МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

М. В. Добролюбова, М. В. Філіппова, О. М. Маркіна

ПРОГРАМУВАННЯ БАЗ ДАНИХ

ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою
«Інформаційні вимірювальні технології»
спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка»

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензент *Помазун, О. М.*, к.е.н., доцент, доцент, кафедра інформаційних систем в економіці, ІІТЕ, КНЕУ імені Вадима Гетьмана

Гармаш, О. В., к.т.н., доцент, доцент, кафедра акустичних та мультимедійних електронних систем, ФЕЛ, КПІ ім. Ігоря Сікорського

Відповідальний редактор *Богомазов, С. А.*, к.т.н., доцент

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 8 від 24.06.2021 р.) за поданням Вченої ради Приладобудівного факультету (протокол № 5/21 від 31.05.2021 р.)

Електронне мережне навчальне видання

Добролюбова Марина Валеріївна, канд. техн. наук, доц.
Філіппова Марина Вячеславівна, канд. техн. наук, доц.
Маркіна Ольга Миколаївна, канд. техн. наук, доц.

ПРОГРАМУВАННЯ БАЗ ДАНИХ
ПРАКТИКУМ

Програмування баз даних. Практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка» / М. В. Добролюбова, М. В. Філіппова, О. М. Маркіна ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,88Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 164 с.

Навчальний посібник забезпечує курс комп'ютерних практикумів дисципліни «Програмування баз даних» та знайомить студентів з концепціями написання програм з використанням SQL Server та мови структурованих запитів T-SQL при роботі з реляційними базами даних. В рамках комп'ютерних практикумів розглядаються не лише основи, але і теми, пов'язані з оптимізацією запитів і проєктуванням бази даних.

Навчальний посібник є важливим для студентів зазначеної спеціальності, оскільки при опрацюванні вимірювальної інформації повноцінну інформаційно-вимірювальну систему важко уявити без наявності зв'язку з базою даних.

ПЕРЕДМОВА

Навчальний посібник «Програмування баз даних. Практикум» складено авторами на основі багаторічного досвіду викладання дисциплін, пов'язаних з теоретичними положеннями, розробкою, програмуванням та адмініструванням баз даних. Він відповідає освітній програмі «Інформаційні вимірювальні технології» за спеціальністю 152 «Метрологія та інформаційно-вимірювальна техніка».

Курс комп'ютерних практикумів призначений для організації самостійної роботи студентів з вивчення і систематизації знань за темами: середовище проєктування та управління БД – SQL Server Management Studio, основи мови опису даних (DDL) та мови маніпулювання даними (DML), проєктування баз даних, операції реляційної алгебри, представлення, індексування, процедури і функції, транзакції та тригери. Курс комп'ютерних практикумів містить теоретичний матеріал, приклади, загальні і індивідуальні завдання, рекомендовані ресурси та контрольні питання з відповідних тем, необхідні для закріплення здобутих знань і навичок.

ЗМІСТ

Комп'ютерний практикум 1

Вступ до SQL

Мета роботи: розгляд основних понять БД та мови запитів T-SQL; розгляд типів даних в T-SQL; вивчення інструментів розробки БД; створення бази даних та таблиць.

Теоретичні відомості

- ***База даних*** – сукупність зв'язаних даних, організована за певними правилами, які передбачають загальні принципи опису, збереження та маніпулювання даними, та незалежна від прикладних програм.
- ***Реляційна база даних*** – сукупність зв'язаних даних, що зберігаються у двовимірних таблицях.
- ***Система управління базами даних (СУБД)*** – система програмного забезпечення, яка дозволяє оброблювати звернення до бази даних, які надходять від прикладних програм.
- ***Реляційна СУБД (РСУБД)*** – СУБД, яка управляє реляційними базами даних.
- ***Представлення*** – це спосіб виведення обмеженого набору стовпців з реальної таблиці у вигляді віртуальної таблиці.
- ***Об'єкт користувач*** – ідентифікатор деякої особи, яка бажає увійти до системи.
- T-SQL містить 7 категорій типів даних:
  - цілі числа;
  - числа з фіксованою комою;
  - числа з плаваючою комою;
  - дата та час;
  - текстові дані;
  - двійкові дані;
  - типи користувача.
- MS SQL Server підтримує всі основні прості типи даних, які використовуються в сучасних мовах програмування.

Таблиця 1 – Типи даних в MS SQL Server

Тип	Розмір (байт)	Діапазон значень
Цілі числа		
`Bit`	1 байт	Перший біт в таблиці буде займати один байт, однак наступні сім біт в цій таблиці будуть зберігатись в тому ж байті
`Int`	4 байти	(-2 147 483 648; +2 147 483 647)
`BigInt`	8 байт	$(-2^{63}; +2^{63}-1)$

