може призвести до сумних наслідків.

За допомогою оператора UPDATE можна оновлювати кілька стовпців за один прийом. Це демонструє такий приклад:

```
UPDATE WORK
SET Copy = '99/100', Description = 'Very nice'
WHERE WorkID = 506;
```

Ця команда змінює значення стовпців Copy та Description для зазначеної роботи.

Умов може також бути кілька з'єднаних за допомогою ключових слів AND (і) або OR (або), наприклад:

```
UPDATE WORK
SET Copy = '99/100'
WHERE title = 'Mystic Fabric'
AND ArtistID=14;
```

Видалення рядків (строк)

Для видалення рядків служить SQL-оператор DELETE. До нього відносяться ті самі застереження, що і до оператора UPDATE. Він оманливе простий у використанні, і необережне застосування може призвести до найнесподіваніших наслідків.

Загальний вигляд інструкції DELETE:

```
DELETE FROM table_name
[WHERE condition]
```

Наступний оператор видаляє з таблиці ARTIST рядок, в якому стовпець ArtistID дорівнює 15:

```
DELETE
FROM ARTIST
WHERE ArtistID = 15;
```

Як і у випадку з оператором UPDATE, якщо ви забудете вказати умову WHERE, наслідки можуть бути катастрофічними. Наприклад, наступний оператор видалить усі рядки з відношення ARTIST:

```
DELETE
FROM ARTIST
```

Тут слід звернути увагу на процедуру забезпечення цілісності посилань між таблицями ARTIST та WORK. Якщо ми спробуємо виконати наступну команду, то нас буде спіткати невдача, оскільки вказаний рядок таблиці ARTIST має дочірні рядки в таблиці WORK:

```
DELETE
FROM ARTIST
WHERE ArtistID = 14;
```

Є ще одна інструкція видалення всіх рядків таблиці. Це оператор TRUNCATE.

```
TRUNCATE TABLE table_name;
```

Оператор TRUNCATE завжди видаляє всі рядки таблиці. Цей оператор еквівалентний оператору `DELETE FROM table_name`, проте виконується набагато швидше і при видаленні обнуляє лічильник IDENTITY. Розглянемо це на невеликому прикладі.

Нехай визначено таблицю COLORS (ColorID, Color), у якій стовпець ColorID визначено як лічильник IDENTITY. Дані для таблиці COLORS наведено у таблиці 1.3.

Таблиця 1. Дані для таблиці COLORS

ColorID	Color
1	Red
2	Blue
3	White
4	Black
5	Orange

Якщо видалити з таблиці всі рядки інструкцією DELETE, а потім додати новий колір Yellow, тоді таблиці COLORS з'явиться рядок (6, Yellow).

```
DELETE
FROM COLORS
INSERT INTO COLORS(Color) VALUES ('Yellow')
```

Якщо видалити з таблиці всі рядки інструкцією TRUNCATE, та додати новий колір Yellow, тоді таблиці COLORS з'явиться рядок (1, Yellow), тобто лічильник ColorID почне рахувати заново, тоді як у першому випадку продовжить свої значення.

```
TRUNCATE TABLE COLORS
INSERT INTO COLORS(Color) VALUES ('Yellow')
```

Оператор TRUNCATE має обмеження. Не можна використовувати для видалення даних з таблиць, на які є посилання в інших таблицях. Спочатку потрібно видалити всі зв'язки, а потім вже використовувати інструкцію TRUNCATE.

```
TRUNCATE TABLE Comfort
```

```
Msg 4712, Level 16, State 1, Line 1
Cannot truncate table 'Comfort' because it is being referenced
by a FOREIGN KEY constraint.
```

Вибірка даних

Оператор SELECT має такий формальний опис:

```
SELECT [DISTINCT | ALL] {* | [columnExpression [AS newName]]
[ ,  . . .] }
FROM TableName [ , . . ]
[WHERE condition]
[ORDER BY columnList]
```

Тут параметр ***columnExpression*** є ім'ям стовпця або виразом з декількох імен. Якщо необхідно вибрати всі стовпці з кожної таблиці, вказаної в операторі FROM, замість списку імен стовпців можна вказати символ *.

Параметр ***newName*** визначає нове ім'я стовпця при відображенні вибірки, при цьому не змінюючи його в самій таблиці.

Параметр ***TableName*** є ім'ям існуючої в базі даних таблиці (або уявлення), до якої необхідно отримати доступ.

Ключові слова виділені синім шрифтом.

Те, що укладено у квадратні дужки [ ], є необов'язковими параметрами конструкції запиту.

Нижче наведено опис ключових слів.

- SELECT – встановлюється, які стовпці мають бути присутніми у вихідних даних.
- FROM – визначаються імена однієї або декількох таблиць (VIEW), що використовується.
- WHERE – фільтрує рядки таблиці відповідно до заданих умов.
- ORDER BY – визначається впорядкованість результатів виконання оператора.
- DISTINCT – визначає, що в результаті виконання запиту не буде рядків, що дублюються.
- ALL – визначає, що в результаті виконання запиту будуть обрані всі рядки, що навіть дублюються. ALL вибрано за умовчанням, можна опускати.

Розглянемо приклад простих однотабличних запитів на прикладі. Нехай дано відношення *Student (NumGroup, NumStudent, Fio, Year, Ball)*.

Таблиця 1. Дані таблиці STUDENT

NumGroup	NumStudent	Fio	Year	Ball
001	001851	Степаненко М. В.	1985	5
001	001762	Тарасенко А. К.	1976	4,1
101	101791	Адрушенко Р. П.	1979	4,5
101	101782	Тимченко С. А.	1978	4,5
101	101793	Кононенко Е. С.	1979	4,25
101	101774	Євтушенко Н. В.	1977	4,75
102	102771	Семенов Н. П.	1977	4,5
102	102772	Федоренко Д. К.	1977	4,25
102	102773	Єна І. Р.	1977	4,5
103	103781	Балабан Г. М.	1978	4,25
103	103792	Остапенко О. К.	1979	4,75

Приклад 1

```
SELECT NumGroup, NumStudent, Fio, Year, Ball
FROM Student;
```

Оскільки вибірка всіх стовпців, що є в таблиці, виконується досить часто, в мові SQL визначено спрощений варіант запису значення "всі стовпці" – замість імен стовпців вказується символ зірочки (*). Наведений нижче оператор повністю еквівалентний першому і є спрощеним варіантом запису того ж самого запиту:

```
SELECT *
FROM Student;
```

Результат виконання запиту повністю співпадатиме з таблицею Student.

Приклад 2

Нехай необхідно створити звіт про прохідний бал усіх студентів із зазначенням лише номера студента (NumStudent), його імені та прізвища, а також відомостей про бал.

```
SELECT NumStudent, Fio, Ball
FROM Student;
```

У цьому прикладі на основі таблиці Student створюється нова таблиця, що включає лише вказані в запиті стовпці NumStudent, Fio, Ball, причому саме у цьому порядку. Результат виконання цього запиту наведено у табл. 2. Зверніть

увагу, що рядки в результуючій таблиці можуть виявитися несортованими, якщо не дано спеціальну вказівку.

Таблиця 2. Результат виконання запиту на прикладі 2

NumStudent	Fio	Ball
001851	Степаненко М. В.	5
001762	Тарасенко А. К.	4,1
101791	Адрушенко Р. П.	4,5
101782	Тимченко С. А.	4,5
101793	Кононенко Е. С.	4,25
101774	Євтушенко Н. В.	4,75
102771	Семенов Н. П.	4,5
102772	Федоренко Д. К.	4,25
102773	Єна І. Р.	4,5
103781	Балабан Г. М.	4,25
103792	Остапенко О. К.	4,75

Приклад 3

Якщо ми хочемо змінити імена стовпців при відображенні результату запиту, тоді після кожного імені стовпця, яке ми хочемо змінити, вказує його нове ім'я за допомогою ключового слова AS, наприклад, змінимо в запиті прикладу 2 назву стовпця ***Ball*** на «***Прохідний бал***», ***Fio*** на «***Прізвище***», а ім'я стовпця ***NumStudent*** залишимо без зміни. Тоді запит та результуюча таблиця будуть виглядати так.

```
SELECT  NumStudent, Fio AS "Прізвище",  Ball AS "Прохідний бал"
FROM  Student;
```

Таблиця 3. Результат виконання запиту на прикладі 3

NumStudent	Прізвище	Прохідний бал
001851	Степаненко М. В.	5
001762	Тарасенко А. К.	4,1
101791	Адрушенко Р. П.	4,5
101782	Тимченко С. А.	4,5
101793	Кононенко Е. С.	4,25
101774	Євтушенко Н. В.	4,75
102771	Семенов Н. П.	4,5
102772	Федоренко Д. К.	4,25
102773	Єна І. Р.	4,5
103781	Балабан Г. М.	4,25
103792	Остапенко О. К.	4,75

Приклад 4

Тепер розглянемо приклад використання ключового слова DISTINCT. Припустимо, що необхідно створити список усіх номерів груп, де навчаються студенти.

```
SELECT NumGroup
FROM Student;
```

Результат виконання цього запиту наведено в табл. 4. Зверніть увагу, що результат виконання запиту містить значення, що повторюються, оскільки, на відміну від операції проекції реляційної алгебри, оператор SELECT не виключає значень, що повторюються, при виконанні проекції за значеннями одного або декількох стовпців. Для видалення з результуючої таблиці рядків, що повторюються, використовується ключове слово DISTINCT. Відкоригований запит має такий вигляд:

```
SELECT DISTINCT NumGroup
FROM Student;
```

Результати виконання другого варіанта запиту наведено в табл. 5.