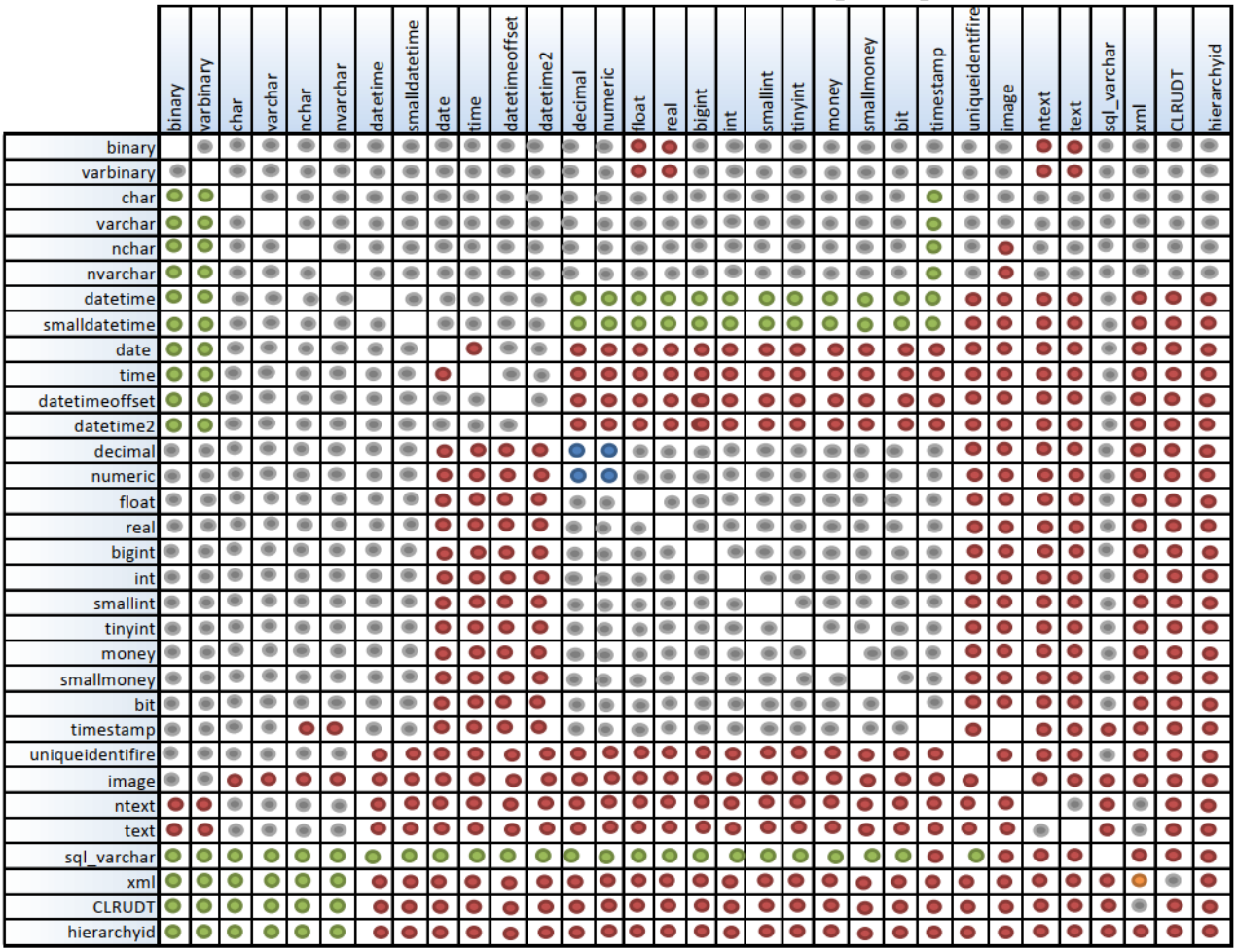
Продовження Таблиці 1

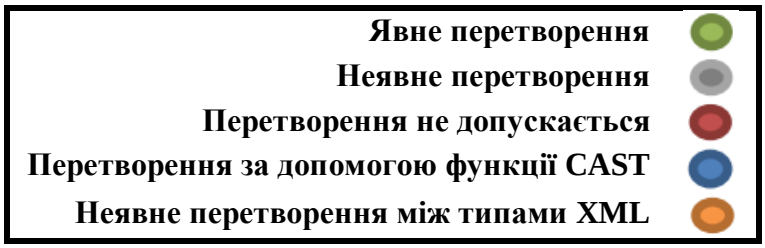
Тип	Розмір (байт)	Діапазон значень
SmallInt	2 байти	(-32 768; +32 767)
TynyInt	1 байт	(0; 255)
Числа з фіксованою комою		
Decimal (Numeric)	8 байт	(-10^{38}-1; $+10^{38}$-1)
Money	8 байт	Грошовий формат, (-2^{63}; $+2^{63}$), з чотирма знаками після коми
SmallMoney	4 байти	Грошовий формат, (-2 147 483 648; +2 147 483 647)
Числа з плаваючою комою		
Float		(-1,79E+308; +1,79E+308)
Дата та час		
DateTime	8 байт	Діапазон значень від 1 січня 1753 року до 31 грудня 9999 року з точністю до трьох сотих секунди
DateTime2	6-8 байт	Новий тип даних, підтримує точність до 0,1 мс
SmallDateTime	4 байти	Діапазон значень від 1 січня 1900 року до 6 червня 2079 року з точністю одна хвилина
DateTimeOffset	8-10 байт	Аналогічний типу **DateTime**, але зберігає ще зсув відносно часу UTC (універсальний синхронізований час)
Date	3 байти	Зберігає лише дату. Діапазон значень від 1 січня 0001 року до 31 грудня 9999 року
Time	3-5 байт	Зберігає лише час з точністю до 0,1 мс
Текстові дані		
Char		Рядок фіксованої довжини. Максимальна довжина рядка 8000 символів
VarChar	до 2^{31} байт	Рядок змінної довжини. Максимальна довжина рядка 8000 символів, але при використанні ключового слова «max» може зберігати до 2^{31} символів
Nchar		Рядок фіксованої довжини в Юнікоді. Максимальна довжина рядка 4000 символів
NvarChar	до 2^{31} байт	Рядок змінної довжини в Юнікоді. Максимальна довжина рядка 4000 символів, але при використанні ключового слова «max» може зберігати до 2^{31} символів