Таблиця 4. Результат виконання запиту із збереженням значень, що повторюються, з прикладу 4

NumGroup
001
001
101
101
101
101
102
102
102
103
103

Таблиця 5. Результат виконання запиту з виключенням значень, що повторюються, з прикладу 4

NumGroup
001
101
102
103

Вибірка (конструкція WHERE)

У наведених вище прикладах у результаті виконання операторів SELECT вибиралися усі рядки вказаної таблиці. Однак дуже часто потрібно тим чи іншим чином обмежити набір рядків, які розміщуються в результуючу таблицю запиту. Це досягається за допомогою конструкції WHERE у запиті. Вона складається з ключового слова WHERE, за яким слідує перелік умов пошуку, що визначають ті рядки, які мають бути обрані під час виконання запиту. Існує п’ять основних типів умов пошуку.

- ***Порівняння***. Порівнюються результати обчислення одного виразу з результатами обчислення іншого виразу.

- ***Діапазон***. Перевіряється, чи потрапляє результат обчислення виразу в заданий діапазон значень.
- ***Приналежність до множини***. Перевіряється, чи належить результат обчислення виразу до заданої множини значень.
- ***Відповідність шаблону.*** Перевіряється, чи відповідає рядкове значення заданому шаблону.
- ***Значення NULL.*** Перевіряється, чи містить цей стовпець NULL (невизначене значення).

Конструкція WHERE еквівалентна операції вибірки реляційної алгебри. Розглянемо приклади використання всіх зазначених типів умов пошуку.

Приклад 5. Умови пошуку шляхом порівняння.

Нехай необхідно вивести всіх студентів із прохідним балом більше ніж 4.75.

```
SELECT NumGroup, NumStudent, Fio, Year, Ball
FROM Student
WHERE Ball>4.75;
```

У мові SQL можна використовувати прості операції порівняння, перелічені у таблиці 1.

Таблиця 6. Операції порівняння

Знак операції	Призначення
=	Дорівнює
<>	Не дорівнює (стандарт ISO)
!=	Не дорівнює (використовується в деяких діалектах)
<	Менше
>	Більше
<=	Менше або дорівнює
>=	Більше або дорівнює

Більш складні предикати можуть бути побудовані за допомогою логічних операцій AND, OR або NOT, а також за допомогою дужок, що використовуються визначення порядку обчислення виразу (якщо це необхідно або бажано). Обчислення виразів за умов виконується за такими правилами.

- Вираз обчислюється зліва направо.
- Першими обчислюються вирази у дужках.
- Операції NOT виконуються перед операціями AND та OR.
- Операції AND виконуються перед операціями OR.

Для усунення будь-якої неоднозначності рекомендується використовувати круглі дужки.

Приклад 6. Складні умови пошуку (оператор OR).

Припустимо, що потрібно перерахувати всіх студентів із груп 101 та 103.

```
SELECT NumGroup, NumStudent, Fio
FROM Student
WHERE NumGroup =101
OR NumGroup=103;
```

Приклад 7. Складні умови пошуку (оператор AND).

Припустимо, що потрібно перерахувати всіх студентів із груп 101, середній бал яких більший або дорівнює 4.5.

```
SELECT NumStudent, Fio,  Ball
FROM Student
WHERE NumGroup =101
AND Ball >=4.5;
```

Приклад 8. Використання діапазонів (BETWEEN/NOT BETWEEN) за умов пошуку.

Для вибірки всіх студентів з прохідним балом від 4.5 до 4.75:

```
SELECT NumGroup, NumStudent, Fio,  Ball
FROM Student
WHERE Ball BETWEEN 4.5 AND 4.75;
```

Приклад 9. Умови пошуку з перевіркою приналежності до множини (IN/NOT IN).

Запит, що реалізує складання списку всіх студентів із груп 101, 102 та 103.

```
SELECT NumGroup, NumStudent, Fio,  Ball
FROM Student
WHERE NumGroup IN (101,102,103);
```

Перевірка приналежності до множини забезпечується за допомогою ключового слова IN. При цьому перевіряється, чи відповідає результат обчислення виразу одному із значень у наведеному списку – у нашому випадку це числа 101, 102 та 103.

Приклад 10. Умови пошуку із зазначенням шаблонів (LIKE/NOT LIKE).

Припустимо, необхідно знайти всіх студентів, у прізвищах яких міститься буква 'С'. При виконанні цього запиту необхідно організувати пошук рядка С, який може знаходитися в будь-якому місці значень стовпця Fio таблиці Student.

```
SELECT NumGroup, NumStudent, Fio
FROM Student
WHERE Fio LIKE '%С%';
```

У мові SQL існують два спеціальні символи шаблону, які використовуються під час перевірки символьних значень.

- **%** – символ відсотка представляє будь-яку послідовність з нуля або більше символів (тому часто називається також символом підстановки).
- _ – символ підкреслення представляє окремий символ.

Решта символів у шаблоні представляють себе.

Якщо нам потрібно зробити спеціальні символи (_ та %) як звичайні символи для пошуку, тоді треба їх помістити у квадратні дужки. [_] або [%]

- `Fio LIKE 'C%'`
  Цей шаблон означає, що перший символ обов'язково повинен бути символом C, а всі інші символи не становлять інтересу і не перевіряються.

- `Fio LIKE 'C_'`
  Цей шаблон означає, що значення має мати довжину, що дорівнює строго два символи, причому першим символом обов'язково повинен бути символ 'C'.

- **`Fio LIKE '%a'`**
  Цей шаблон визначає будь-яку послідовність символів довжиною не менше одного символу, причому останнім символом обов’язково має бути символ 'a'.

- `Fio LIKE '%c%'`
  Цей шаблон означає, що нас цікавить будь-яка послідовність символів, що включає підрядок 'c';

- `Fio NOT LIKE 'K%'`
  Цей шаблон вказує на те, що потрібні будь-які рядки, які не починаються з символу К.

Завдання

1. Розробити концептуальну модель бази даних (БД) певної предметної галузі.
2. Відповідно до побудованої моделі розробити схему БД.
3. Відповідно до схеми створити базу даних, використовуючи реляційну СУБД, та заповнити таблиці даними (не менше 5 записів у кожній таблиці).
4. Розробити 10 запитів до індивідуальної бази даних, вказавши їхній фізичний зміст.

Перелік документів для включення до звіту ІР № 1

- титульний лист
- постановка задачі
- концептуальна модель БД у вигляді ER діаграми
- схема БД у вигляді таблиць та зв’язків між ними
- перелік запитів на створення об’єктів БД
- таблиці із заповненими даними
- реалізація 10 запитів, вигаданих самостійно (не більше двох запитів можуть бути однотабличними)

Зразок виконання індивідуального завдання № 1

Постановка задачі

Розробити базу даних для зберігання та обробки інформації про діяльність деякої великої торгової фірми, яка займається постачанням товарів замовникам зі складів. Фірма має кілька філій у різних містах та центральний склад. Постачання товару здійснюються на основі зробленого замовлення, яке має певний номер і дату. Кожне замовлення може стосуватися лише конкретної одиниці товару.

Клієнтам фірми надається деякий кредит і призначається службовець, який забезпечує роботу з даним клієнтом і відповідатиме за зроблені ним замовлення. У базі даних повинна зберігатись інформація про кредит, наданий даному клієнту.

У базі даних має бути подана інформація про стан справ на центральному складі. Кількість одиниць товару, яка залишилася, і ціна одиниці товару. Кожен товар має свою серію та номер, які у парі однозначно характеризують товар. Вважається, що конкретна одиниця товару має фіксовану ціну та компанія працює з конкретним переліком продукції.

Усі співробітники фірми (крім керівника) мають менеджера, який безпосередньо керує їхньою роботою. Кожному зі співробітників призначається плановий обсяг продажу та ведеться облік фактичного обсягу продажів, що складається із суми замовлень, які були виконані через цього службовця.

Концептуальна модель БД (ER діаграма)

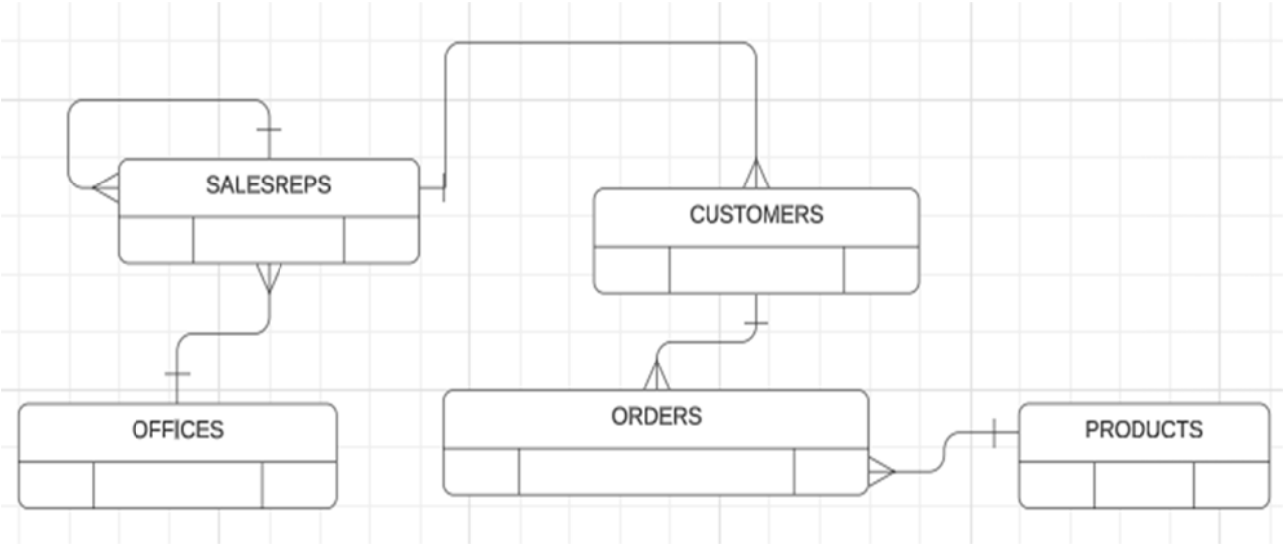
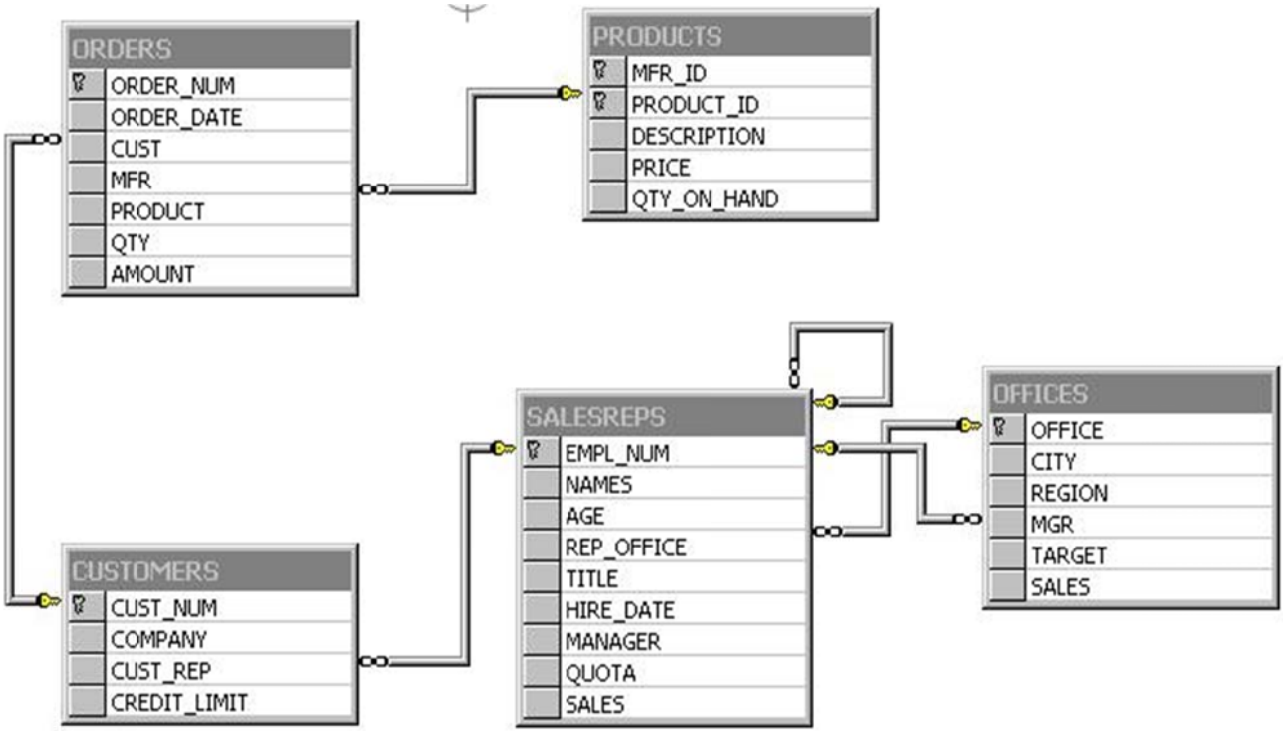


Схема БД у вигляді таблиць та зв’язків між ними



Перелік запитів на створення об’єктів БД

```
CREATE TABLE OFFICES
(
OFFICE SMALLINT NOT NULL PRIMARY KEY,
CITY CHAR (20),
REGION CHAR (20),
MGR SMALLINT,
TARGET MONEY,
SALES MONEY
)

CREATE TABLE SALESREPS
(
EMPL_NUM SMALLINT NOT NULL PRIMARY KEY,
NAMES CHAR (30) NOT NULL,
AGE TINYINT,
REP_OFFICE SMALLINT,
TITLE CHAR(10),
HIRE_DATE SMALLDATETIME,
MANAGER SMALLINT,
QUOTA MONEY,
SALES MONEY
)

CREATE TABLE CUSTOMERS
(
CUST_NUM INT NOT NULL PRIMARY KEY,
COMPANY CHAR(30),
CUST_REP SMALLINT,
CREDIT_LIMIT MONEY
)
```

```
 CREATE TABLE PRODUCTS
(
MFR_ID CHAR(5) NOT NULL,
PRODUCT_ID CHAR(8) NOT NULL,
DESCRIPTION CHAR(25),
PRICE MONEY,
QTY_ON_HAND INT
)

ALTER TABLE PRODUCTS ADD CONSTRAINT PK_PRODUCTS
PRIMARY KEY (MFR_ID,PRODUCT_ID)

CREATE TABLE ORDERS
(
ORDER_NUM INT NOT NULL PRIMARY KEY,
ORDER_DATE SMALLDATETIME,
CUST INT,
MFR CHAR(5),
PRODUCT CHAR(8),
QTY INT,
AMOUNT MONEY
)

ALTER TABLE OFFICES ADD CONSTRAINT FK_HASMGR
FOREIGN KEY (MGR) REFERENCES SALESREPS(EMPL_NUM)

ALTER TABLE SALESREPS ADD CONSTRAINT FK_WORKSIN
FOREIGN KEY (REP_OFFICE) REFERENCES OFFICES(OFFICE)

ALTER TABLE SALESREPS ADD CONSTRAINT FK_RUK
FOREIGN KEY (MANAGER) REFERENCES SALESREPS(EMPL_NUM)

ALTER TABLE CUSTOMERS ADD CONSTRAINT FK_HASREP
FOREIGN KEY (CUST_REP) REFERENCES SALESREPS(EMPL_NUM)

ALTER TABLE ORDERS ADD CONSTRAINT FK_PLACEDBY
FOREIGN KEY (CUST) REFERENCES CUSTOMERS(CUST_NUM)

ALTER TABLE ORDERS ADD CONSTRAINT FK_ISFOR
FOREIGN KEY (MFR,PRODUCT) REFERENCES PRODUCTS(MFR_ID, PRODUCT_ID)
```

Таблиці із заповненими даними

CUST_NUM	COMPANY	CUST_REP	CREDIT_LIMIT
2111	JCP Inc.	103	$50,000.00
2102	First Corp.	101	$65,000.00
2103	Acme Mfg.	105	$50,000.00
2123	Carter & Sons	102	$40,000.00
2107	Ace International	110	$35,000.00
2115	Smithson Corp.	101	$20,000.00
2101	Jones Mfg.	106	$65,000.00
2112	Zetacorp	108	$50,000.00
2121	QMA Assoc.	103	$45,000.00
2114	Orion Corp.	102	$20,000.00
2124	Peter Brothers	107	$40,000.00
2108	Holm & Landis	109	$55,000.00
2117	J.P. Sinclair	106	$35,000.00
2122	Three-Way Lines	105	$30,000.00
2120	Rico Enterprises	102	$50,000.00
2106	Fred Lewis Corp.	102	$65,000.00
2119	Solomon Inc.	109	$25,000.00
2118	Midwest Systems	108	$60,000.00
2113	Ian & Schmidt	104	$20,000.00
2109	Chen Associates	103	$25,000.00
2105	AAA Investments	101	$45,000.00

Таблиця CUSTOMERS

EMPL_NUM	NAME	AGE	REP_OFFICE	TITLE	HIRE_DATE	MANAGER	QUOTA	SALES
105	Bill Adams	37	13	Sales Rep	12-FEB-88	104	$350,000.00	$367,911.00
109	Mary Jones	31	11	Sales Rep	12-OCT-89	106	$300,000.00	$392,725.00
102	Sue Smith	48	21	Sales Rep	10-DEC-86	108	$350,000.00	$474,050.00
106	Sam Clark	52	11	VP Sales	14-JUN-88	NULL	$275,000.00	$299,912.00
104	Bob Smith	33	12	Sales Mgr	19-MAY-87	106	$200,000.00	$142,594.00
101	Dan Roberts	45	12	Sales Rep	20-OCT-86	104	$300,000.00	$305,673.00
110	Tom Snyder	41	NULL	Sales Rep	13-JAN-90	101	NULL	$75,985.00
108	Larry Fitch	62	21	Sales Mgr	12-OCT-89	106	$350,000.00	$361,865.00
103	Paul Cruz	29	12	Sales Rep	01-MAR-87	104	$275,000.00	$286,775.00
107	Nancy Angelli	49	22	Sales Rep	14-NOV-88	108	$300,000.00	$186,042.00

Таблиця SALESREPS

OFFICE	CITY	REGION	MGR	TARGET	SALES
22	Denver	Western	108	$300,000.00	$186,042.00
11	New York	Eastern	106	$575,000.00	$692,637.00
12	Chicago	Eastern	104	$800,000.00	$735,042.00
13	Atlanta	Eastern	105	$350,000.00	$367,911.00
21	Los Angeles	Western	108	$725,000.00	$835,915.00