Продовження Таблиці 1

Тип	Розмір (байт)	Діапазон значень
Ntext (застарілий, не використовується)		Аналогічний типу **Text**, але призначений для роботи з Юнікод. Зараз рекомендується використовувати **nvarchar(max)**. Залишений для зворотної сумісності.
Двійкові дані		
Binary		Дозволяє зберігати двійкові дані розміром до 8000 байт
VarBinary	до 2^{31} байт	Тип даних змінної довжини. Дозволяє зберігати до 8000 байт, але при використанні ключового слова «max» до 2^{31} байт, (**varbinary(max)**)
Image (застарілий, не використовується)		Використовувався для збереження великих об'ємів даних, зараз рекомендується використовувати **varbinary(max)**. Залишений для зворотної сумісності.
Інші типи даних		
Table		Особливий тип даних,який використовується в основному для тимчасового збереження таблиць і для передачі в якості параметра в функції
Hierarchyll		Використовується для відображення положення в ієрархічній структурі
Sql_variant		Тип даних, який зберігає значення різних типів даних, які підтримуються MS SQL Server
XML		Дозволяє зберігати XML-дані
Cursor	1 байт	Тип даних для змінних або вихідних параметрів зберігаємих процедур, які містять посилання на курсор
Timestamp / rowversion	8 байт	Тип даних, який представляє автоматично сформовані унікальні двійкові числа в базі даних. Його значення генерується БД автоматично при вставці або зміні запису
Uniqueldentifier	16 байт	Представляє собою GUID (Special Globally Unique Identifire). Гарантується унікальність даного значення