Таблиця OFFICES

ORDER_NUM	ORDER_DATE	CUST	REP	MFR	PRODUCT	QTY	AMOUNT
112961	17-DEC-89	2117	106	REI	2A44L	7	$31,500.00
113012	11-JAN-90	2111	103	ACI	41003	35	$3,745.00
112989	03-JAN-90	2101	106	FEA	114	6	$1,458.00
113051	10-FEB-90	2118	108	QSA	XK47	4	$1,420.00
112968	12-OCT-89	2102	101	ACI	41004	34	$3,978.00
113036	30-JAN-90	2107	110	ACI	4100Z	9	$22,500.00
113045	02-FEB-90	2112	108	REI	2A44R	10	$45,000.00
112963	17-DEC-89	2103	105	ACI	41004	28	$3,276.00
113013	14-JAN-90	2118	108	BIC	41003	1	$652.00
113058	23-FEB-90	2108	109	FEA	112	10	$1,480.00
112997	08-JAN-90	2124	107	BIC	41003	1	$652.00
112983	27-DEC-89	2103	105	ACI	41004	6	$702.00
113024	20-JAN-90	2114	102	QSA	XK47	20	$7,100.00
113062	24-FEB-90	2124	107	FEA	114	10	$2,430.00
112979	12-OCT-89	2114	102	ACI	4100Z	6	$15,000.00
113027	22-JAN-90	2103	105	ACI	41002	54	$4,104.00
113007	08-JAN-90	2112	108	IMM	773C	3	$2,925.00
113069	02-MAR-90	2109	103	IMM	775C	22	$31,350.00
113034	29-JAN-90	2107	110	REI	2A45C	8	$632.00
112992	04-NOV-89	2118	108	ACI	41002	10	$760.00
112975	12-OCT-89	2111	103	REI	2A44G	6	$2,100.00
113055	15-FEB-90	2108	109	ACI	4100X	6	$150.00
113048	10-FEB-90	2120	102	IMM	779C	2	$3,750.00
112993	04-JAN-89	2106	102	REI	2A45C	24	$1,896.00
113065	27-FEB-90	2106	102	QSA	XK47	6	$2,130.00
113003	25-JAN-90	2108	109	IMM	779C	3	$5,625.00
113049	10-FEB-90	2118	108	QSA	XK47	2	$710.00
112987	31-DEC-89	2103	105	ACI	4100Y	10	$27,500.00
113057	18-FEB-90	2111	103	ACI	4100X	24	$600.00
113042	02-FEB-90	2113	104	REI	2A44R	5	$22,500.00

Таблиця ORDERS

MFR_ID	PRODUCT_ID	DESCRIPTION	PRICE	QTY_ON_HAND
REI	2A45C	Ratchet Link	$79.00	210
ACI	4100Y	Widget Remover	$2,750.00	25
QSA	XK47	Reducer	$355.00	38
BIC	41672	Plate	$180.00	0
IMM	779C	900-lb Brace	$1,875.00	9
ACI	41003	Size 3 Widget	$107.00	207
ACI	41004	Size 4 Widget	$117.00	139
BIC	41003	Handle	$652.00	3
IMM	887P	Brace Pin	$250.00	24
QSA	XK48	Reducer	$134.00	203
REI	2A44L	Left Hinge	$4,500.00	12
FEA	112	Housing	$148.00	115
IMM	887H	Brace Holder	$54.00	223
BIC	41089	Retainer	$225.00	78
ACI	41001	Size 1 Widget	$55.00	277
IMM	775C	500-lb Brace	$1,425.00	5
ACI	4100Z	Widget Installer	$2,500.00	28
QSA	XK48A	Reducer	$177.00	37
ACI	41002	Size 2 Widget	$76.00	167
REI	2A44R	Right Hinge	$4,500.00	12
IMM	773C	300-lb Brace	$975.00	28
ACI	4100X	Widget Adjuster	$25.00	37
FEA	114	Motor Mount	$243.00	15
IMM	887X	Brace Retainer	$475.00	32
REI	2A44G	Hinge Pin	$350.00	14

Таблиця PRODUCTS

Запити до БД

1. Вивести список всіх службовців компанії із зазначенням імені, віку та займаної посади. Стовпці, що виводяться, повинні мати імена "ПІБ", "ВІК", "ПОСАДА" відповідно. Зберегти запит як view.

```
-- реалізація запиту
SELECT NAME AS "ПІБ", AGE AS "ВІК", TITLE AS "ПОСАДА"
FROM SALESREPS

-- створення VIEW на основі запиту
CREATE VIEW TASK1 AS
SELECT NAME AS "ПІБ", AGE AS "ВІК", TITLE AS "ПОСАДА"
FROM SALESREPS
```

2. Вивести список регіонів, у яких існують філії компанії, відсортувавши їх за абеткою. Зберегти запит як VIEW.

```
-- створення VIEW на основі запиту
CREATE VIEW TASK2 AS
SELECT DISTINCT REGION AS "РЕГІОНИ"
FROM OFFICES

-- виведення відповідного VIEW
SELECT * FROM TASK2
ORDER BY REGION
```

3. Вивести список одиниць продукції, яку реалізує компанія із зазначенням серії, номера та назви товару, а також вказавши ціну одиниці товару, наявну кількість товару та загальну вартість. Стовпцям, що виводяться, дати відповідні імена.

```
SELECT MFR_ID AS "СЕРІЯ", PRODUCT_ID AS "НОМЕР", DESCRIPTION
AS "Найменування",
PRICE AS "ЦІНА", QTY_ON_HAND AS "КІЛЬКІСТЬ", QTY_ON_HAND *
PRICE AS "ВАРТІСТЬ"
FROM PRODUCTS
```

4. Вивести список тих замовників, у яких кредит перевищує $50.000 (із зазначенням розміру кредиту). Список відсортувати відповідно до зменшення розміру кредиту.

```
SELECT COMPANY, CREDIT_LIMIT
FROM CUSTOMERS
WHERE CREDIT_LIMIT > 50.000
ORGER BY CREDIT_LIMIT DESC
```

5. Вивести список тих службовців, чий вік знаходиться в діапазоні від 30 до 50 (включно), відсортувавши рядки відповідно до дати прийому на роботу.

```
--перший варіант запиту
SELECT NAME, AGE, HIRE_DATE FROM SALESREPS
WHERE AGE BETWEEN 30 AND 50
ORGER BY HIRE_DATE
```

```
--другий варіант запиту
SELECT NAME, AGE, HIRE_DATE
FROM SALESREPS
WHERE AGE >= 30 AND AGE <= 50
ORGER BY HIRE_DATE
```

6. Вивести список одиниць товару, що належать до серій ACI, REI та IMM, із зазначенням ціни за одиницю товару.

```
--перший варіант запиту
SELECT MFR_ID, DESCRIPTION, PRICE
FROM PRODUCTS
WHERE MFR_ID IN ('ACI','REI','IMM')
--другий варіант запиту
SELECT MFR_ID, DESCRIPTION, PRICE
FROM PRODUCTS
WHERE (MFR_ID = 'ACI') OR (MFR_ID = 'REI') OR (MFR_ID = 'IMM')
```

7. Вивести список тих співробітників, які не є менеджерами та чиє ім'я починається з літери S.

```
SELECT NAME, TITLE
FROM SALESREPS
WHERE (NAME LIKE 'S%') AND (TITLE NOT IN ('Sales Mgr'))
```

8. Вивести список тих співробітників, які не є менеджерами та чиє ім'я починається з літери S.

```
SELECT NAME, TITLE
FROM SALESREPS
WHERE (NAME LIKE 'S%') AND (TITLE NOT IN ('Sales Mgr'))
```

9. Вивести список співробітників, вік яких не перевищує 40 років, вказавши ім'я міста в якому розташована філія, де працює відповідний співробітник.

```
--перший варіант запиту
SELECT SALESREPS.NAME, SALESREPS.AGE, OFFICE.CITY
FROM SALESREPS, OFFICES
WHERE SALESREPS.REP_OFFICE = OFFICES.OFFICE
AND SALESREPS.AGE <= 40
--другий варіант запиту (можливий так, як імена стовпців не повторюються)
SELECT NAME, AGE, CITY
FROM SALESREPS, OFFICES
WHERE REP_OFFICE = OFFICE
AND AGE <= 40
```

10. Вивести список співробітників, вказавши імена їхніх менеджерів.

```
SELECT S1.NAME AS "СПІВРОБІТНИК", S2.NAME AS "КЕРІВНИК"
FROM SALESREPS AS S1, SALESREPS AS S2
WHERE S1.MANAGER=S2.EMPL_NUM
```

11. Вивести список замовлень, виконаних у літній період 1990 року, включаючи ім'я службовця, який прийняв замовлення, та ім'я клієнта, який його зробив. Стовпцям, що виводяться, дати відповідні імена.

```
SELECT ORDERS.ORDER_NUM AS "НОМЕР", ORDERS.ORDER_DATE AS "ДАТА",
SALESREPS.NAME AS "СПІВРОБІТНИК", CUSTOMERS.COMPANY AS "КЛІЄНТ"
FROM ORDERS, SALESREPS, CUSTOMERS
WHERE ORDERS.CUST = CUSTOMERS.CUST_NUM
AND ORDERS.REP = SALESREPS.EMPL_NUM
AND ORDERS.ORDER_DATE BETWEEN '1990-06-01' AND '1990-08-31'
```

Питання для самоконтролю

1. Якого типу об'єкти можна задати в конструкції FROM оператора SELECT?
2. Які методи дозволяють модифікувати дані в БД?
3. Якими властивостями повинен володіти ключ.
4. Як реалізується зв'язок «один до багатьох» з обов'язковим класом приналежності для однієї сутності і з необов'язковим для іншої.
5. Як реалізується зв'язок «один до багатьох» з обов'язковим класом приналежності для обох сутностей.
6. Як реалізується зв'язок «багатьох до багатьох».
7. Що таке обмеження FOREIGN KEY і як його створити?

Індивідуальна робота № 2
ПІДСУМКОВІ ЗАПИТАННЯ НА ВИБІРКУ. ПІДЛЕГЛІ ЗАПИТИ

Теоретичний матеріал

Розгляньмо наступну таблицю

*Student (**Id_Student, fio, dateOfBirth, Course, email,ball***

Id_Student	fio	dateOfBirth	Course	email	ball
74898	Постова Ольга	13.03.2003	1	postova003@gmail.com	4
74899	Ряба Анастасія	13.06.2002	1	anastasiiariaba13@gmail.com	3.8
74900	Слободчикова Тетяна	21.01.2003	1	marsreklama2017@gmail.com	4.2
74901	Кулик Кирило	31.05.2003	1	dr.frost2001@gmail.com	3.3
74902	Капуста Юлія	31.01.2003	1	yulia.kapusta@gmail.com	4.9
74903	Мальована Анастасія	03.02.2003	1	malya82@mail.ru	5
74904	Коваленко Вікторія	01.06.2000	2	NULL	3.5
74905	Ангеловська Карина	06.05.2002	2	angel.karina605@gmail.com	4.5
74906	Сітченко Ксенія	14.01.2003	1	sitksyalex@gmail.com	4.5
74907	Маланія Надія	03.12.2002	1	nmalaniya02@gmail.com	3
74908	Чередниченко Артем	27.07.2003	1	thechirikchannel@gmail.com	4
74909	Кубишкін Антон	10.03.2003	1	kryachko03@ukr.net	4.6
74910	Маловічко Анна	16.10.2000	3	NULL	4.8
74911	Мирошниченко Богдан	04.05.2001	2	iw.ronniel@gmail.com	3
74912	Зубенко Поліна	10.07.2003	1	zksena2017@gmail.com	3.1
74913	Комбарова Анастасія	25.07.2003	1	nastyakombarova563@gmail.com	3.5
74914	Балюк Ждан	05.01.2003	1	bakyuk61903@gmail.com	4.2
74915	Мхікян Лєна	13.05.2002	2	NULL	4
74916	Тригуб Кирило	31.01.2003	1	vlad.trigub.1978@gmail.com	4.1
74917	Бакуменко Анна	22.09.2002	1	anna_bakumenko@ukr.net	3.3
74918	Соловйова Поліна	23.05.2003	1	copsergsv@gmail.com	4.2
74919	Древаль Ігор	19.03.2003	1	dreval@agrosfera.ua	4.7

Для реалізації підсумкових запитів на вибірку існують вбудовані функції агрегації. Список функцій агрегації наведено далі.

COUNT(*) – підрахунок кількості усіх рядків таблиці.

```
SELECT COUNT (*)
FROM Student
```

Або рядків, які відповідають умові (дата народження =…)

```
SELECT COUNT(*) AS “cnt”
FROM Student
WHERE dateOfBirth ='31.01.2003'
```

COUNT(column_name) – кількість рядків, в яких стовпчик ***column_name*** визначений (тобто ***column_name*** IS NOT NULL).

Вивести кількість студентів, у яких вказано email

```
SELECT COUNT(email)
FROM Student
```

COUNT(DISTINCT column_name) – кількість різних рядків, в яких стовпчик ***column_name*** визначений.

Вивести скільки різних курсів в таблиці Студент

```
SELECT COUNT (DISTINCT course)
FROM Student
```

AVG(column_name) – середнє арифметичне значення за всіма визначеними значеннями поля column_name (тобто значення NULL ігноруються).

Вивести середній бал серед всіх студентів

```
SELECT AVG(ball) AS "avg ball"
FROM student
```

MIN(column_name) – мінімальне значення стовпця.

Вивести мінімальний бал серед всіх студентів

```
SELECT MIN(ball) AS "min ball"
FROM student
```

MAX(column_name) – максимальне значення стовпця.

Вивести максимальний бал серед всіх студентів

```
SELECT max(ball) AS "max ball"
FROM student
```

SUM(column_name) – сума всіх визначених (NOT NULL) значень поля.

SUM(DISTINCT column_name) – сума всіх значень, які не повторюються.

```
SELECT SUM(number)
FROM [order]
```

або

```
SELECT SUM(number)
FROM [order], product
WHERE title='milk' AND order.code=product.code
```

Можна вибрати відразу кілька значень, що агрегують.

```
SELECT MAX(ball), MIN(ball)
FROM Student
```

MAX, MIN, AVG, SUM ігнорують значення NULL

НЕ МОЖНА *в запитах, що агрегують, виводити стовпці таблиці без інструкції* ***GROUP BY.***

Цей запит містить помилку і не буде виконаний.

```
SELECT fio, AVG(ball)
FROM student
```

Групування результатів

GROUP BY використовується спільно з функціями агрегації і розбиває таблицю на групи, а функції агрегації обчислюють для кожної з них підсумкове значення.

HAVING для накладання умов на рядки групи (ті ж функції що й WHERE у звичайних запитах).

Приклад 1

Для кожного курсу вивести кількість студентів, які навчаються на цьому курсі.

```
SELECT Course, COUNT(*) AS NumberOfStudents
FROM Student
GROUP BY Course
```

Нижче наведено алгоритм виконання запиту.

1. Розбиваємо на групи по курсу, тобто однакові значення курсу належать одній групі.

У нашому прикладі буде 3 групи:

1 курс (рядки виділені червоним кольором).

2 курс (рядки виділені синім кольором).

3 курс (рядки виділені зеленим кольором).

Id_Student	fio	dateOfBirth	Course	email	ball
74898	Постова Ольга	13.03.2003	1	postova003@gmail.com	4
74899	Ряба Анастасія	13.06.2002	1	anastasiiariaba13@gmail.com	3.8
74900	Слободчикова Тетяна	21.01.2003	1	marsreklama2017@gmail.com	4.2
74901	Кулик Кирило	31.05.2003	1	dr.frost2001@gmail.com	3.3
74902	Капуста Юлія	31.01.2003	1	yulia.kapusta@gmail.com	4.9
74903	Мальована Анастасія	03.02.2003	1	malya82@mail.ru	5
74904	Коваленко Вікторія	01.06.2000	2	NULL	3.5
74905	Ангеловська Карина	06.05.2002	2	angel.karina605@gmail.com	4.5
74906	Сітченко Ксенія	14.01.2003	1	sitksyalex@gmail.com	4.5
74907	Маланія Надія	03.12.2002	1	nmalaniya02@gmail.com	3
74908	Чередниченко Артем	27.07.2003	1	thechirikchannel@gmail.com	4
74909	Кубишкін Антон	10.03.2003	1	kryachko03@ukr.net	4.6
74910	Маловічко Анна	16.10.2000	3	NULL	4.8
74911	Мирошниченко Богдан	04.05.2001	2	iw.ronniel@gmail.com	3
74912	Зубенко Поліна	10.07.2003	1	zksena2017@gmail.com	3.1
74913	Комбарова Анастасія	25.07.2003	1	nastyakombarova563@gmail.com	3.5
74914	Балюк Ждан	05.01.2003	1	bakyuk61903@gmail.com	4.2
74915	Мхікян Лєна	13.05.2002	2	NULL	4
74916	Тригуб Кирило	31.01.2003	1	vlad.trigub.1978@gmail.com	4.1
74917	Бакуменко Анна	22.09.2002	1	anna_bakumenko@ukr.net	3.3
74918	Соловйова Поліна	23.05.2003	1	copsergsv@gmail.com	4.2
74919	Древаль Ігор	19.03.2003	1	dreval@agrosfera.ua	4.7

2. Для кожної групи рахуємо кількість рядків, які потрапили до групи.

Course	NumberOfStudents
1	17
2	4
3	1

Приклад 2

Для кожного курсу вивести кількість студентів, які навчаються на цьому курсі, АЛЕ лише для тих курсів, на яких навчаються більше ніж 1 студент.

```
SELECT Course, COUNT(*) AS NumberOfStudents
FROM Student
GROUP BY Course
HAVING COUNT(*)>1
```

Нижче наведено алгоритм виконання запиту.

1. Розбиваємо на групи по курсу, тобто однакові значення курсу належать одній групі (як і в попередньому прикладі).
2. Для кожної групи рахуємо кількість рядків, які потрапили до групи (як і в попередньому прикладі).
3. Фільтруємо результат і залишаємо лише ті рядки, де кількість більша за 1.

Результат

Course	NumberOfStudents
1	17
2	4

Приклад 3

Для кожного курсу вивести кількість студентів, які навчаються на цьому курсі, чий бал більший або дорівнює 4.

```
SELECT Course, COUNT(*) AS NumberOfStudents
FROM Student
WHERE ball>=4
GROUP BY Course
```

Нижче наведено алгоритм виконання запиту.

1. Фільтруємо таблицю Student і залишаємо лише ті рядки, де середній бал більший або дорівнює 4.