Таблиця 2 – Перетворення типів даних





Практичні приклади

```
--*************************************************************************
--                        Прості типи даних в T-SQL
--*************************************************************************

-- Створюємо змінну з ім'ям - 'myVar', типу Int та присвоюємо їй значення - 7.
DECLARE @myVar INT = 7;

-- Вводимо на екран значення змінної @myVar.
PRINT @myVar;    -- 7

-- Змінній а присвоюємо значення - 77.
SET @myVar = 77;

-- Вводимо на екран значення змінної @myVar.
PRINT @myVar;    -- 77

--*************************************************************************
--                        Цілі типи даних в T-SQL
--                  Bit, TinyInt, SmallInt, Int, BigInt
--*************************************************************************
```

```
------------------------------------ Bit ---------------------------------

-- Для збереження цілих чисел без знака - Bit. Діапазон значень (0 або 1).
-- При спробі присвоєння значення, відмінного від 0 і 1,
-- це значення буде неявно перетворене в 1.

PRINT 'Bit';

DECLARE @myVar Bit = 0;
PRINT @myVar;

SET @myVar = 1;
PRINT @myVar;

SET @myVar = -7; -- (-7) буде неявно перетворене в 1.
PRINT @myVar;    -- @myVar = 1

------------------------------------ TinyInt -------------------------------

-- Для збереження цілих чисел без знака - TinyInt. Діапазон значень від 0 до 255.

PRINT 'TinyInt';

DECLARE @myVar TinyInt = 0;
PRINT @myVar;

SET @myVar = 45;
PRINT @myVar;

SET @myVar = 256; -- Помилка присвоєння: Арифметичне переповнення.
PRINT @myVar;     -- Значення змінної @myVar залишиться рівним попередньому значенню.

------------------------------------ SmallInt -------------------------------

-- Для збереження цілих чисел зі знаком - SmallInt. Діапазон значень від -32 768 до 32 767.

PRINT 'SmallInt';

DECLARE @myVar SmallInt = 32767;
PRINT @myVar;

SET @myVar = -32768;
PRINT @myVar;

 ------------------------------------ Int ---------------------------------

 -- Для збереження цілих чисел зі знаком - Int. Діапазон значень від -2147483648 до 2147483647.

PRINT 'Int'

DECLARE @int Int = 2147483647;
PRINT @int;

SET @int = -2147483648;
PRINT @int;

------------------------------------ BigInt -------------------------------

-- Для збереження цілих чисел зі знаком - BigInt. Діапазон значень від -(2^63) до 2^63-1.

PRINT 'Bigint';

DECLARE @bigint Bigint = 9223372036854775807;
PRINT @bigint;

SET @bigint  = -9223372036854775808;
PRINT @bigint;

--*************************************************************************
--                      Дійсні типи даних в T-SQL
--*************************************************************************

------------------------------------ Float -------------------------------

-- Дійсне зі знаком, з плаваючею комою Float.
-- Діапазон значень від -1.79E+308 до 1.79E+308.
```

```
-- Числовий тип даних з плаваючею комою записується як Float(n),
-- де n - визначає точність(за замовчуванням n = 53)

-- Для типу Float є лише два види точності: 7 та 15 знаків після коми.
-- 7 знаків - при n, в діапазоні від 1 до 24; (виділення 4 байт)
-- 15 знаків - при n, в діапазоні від 25 до 53; (виділення 8 байт)

PRINT 'Float'

DECLARE @float float(24) = 1214782.123;
PRINT @float;

SET @float = 2147482435234412412.43215235987467376 54;