Id_Student	fio	dateOfBirth	Course	email	ball
74898	Постова Ольга	13.03.2003	1	postova003@gmail.com	4
~~74899~~	~~Ряба Анастасія~~	~~13.06.2002~~	~~1~~	~~anastasiiariaba13@gmail.com~~	~~3.8~~
74900	Слободчикова Тетяна	21.01.2003	1	marsreklama2017@gmail.com	4.2
~~74901~~	~~Кулик Кирило~~	~~31.05.2003~~	~~1~~	~~dr.frost2001@gmail.com~~	~~3.3~~
74902	Капуста Юлія	31.01.2003	1	yulia.kapusta@gmail.com	4.9
74903	Мальована Анастасія	03.02.2003	1	malya82@mail.ru	5
~~74904~~	~~Коваленко Вікторія~~	~~01.06.2000~~	~~2~~	~~NULL~~	~~3.5~~
74905	Ангеловська Карина	06.05.2002	2	angel.karina605@gmail.com	4.5
74906	Сітченко Ксенія	14.01.2003	1	sitksyalex@gmail.com	4.5
~~74907~~	~~Маланія Надія~~	~~03.12.2002~~	~~1~~	~~nmalaniya02@gmail.com~~	~~3~~
74908	Чередниченко Артем	27.07.2003	1	thechirikchannel@gmail.com	4
74909	Кубишкін Антон	10.03.2003	1	kryachko03@ukr.net	4.6
74910	Маловічко Анна	16.10.2000	3	NULL	4.8
~~74911~~	~~Мирошниченко Богдан~~	~~04.05.2001~~	~~2~~	~~iw.ronniel@gmail.com~~	~~3~~

Id_Student	fio	dateOfBirth	Course	email	ball
~~74912~~	~~Зубенко Поліна~~	~~10.07.2003~~	~~1~~	~~zksena2017@gmail.com~~	~~3.1~~
~~74913~~	~~Комбарова Анастасія~~	~~25.07.2003~~	~~1~~	~~nastyakombarova563@gmail.com~~	~~3.5~~
74914	Балюк Ждан	05.01.2003	1	bakyuk61903@gmail.com	4.2
74915	Мхікян Лєна	13.05.2002	2	NULL	4
74916	Тригуб Кирило	31.01.2003	1	vlad.trigub.1978@gmail.com	4.1
~~74917~~	~~Бакуменко Анна~~	~~22.09.2002~~	~~1~~	~~anna_bakumenko@ukr.net~~	~~3.3~~
74918	Соловйова Поліна	23.05.2003	1	copsergsv@gmail.com	4.2
74919	Древаль Ігор	19.03.2003	1	dreval@agrosfera.ua	4.7

2. Розбиваємо на групи по курсу, тобто однакові значення курсу належать одній групі

Id_Student	fio	dateOfBirth	Course	email	ball
74898	Постова Ольга	13.03.2003	1	postova003@gmail.com	4
74900	Слободчикова Тетяна	21.01.2003	1	marsreklama2017@gmail.com	4.2
74902	Капуста Юлія	31.01.2003	1	yulia.kapusta@gmail.com	4.9
74903	Мальована Анастасія	03.02.2003	1	malya82@mail.ru	5
74905	Ангеловська Карина	06.05.2002	2	angel.karina605@gmail.com	4.5
74906	Сітченко Ксенія	14.01.2003	1	sitksyalex@gmail.com	4.5
74908	Чередниченко Артем	27.07.2003	1	thechirikchannel@gmail.com	4
74909	Кубишкін Антон	10.03.2003	1	kryachko03@ukr.net	4.6
74910	Маловічко Анна	16.10.2000	3	NULL	4.8
74914	Балюк Ждан	05.01.2003	1	bakyuk61903@gmail.com	4.2
74915	Мхікян Лєна	13.05.2002	2	NULL	4
74916	Тригуб Кирило	31.01.2003	1	vlad.trigub.1978@gmail.com	4.1
74918	Соловйова Поліна	23.05.2003	1	copsergsv@gmail.com	4.2
74919	Древаль Ігор	19.03.2003	1	dreval@agrosfera.ua	4.7

3. Для кожної групи рахуємо кількість рядків, що потрапили до групи.

Результат

Course	NumberOfStudents
1	11
2	2
3	1

Підлеглі запити

Підлеглі запити, внутрішні або вкладені – це запит всередині запиту. Підлеглий запит поміщається у круглі дужки.

Види підлеглих запитів:

- Скалярний – повертає єдине значення одного стовпця.
- Строковий – може бути кілька стовпців, але у вигляді єдиного рядка.
- Табличний простий – один стовпець багато рядків.
- Табличний складений – декілька стовпців та рядків (тільки у конструкціях WHERE, HAVING, FROM).

Скалярний

Скалярний можна використовувати для операторів порівняння =, <. >.

Обмеження

Підзапит повинен повертати лише 1 стовпець 1 рядок (або 1 агрегацію).

Вивести всіх студентів групи 'МФ-21'

Реалізація у вигляді об'єднання таблиць:

```
SELECT fio
FROM Student inner join [Group]
            ON ref_group = Id_group
 WHERE title = 'МФ-21'
```

Реалізація у вигляді підлеглого запиту:

```
SELECT fio
FROM Student
WHERE ref_group = (SELECT Id_group
             FROM [Group] WHERE title = 'МФ-21')
```

Табличний простий

Можна використовувати в конструкціях IN (NOT IN).

Вивести всіх студентів (ПІБ), які навчаються в групах з префіксом «МФ».

```
SELECT fio
FROM Student
WHERE ref_group IN (SELECT Id_group
             FROM [Group] WHERE title LIKE 'МФ%')
```

Посилання на зовнішній запит із підзапиту

Вивести список студентів, середній бал яких більший за середній бал у їх групі

```
SELECT fio, ball
FROM Student s
WHERE ball > (SELECT AVG(s1.ball) FROM Student s1
         WHERE s1.ref_group = s.ref_group)
```

EXISTS / NOT EXISTS

EXISTS (якщо існує хоча б 1 рядок у результуючій вибірці).

NOT EXISTS (якщо не існує жодного рядка у результуючій вибірці) як умова в WHERE або HAVING.

Приклад

Вивести назву всіх груп, у яких немає студентів

```
SELECT title FROM [group]
WHERE NOT EXISTS (SELECT * FROM Student
                  WHERE ref_group=id_group)
```

```
SELECT title FROM [group]
WHERE (SELECT count(*) FROM Student WHERE
ref_group=id_group)=0
```

Завдання

1. Розробити 10 підсумкових запитів до індивідуальної бази даних, вказавши їхній фізичний зміст.
2. Розробити 10 підлеглих запитів до індивідуальної бази даних, вказавши їхнє фізичне значення.

Перелік документів для включення до звіту ІР № 2

- титульний лист;
- постановка задачі;
- схема БД у вигляді таблиць та зв'язків між ними;
- реалізація 10 підсумкових запитів, придуманих самостійно (**мінімум два запити повинні містити інструкцію HAVING.**);
- реалізація 5 підлеглих запитів, вигаданих самостійно.

Зразки виконання запитів ІР № 2 (на прикладі навчальної бази даних)

1. Вивести середнє значення планових та фактичних обсягів продажів по всіх офісах, розташованих у східному регіоні.

```
-- реалізація запиту
SELECT AVG(TARGET), AVG(SALES) FROM OFFICES
WHERE REGION = 'Eastern'
```

2. Вивести кількість співробітників, які працюють у кожному з регіонів.

```
-- реалізація запиту
SELECT COUNT(*), REGION FROM OFFICES, SALSESREPS
WHERE REP_OFFICE=OFFICE
GROUP BY REGION
```

3. Для кожного із співробітників вказати кількість клієнтів, з якими він працює та загальну суму кредиту, надану їм.

```
SELECT NAMES, COUNT(*) FROM CUSTOMERS, SALESREPS
WHERE CUST_REP=EMPL_NUM
GROUP BY NAMES
```

4. Для кожного співробітника вивести сумарну вартість замовлень, які було виконано через нього.

```
SELECT NAMES, COUNT(*)
FROM ORDERS, CUSTOMERS, SALESREPS
WHERE ORDERS.CUST = CUSTOMERS.CUST_NUM AND
CUSTOMERS.CUST_REP=SALESREPS.EMPL_NUM
GROUP BY NAMES
```

5. Вивести сумарну вартість товарів кожної серії.

```
SELECT MFR_ID, SUM(PRICE*QTY_ON_HAND)
FROM PRODUCTS
GROUP BY (MFR_ID)
```

6. Для кожного зі співробітників вказати загальну кількість підлеглих співробітників.

```
SELECT C2.NAMES, COUNT(*)
FROM SALESREPS AS C1, SALESREPS AS C2
WHERE C1.MANAGER=C2.EMPL_NUM
GROUP BY(C2.NAMES)
```

7. Обчислити середню вартість замовлення кожного співробітника з тих, які мають загальну вартість замовлень, яка перевищує $30000.

```
SELECT REP, AVG(AMOUNT)
FROM ORDERS
GROUP BY REP
HAVING SUM(AMOUNT) > 30000.00
```

8. Для кожного офісу, в якому працюють двоє або більше осіб, обчислити загальний плановий та фактичний обсяг продажів для всіх співробітників офісу.

```
SELECT CITY, SUM(QUOTA), SUM(SALESREPS.SALES)
FROM OFFICES, SALESREPS
WHERE OFFICE=REP_OFFICE
GROUP BY CITY
HAVING COUNT(*) >=2
```

9. Знайти середню вартість замовлень, загальну вартість замовлень, середню вартість замовлень у відсотках від лімітів кредиту клієнтів, і навіть середню вартість замовлень у відсотках від планових обсягів продажу службовців.

```
SELECT AVG(AMOUNT), SUM(AMOUNT),
(100*AVG(AMOUNT/CREDIT_LIMIT)),
(100*AVG(AMOUNT/QUOTA))
FROM ORDERS, CUSTOMERS, SALESREPS
WHERE CUST = CUST_NUM
AND REP=EMPL_NUM
```