PRINT @float;

---------------------------------- Decimal --------------------------------

-- Дійсне зі знаком, з фіксованою комою Decimal (Numeric).
-- Діапазон значень від -10^38+1 до 10^38-1.

-- Числовий тип даних з фіксованою комою Decimal записується як Decimal(p, s),
-- де p - визначає точність - максимальна кількість знаків, з яких складається повне число
-- (за замовчуванням p = 18, максимальне значення p = 38, мінімальне значення p = 1),
-- а s - визначає масштаб - максимальна кількість знаків після коми.

PRINT 'Decimal / Numeric'

DECLARE @decimal Decimal(5, 3);   -- p = 5, s = 3( не більше p-1)

SET @decimal = 1.42;
PRINT @decimal;

SET @decimal = 2.234654; -- Відбудеться округлення дробової частини.
PRINT @decimal;

SET @decimal = 41.12345; -- Значення округлиться, оскільки після коми більше 3-х знаків.
PRINT @decimal;

--------------------------------- SmallMoney ------------------------------

-- SmallMoney.
-- Діапазон значень від -214 748.3648 до 214 748.3647

PRINT 'SmallMoney';

DECLARE @smoney SmallMoney = 214748.3647;
PRINT @smoney;

SET @smoney = -214748.3648;
PRINT @smoney;

----------------------------------- Money --------------------------------

-- Дійсне зі знаком, з фіксованою комою Money.
-- Діапазон значень від -2^63 до 2^63-1

PRINT 'Money';

DECLARE @money Money = 1.4234;
PRINT @money;

SET @money = 2.234954;
PRINT @money;

--*************************************************************************
--                      Текстові типи даних в T-SQL
--*************************************************************************

-------------------------------------- Char -------------------------------

-- Текстовий тип даних записується як Char,
-- припустима кількість символів від 1 до 8000 символів.

-- Текстовий тип даних записується як Char(n),
-- де n - визначає максимальну кількість символів(максимально n = 8000).

PRINT 'Char'
```

```
DECLARE @char char(5)= 'Hello';
PRINT @char;

SET @char = '1234dfghdf'; -- Частина рядка "fghdf" не запишеться, оскільки припустима кількість
символів (5)
PRINT @char;

------------------------------------ VarChar ------------------------------

-- Текстовий тип даних записується як VarChar,
-- діапазон значень від 1 до 2^31 символів.

-- Текстовий тип даних записується як VarChar(n/max),
-- де n - визначає максимальну кількість символів (максимально n = 8000),
-- або ж замість n записується max, тоді максимальна кількість символів становить 2^31.

PRINT 'VarChar'

DECLARE @vchar varchar(max)='Hello'; -- дозволено записувати 2^31 символів.
PRINT @vchar;

SET @vchar = '1234dfghdf';
PRINT @vchar;

------------------------------------ NChar --------------------------------

-- Текстовий тип даних в Unicode кодуванні записується як NChar,
-- діапазон значень від 1 до 4000 символів.

-- Текстовий тип даних записується як NChar(n),
-- де n - визначає максимальну кількість символів (максимально n = 4000).
-- Якщо кількість введених символів меньша n, то різниця заповниться пробілами

PRINT 'NChar'

DECLARE @Nchar Nchar(7) = 'Привіт!';
PRINT @Nchar;

SET @Nchar = '1234 приїхали'; -- Частина рядка "їехали" не запишеться, оскільки припустима
кількість символів (7)
PRINT @Nchar;

------------------------------------ NvarChar ------------------------------

-- Текстовий тип даних в Unicode кодуванні записується як NvarChar,
-- діапазон значень від 1 до 2^31 символів.

-- Текстовий тип даних записується як NvarChar(n/max),
-- де n - визначає максимальну кількість символів (максимально n = 4000),
-- або ж замість n записується max, тоді максимальна кількість символів становить 2^31.

PRINT 'NvarChar'

DECLARE @NvarChar NvarChar(max) = 'Привіт Світ!';
PRINT @NvarChar;

SET @NvarChar = 'Багато багато багато багато багато багато багато багато багато багато багато
слів';
PRINT @NvarChar;
--*************************************************************************
--                    Типи даних дати та часу в T-SQL
--*************************************************************************

------------------------------------ DateTime ------------------------------

-- Типи даних дати та часу DateTime.
-- Діапазон значень від 1 січня 1753 року до 31 грудня 9999 року.

PRINT 'DateTime'

-- CAST (expression AS data_type) - це функція перетворення типів,
-- перетворює значення expression до типу data_type.

DECLARE @datetime DateTime = CAST('2007-05-08 12:35:29.123' AS DateTime);
PRINT @datetime;

SET @datetime = CURRENT_TIMESTAMP; -- Поточні дата та час
PRINT @datetime;
```

```
---------------------------------- DateTime2 -----------------------------

-- Типи даних дати та часу DateTime2.
-- Діапазон значень від 1 січня 1 року до 31 грудня 9999 року.

-- Тип даних дати та часу записується як DateTime2(n),
-- де n - визначає точність до 100нс ( 0<n<7 )

PRINT 'DateTime2';

DECLARE @datetime2 DateTime2(5) = CAST('2007-05-08 12:35:29.1234567' AS DateTime2)
PRINT @datetime2;

SET @datetime2 = CURRENT_TIMESTAMP; -- Поточні дата та час
PRINT @datetime2;

------------------------------ SmallDateTime -----------------------------

-- Типы данных даты и времени SmallDateTime.
-- Діапазон значень від 1 січня 1900 року до 6 червня 2079 року.

PRINT 'SmallDateTime';

DECLARE @smalldatetime SmallDateTime = CAST('2007-05-08 12:35:29' AS SmallDateTime)
PRINT @smalldatetime;

SET @smalldatetime = CURRENT_TIMESTAMP; -- Поточні дата та час
PRINT @smalldatetime;

------------------------------ DateTimeOffSet -----------------------------

-- Типы данных даты и времени DateTimeOffSet.
-- Визначає дату, об'єднану з часом дня, з врахуванням часового поясу в 24-годинному форматі.
-- Діапазон значень від 1 січня 1 року до 31 грудня 9999 року.

-- Тип даних дати та часу записується як DateTimeOffSet(n),
-- де n - визначає точність до 100нс (0<n<7)

PRINT 'DateTimeOffSet';

DECLARE @dateTimeOffSet DateTimeOffSet(5) = CAST('2007-05-08 12:35:29.1234567 +12:15' AS
DateTimeOffSet)
PRINT @dateTimeOffSet;

SET @dateTimeOffSet = CURRENT_TIMESTAMP; -- Поточні дата та час
PRINT @dateTimeOffSet;

---------------------------------- Date ----------------------------------

-- Типы данных даты Date.
-- Діапазон значень від 1 січня 1 року до 31 грудня 9999 року.

PRINT 'Date';

DECLARE @date Date = CAST('2007-05-08 12:35:29.1234567' AS Date)
PRINT @date;

SET @date = CURRENT_TIMESTAMP; -- Поточні дата та час
PRINT @date;

---------------------------------- Time ----------------------------------

-- Типи даних часу Time.
-- Тип даних часу записується як Time(n),
-- де n - визначає точність до 100нс (0<n<7)

PRINT 'Time';

DECLARE @time Time(5) = CAST('2007-05-08 12:35:29.1234567' AS Time);
PRINT @time;

SET @time = CURRENT_TIMESTAMP;