10. Вивести імена та дані про вік співробітників в офісах західного регіону.

```
--Виконання запиту за допомогою об'єднання двох таблиць
SELECT NAME, AGE
FROM SALESREPS, OFFICES
WHERE REP_OFFICE = OFFICE
AND REGION = 'WESTERN'

--Виконання запиту за допомогою підлеглого запиту
SELECT NAME, AGE
FROM SALESREPS
WHERE REP_OFFICE IN (SELECT OFFICE
FROM OFFICES
WHERE REGION = 'WESTERN')
```

11. Вивести імена та дані про вік співробітників, для яких плановий обсяг продажів вищий за середній.

```
-- цей запит не можна виконати за допомогою об'єднання таблиць
SELECT NAME, AGE
FROM SALESREPS
WHERE QUOTA > (SELECT AVG (QUOTA) FROM SALESREPS)
```

12. Вивести список клієнтів, яких обслуговує Біл Адамс (BILL ADAMS).

```
SELECT COMPANY
FROM CUSTOMERS
WHERE CUST_REP = (SELECT EMPL_NUM
FROM SALESREPS
WHERE NAME = 'BILL ADAMS')
```

13. Вивести список співробітників тих офісів, де фактичний обсяг продажу перевищує плановий.

```
SELECT NAME
FROM SALESREPS
WHERE REP_OFFICE IN (SELECT OFFICE
FROM OFFICES
WHERE SALES > [TARGET])
```

14. Вивести список клієнтів, закріплених за співробітниками, які працюють у офісах східного регіону.

```
SELECT COMPANY
FROM CUSTOMERS
WHERE CUST_REP IN (SELECT EMPL_NUM
FROM SALESREPS
WHERE REP_OFFICE IN (SELECT OFFICE
FROM OFFICES
WHERE REGION = 'EASTERN'))
```

15. Вивести список співробітників, чий плановий обсяг продажу складає менше 10 % від планового обсягу продажів усієї компанії.

```
SELECT NAMES
FROM SALESREPS
WHERE QUOTA < (.1 * (SELECT SUM([TARGET])
FROM OFFICES))
```

Питання для самоконтролю

1. Які функції агрегації ігнорують NULL значення.
2. Коли функція агрегації може повернути NULL значення.
3. Що таке підлеглий запит?
4. Перерахуйте види підлеглих запитів відповідно до типу результату підлеглого запиту.
5. В яких частинах запиту можуть розміщуватися підлеглі запити.

Індивідуальна робота № 3
ЗОВНІШНЄ ОБ'ЄДНАННЯ ТАБЛИЦЬ. ЛІВЕ І ПРАВЕ ЗОВНІШНЄ ОБ'ЄДНАННЯ ТАБЛИЦЬ. ЗБЕРЕЖЕНІ ПРОЦЕДУРИ

Теоретичний матеріал

З'єднання є підмножиною більш загальної комбінації даних двох таблиць, яка називається **декартовим добутком**.

Декартовий добуток двох таблиць являє собою іншу таблицю, що складається з усіх можливих пар рядків, що входять до складу обох таблиць. Набір стовпців результуючої таблиці є всі стовпці першої таблиці, за якими слідують усі стовпці другої таблиці.

Якщо ввести запит до двох таблиць без завдання конструкції WHERE, результат виконання запиту в середовищі SQL буде декартовим добутком цих таблиць. Крім того, стандарт ISO передбачає спеціальний формат оператора SELECT, що дозволяє обчислити декартовий добуток двох таблиць:

```
SELECT  [DISTINCT | ALL] {* | columnList}
FROM  tableName1 CROSS JOIN tableName 2
```

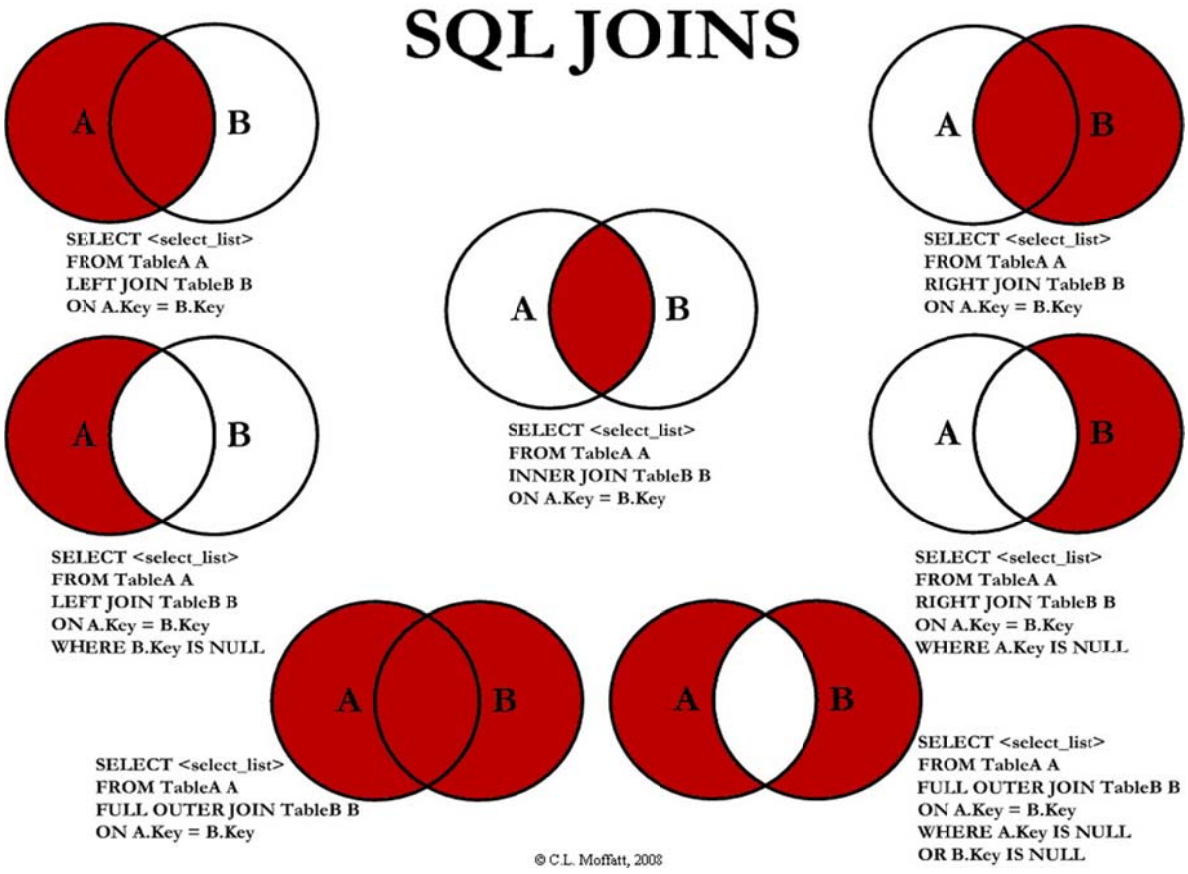
Процедура генерації таблиці, що містить результати з'єднання двох таблиць за допомогою оператора SELECT, полягає у наступному:

1. Формується декартовий добуток таблиць, зазначених у конструкції FROM.
2. Якщо у запиті є конструкція WHERE, застосування умов пошуку до кожного рядка таблиці декартового добутку та збереження в таблиці лише тих рядків, які задовольняють заданим умовам.
3. Для кожного рядка, що залишився, визначається значення кожного елемента, зазначеного в списку вибірки SELECT, в результаті чого формується окремий рядок результуючої таблиці.
4. Якщо у вихідному запиті є конструкція SELECT DISTINCT, з результуючої таблиці видаляються всі рядки-дублікати.
5. Якщо запит, що виконується, містить конструкцію ORDER BY, здійснюється переупорядкування рядків результуючої таблиці.

Зовнішні об'єднання таблиць

При виконанні операції з'єднання дані з двох таблиць комбінуються з утворенням пар зв'язаних рядків, в яких значення стовпців, що зіставляються, є однаковими. **Якщо одне із значень у зіставлюваному стовпці однієї таблиці не збігається з жодним із значень у зіставлюваному стовпці іншої таблиці, то відповідний рядок видаляється з результуючої таблиці.** Саме це правило застосовувалося у всіх розглянутих вище прикладах з'єднання таблиць. Стандартом ISO передбачений інший набір операторів з'єднань, названих **зовнішніми з'єднаннями.**

Іноді може знадобитися, щоб рядок з одного відношення був представлений у результатах з'єднання, навіть якщо в іншому відношенні немає відповідного значення. Ця мета може бути досягнута за допомогою зовнішнього з'єднання.



У зовнішньому з'єднанні в результуючу таблицю містяться також рядки, що не задовольняють умові з'єднання.

Звичайне (внутрішнє) з'єднання цих таблиць виконується за допомогою наступного оператора SQL:

Таблиця 1. Таблиця СпецКурс

gkey	title
01	Бази даних
02	ООП. С++
03	ООП. JAVA

Таблиця 2. Таблиця Студент

stud_num	fio
101	Постова Ольга
102	Ряба Анастасія
103	Слободчикова Тетяна
104	Кулик Кирило
105	Капуста Юлія

Таблиця 3. Таблиця СтудентСпецкурс

fkey_student	fkey_course
101	01
104	01
101	03
103	03
104	03
105	03

Приклад 1. Звичайне (внутрішнє) з'єднання цих таблиць виконується за допомогою наступного оператора SQL:

```
SELECT fio, title
FROM СтудентСпецкурс INNER JOIN СпецКурс ON fkey_course=gkey
INNER JOIN Студент ON fkey_student=stud_num
```

Результати виконання цього запиту наведено в табл. 4.

Таблиця 4. Результат внутрішнього з'єднання таблиць

fio	title
Постова Ольга	Бази даних
Кулик Кирило	Бази даних
Постова Ольга	ООП. JAVA
Слободчикова Тетяна	ООП. JAVA
Кулик Кирило	ООП. JAVA
Капуста Юлія	ООП. JAVA

Зверніть увагу, що у вихідних даних немає назви спецкурсу ООП. С++ як і студента з прізвищем Петров Петро. Якщо в результуючу таблицю потрібно включити і ці рядки, що не мають відповідності, то слід використовувати зовнішнє з'єднання. Існують три типи зовнішнього з'єднання: *ліве, праве* та *повне*. Розглянемо особливості кожного з них на наведених нижче прикладах.

Ліве зовнішнє з'єднання

Приклад 2. Перерахуйте всі спецкурси та студентів, які записані на них, включаючи ті спецкурси, на які не записано жодного зі студентів.

Використовуємо ліве зовнішнє з'єднання цих двох таблиць, яке має такий вигляд:

```
SELECT title, fio
FROM СпецКурс  LEFT JOIN СтудентСпецкурс ON fkey_course=gkey
LEFT JOIN Студент ON fkey_student=stud_num
```

СпецКурс		СтудентСпецкурс		Студент	
gkey	**title**	**fkey_student**	**fkey_course**	**stud_num**	**Fio**
01	Бази даних	101	01	101	Постова Ольга
01	Бази даних	104	01	104	Кулик Кирило
02	ООП. С++	NULL	NULL	NULL	NULL
03	ООП. JAVA	101	03	101	Постова Ольга
03	ООП. JAVA	103	03	103	Слободчикова Тетяна
03	ООП. JAVA	104	03	104	Кулик Кирило
03	ООП. JAVA	105	03	105	Капуста Юлія

Результати виконання цього запиту наведено в таблиці. У цьому прикладі за рахунок застосування лівого зовнішнього з'єднання в результуючу таблицю потрапили не тільки ті спецкурси, в яких є відповідність у таблиці `СтудентСпецкурс`, але також той рядок першої з таблиць, що з'єднуються (лівої), яка не знайшла собі відповідності в другій таблиці (правої). У цьому рядку всі поля другої таблиці заповнені значеннями NULL.

Таблиця 5. Результат виконання запиту на прикладі 2

title	fio
Бази даних	Постова Ольга
Бази даних	Кулик Кирило
ООП. C++	NULL
ООП. JAVA	Постова Ольга
ООП. JAVA	Слободчикова Тетяна
ООП. JAVA	Кулик Кирило
ООП. JAVA	Капуста Юлія

Повне зовнішнє з'єднання

Приклад 3.

Перерахуйте всіх студентів із зазначенням спецкурсів, на які вони записані, включаючи тих студентів, які не записані на жодний зі спецкурсів. А також спецкурси, на яких немає студентів.

Використовуємо повне зовнішнє з'єднання цих таблиць, яке має такий вигляд:

```
SELECT title, fio
FROM СпецКурс  FULL OUTER JOIN СтудентСпецкурс ON fkey_course=gkey
FULL OUTER JOIN Студент ON ref_student=stud_num
```

Результати виконання цього запиту наведено в табл. 10. У разі повного зовнішнього з'єднання в результуючу таблицю поміщаються не тільки ті два рядки, які мають однакові значення в стовпцях, що зіставляються, але і всі інші рядки вихідних таблиць, які не знайшли собі відповідності. У цих рядках всі стовпці тієї таблиці, у якій було знайдено відповідності, заповнюються значеннями NULL.

Таблиця 10. Результат виконання запиту на прикладі 3

title	fio
Бази даних	Постова Ольга
Бази даних	Кулик Кирило
NULL	Ряба Анастасія
ООП. C++	NULL
ООП. JAVA	Постова Ольга
ООП. JAVA	Слободчикова Тетяна
ООП. JAVA	Кулик Кирило
ООП. JAVA	Капуста Юлія

Збережені процедури

Збережені процедури (Stored Procedures) дозволяють переносити частину прикладних функцій, пов'язаних з обробкою даних, у саму БД.

Усі оператори SP – це єдиний пакет. Рядок, що обробляється командою execute – інтерпретується як пакет.

Можливості:

- Умовні оператори IF THEN ELSE
- Цикли WHILE або FOR
- Блоки інструкцій
- Іменовані змінні

- TRY-CATCH
- RAISEERROR(msg,severity,state[,arguments])

@ — ідентифікатор локальної змінної (користувача).
@@ — ідентифікатор глобальної змінної (вбудованої).
У SP, є ім'я і список вхідних і вихідних параметрів.

Конструкція SP:

```
CREATE PROC proc_name
     @input_param_name_1 param_type,
     …                                          } Вхідні параметри
     @input_param_name_n param_type,

     @output_param_name_1 param_type OUTPUT,
     …                                          } Вхідні параметри
     @output_param_name_m param_type OUTPUT
AS
--Оголошення локальних змінних
DECLARE @local_param_name_1 param_type[,...]
BEGIN
          --Оператори SQL
   --Інші конструкції
END
```

Виклик процедури:

```
DECLARE @output_param_name_1 param_type
EXECUTE proc_name  @input_param_value_1 , ...
                   @output_param_name_1  OUTPUT, ...
```

Текст процедури можна зашифрувати, і ніхто не побачить операторів SQL. Збережена процедура компілюється лише один раз і після цього багаторазово викликається. При цьому немає необхідності постійно створювати план виконання SQL операторів, що збільшує продуктивність.

Завдання

1. Розробити 5 запитів на зовнішнє з'єднання до індивідуальної бази даних, вказавши їхній фізичний зміст. Всі запити мають бути виконані за допомогою конструкцій об'єднання: LEFT JOIN, RIGHT JOIN.
2. Розробити 2 збережені процедури до індивідуальної бази даних, вказавши їхнє фізичне значення. Серед наведених збережених процедур, повинні бути процедури, які мають вхідні та вихідні параметри.

Перелік документів для включення до звіту ІР № 3

- титульний лист;
- постановка задачі;
- схема БД у вигляді таблиць та зв'язків між ними;
- привести реалізацію п'яти запитів до індивідуальної бази даних, які потребують використання зовнішнього (лівостороннього або правостороннього) об'єднання;
- привести реалізацію двох збережених процедур до індивідуального завдання.

Зразки виконання запитів ІР № 3
(на прикладі навчальної бази даних)

1. Вивести список товарів, які жодного разу не були замовлені.

```
SELECT MFR_ID, PRODUCT_ID
FROM PRODUCTS LEFT JOIN ORDERS
ON MFR = MFR_ID AND PRODUCT = PRODUCT_ID
WHERE ORDER_NUM IS NULL
```

2. Вивести для кожного клієнта кількість його заказів. Для тих клієнтів, що не зробили жодного заказу, повинно навпроти кількості заказів стояти 0.

```
SELECT COMPANY,  COUNT(ORDER_NUM) as "Amount of Orders"
FROM CUSTOMER LEFT JOIN ORDERS
ON CUST_NUM = CUST
GROUP BY COMPANY
```

3. Створити процедуру, яка б збільшувати фактичний обсяг продажів співробітникам відповідного віку (наприклад, як премії).

```
--створення збереженої процедури

CREATE PROCEDURE add_sales
     @age INTEGER, --оголошення змінних
     @sales MONEY
AS
     BEGIN

UPDATE salesreps
SET quota = quota+@sales
WHERE age=@age;

END

--виконання процедури
EXECUTE add_sales 41 , 100.000
```

4. Створити процедуру, яка б повертала ідентифікатор офісу в даному місті.

```
--створення збереженої процедури
CREATE PROCEDURE get_id_city

     @v_city VARCHAR(15),
     @v_id INT OUTPUT --вихідний параметр
AS
BEGIN
     SELECT @v_id=office
     FROM offices
     WHERE city=@v_city
END
```

```
--виконання процедури
DECLARE @id INT
EXECUTE get_id_city @v_city = 'New York',  @v_id = @id OUTPUT
SELECT @id
```

Питання для самоконтролю

1. Чому складні збереженні процедури при повторному запуску працюють, як правило, швидше, ніж при першому запуску?
2. У чому переваги збережених процедур?
3. У чому відмінності між пакетами, збереженими процедурами і тригерами?
4. Що таке Transact-SQL?
5. У чому відмінність внутрішнього і зовнішнього з’єднання?
6. Які три типи курсорів підтримує SQL Server?
7. Перерахуйте відмінності між функціями і збереженими процедурами.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. Database Systems: The Complete Book.
2. Hernandez Michael J. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design 3rd Edition. Addison-Wesley Professional. 2021.
3. Alan Beaulieu. Learning SQL: Master SQL Fundamentals 3rd Edition. O'Reilly. 2020.
4. Peter Carter. Pro SQL Server 2019 Administration: A Guide for the Modern DBA 2nd ed. Edition. Apress. 2019.
5. Itzik Ben-Gan. T-SQL Fundamentals (Developer Reference) 3rd Edition. Microsoft Press. 2016.

ДОДАТКИ

Додаток № 1

Титульний лист індивідуальної роботи

Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Кафедра теоретичної та прикладної інформатики

Звіт по дисципліні
Вступ до SQL баз даних
Індивідуальне завдання № __

Студента:_____

Групи:_____

Необхідний термін здачі завдання:_____

Фактичний термін здачі завдання:_____

Кількість балів:_____

Харків рік