PRINT @time;

--*************************************************************************
--                    Типи двійкових даних в T-SQL
--*************************************************************************
---------------------------------- Binary ----------------------------------
```

```
-- Типи двійкових даних Binary.
-- Діапазон значень від 1 до 8000 байт.

-- Тип двійкових даних записується як Binary(n),
-- де n - визначає максимальну кількість байт (максимально n = 8000).

PRINT 'Binary'

DECLARE @binary Binary(1) = 16;
PRINT @binary;

---------------------------------- VarBinary -------------------------------

-- Типи двійкових даних VarBinary.
-- Діапазон значень від 1 до 2^31 байт.

-- Тип двійкових даних записується як VarBinary(n/max),
-- де n - визначає максимальну кількість байт(максимально n = 8000),
-- або ж замість n записується max, тоді максимальна кількість символів становить 2^31.

PRINT 'VarBinary'

DECLARE @varBinary VarBinary(max) = 2147483647;
PRINT @varBinary;

--*****************************************************************************
--                              Створення Бази Даних.
--*****************************************************************************

-- Створюємо базу даних з ім'ям DreamHomeDB.
CREATE DATABASE DreamHomeDB
ON                                          -- Задаємо параметри Бази Даних.
(
       NAME = 'DreamHomeDB',         -- Вказуємо логічне ім'я БД (використовується при зверненні
до БД).
       FILENAME = 'D:\DreamHome\DreamHomeDB.mdf',          -- Вказуємо Фізичне повне ім'я файла
БД.
       SIZE = 10MB,                                  -- Задаємо початковий розмір файла БД.
       MAXSIZE = 100MB,                              -- Задаємо максимальный розмір файла БД.
       FILEGROWTH = 10MB                             -- Задаємо значення, на яке буде
збільшуватись розмір файла БД.
)
LOG ON                                        -- Задаємо параметри журнала Бази Даних.
(
       NAME = 'LogDreamHomeDB ',                                  -- Вказуємо логічне ім'я БД
(використовується при зверненні до журналу БД).
       FILENAME = 'D:\DreamHome\DreamHomeDB.ldf',        -- Вказуємо Фізичне повне ім'я файла
журналу БД.
       SIZE = 5MB,                          -- Задаємо початковий розмір файла журналу БД.
       MAXSIZE = 50MB,                      -- Задаємо максимальный розмір файла журналу БД.
       FILEGROWTH = 5MB                     -- Задаємо значення, на яке буде збільшуватись розмір
файла журналу БД.
)
COLLATE Cyrillic_General_CI_AS -- Задаємо кодування для баз даних за замовчуванням

-- Виділити з 6 по 22 рядок і натиснути F5.

-------------------------------------------------------------------------------

-- Виводимо інформацію про Базу Даних - DreamHomeDB.
EXECUTE sp_helpdb DreamHomeDB;

--*****************************************************************************
--                  Створення Таблиці в Базі Даних DreamHomeDB.
--*****************************************************************************

-- УВАГА!
-- Вказуємо явно ім'я Бази Даних, яку необхідно використовувати, оскільки існує ймовірність
-- створення таблиці в Базі Даних, що вказана у викідному списку на панелі інструментів.
-- Наприклад: Часто, помилково створюють таблиці в БД master і здається, що таблиця не
створилась.
USE DreamHomeDB
GO

-------------------------------------------------------------------------------
-- Створюємо таблицю з ім'ям Brunch, яка буде містити сім стовпців.
-- Перший стовпець з ім'ям Branch_No, типу SmallInt з заданим автоінкрементом.
-- Другий стовпець з ім'ям Street, типу Varchar, розмірністю в 25 символів.
```

```
-- Третій стовпець з ім'ям Area, типу Varchar, розмірністю в 15 символів.
-- Четвертий стовпець з ім'ям City, типу Varchar, розмірністю в 15 символів.
-- П'ятий стовпець з ім'ям Postcode, типу Char, розмірністю в 8 символів.
-- Шостий стовпець з ім'ям Tel_No, типу Char, розмірністю в 13 символів.
-- Сьомий стовпець з ім'ям Fax_No, типу Char, розмірністю в 13 символів.

CREATE TABLE Brunch
(
        -- Ключове слово IDENTITY задає початкове значення та встановлює автоінкремент.
        -- За замовчуванням значення першої комірки дорівнює 1 і з кожним новим записом
збільшується на 1.
        Branch_No smallint IDENTITY NOT NULL,
        Street Varchar(25) NOT NULL,
        Area Varchar(15) NULL,
        City Varchar(15) NOT NULL,
        Postcode char(8) NULL,
        Tel_No char(13) NOT NULL,
        Fax_No char(13) NULL

)
GO -- Кінець пакету інструкцій.
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Створіть базу даних з ім’ям «MyDB»

Завдання 3

У створеній БД MyDB створіть 3 таблиці:
1-а містить імена та номери телефонів співробітників деякої компанії.
2-а містить інформацію про їх зарплату та посади.
3-я містить інформацію про їх сімейний стан, дату народження та місце проживання.

Завдання 4

У створеній БД DreamHomeDB створіть 5 таблиць: Staff, Property_for_Rent, Renter, Qwner, Viewing.

Branch	(Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff	(Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent	(Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter	(Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner	(Ono, FName, LName, Address, Tel_No)
Viewing	(Rno, Pno, Date, Comment)

Завдання 5

Зайдіть на сайт MSDN.
Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп’ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

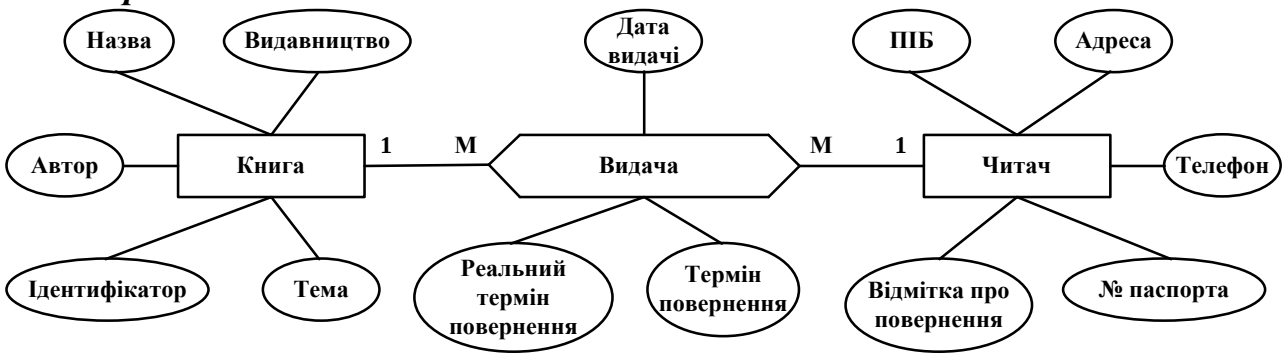
Індивідуальні завдання

Створіть базу даних вказаної предметної області за наданою ER-діаграмою.

Варіант 1

Предметна область: Бібліотека

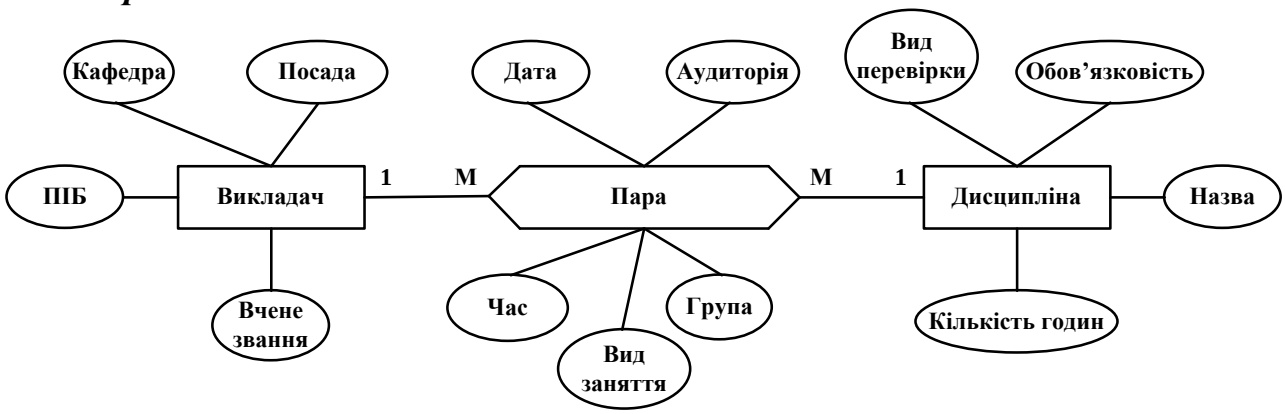
ER-діаграма:



Варіант 2

Предметна область: Університет

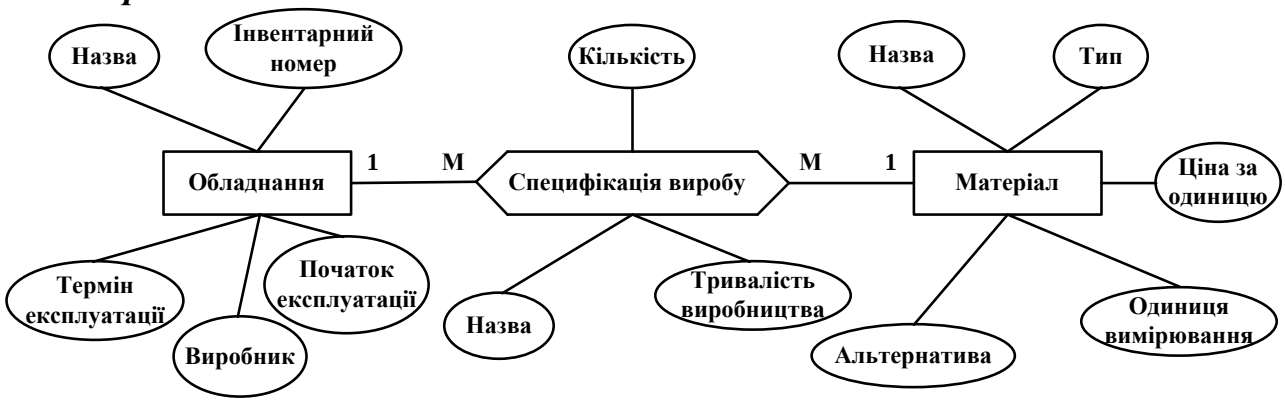
ER-діаграма:



Варіант 3

Предметна область: Виробництво

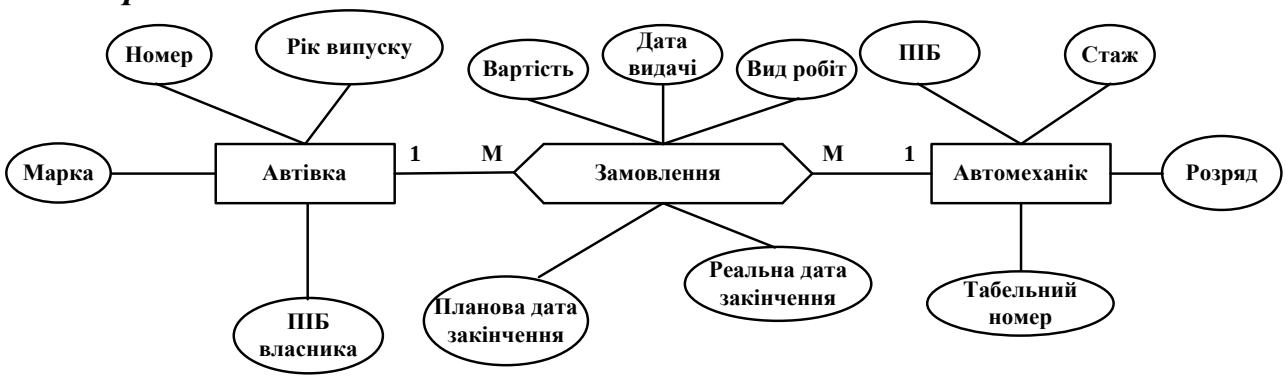
ER-діаграма:



Варіант 4

Предметна область: Автомайстерня

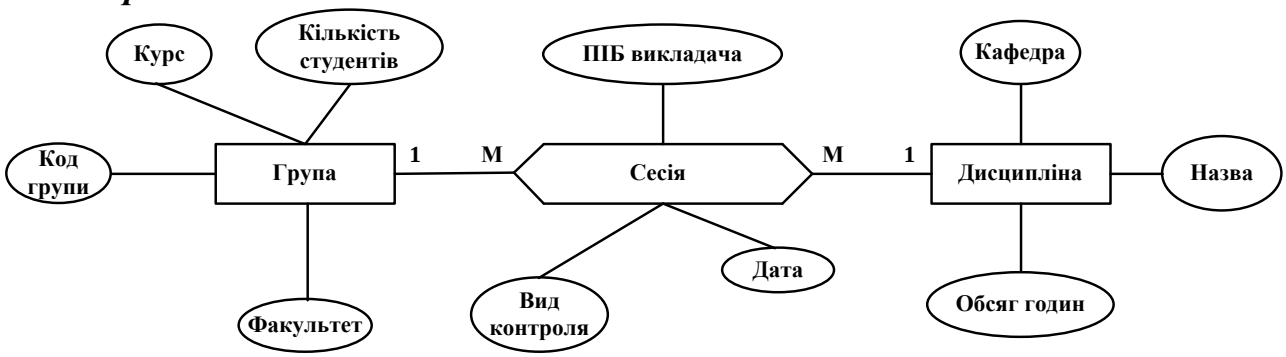
ER-діаграма:



Варіант 5

Предметна область: Сесія

ER-діаграма:

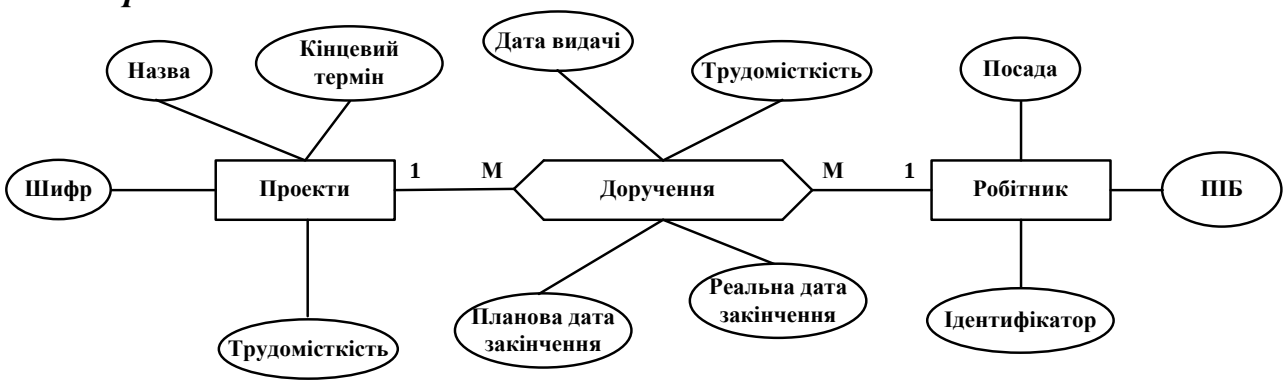


Категорія дисципліни – гуманітарна, комп'ютерна, математична, загальноінженерна тощо. Вид контроля – залік, екзамен.

Варіант 6

Предметна область: Керування проєктом

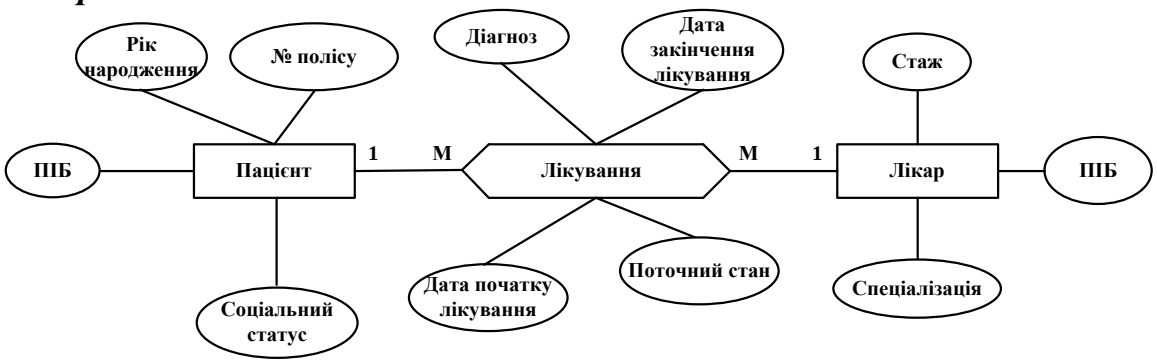
ER-діаграма:



Варіант 7

Предметна область: Поліклініка

ER-діаграма:

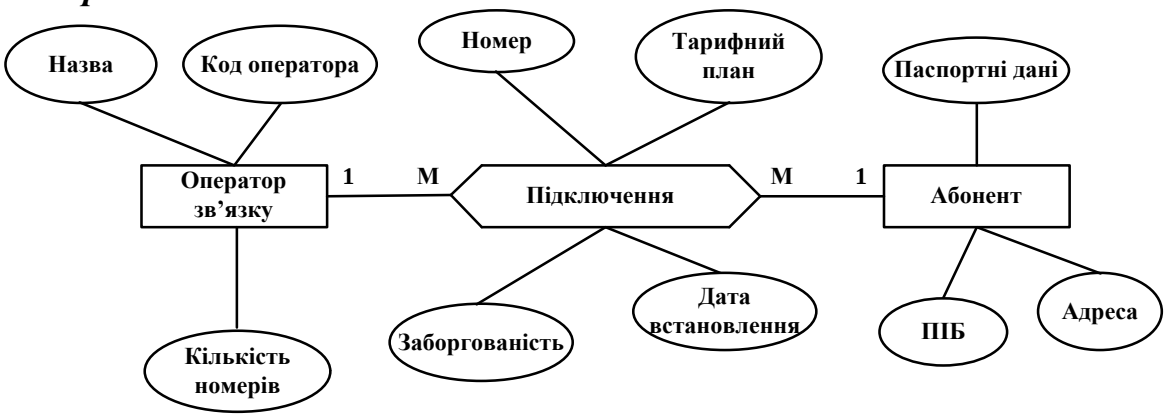


Поточний стан – середньої тяжкості, тяжкий, направлений до стаціонару, помер. Соціальний статус пацієнта – що навчається, працюючий, тимчасово непрацюючий, інвалід, пенсіонер. Спеціалізація лікаря – терапевт, хірург тощо.

Варіант 8

Предметна область: Телефонізація

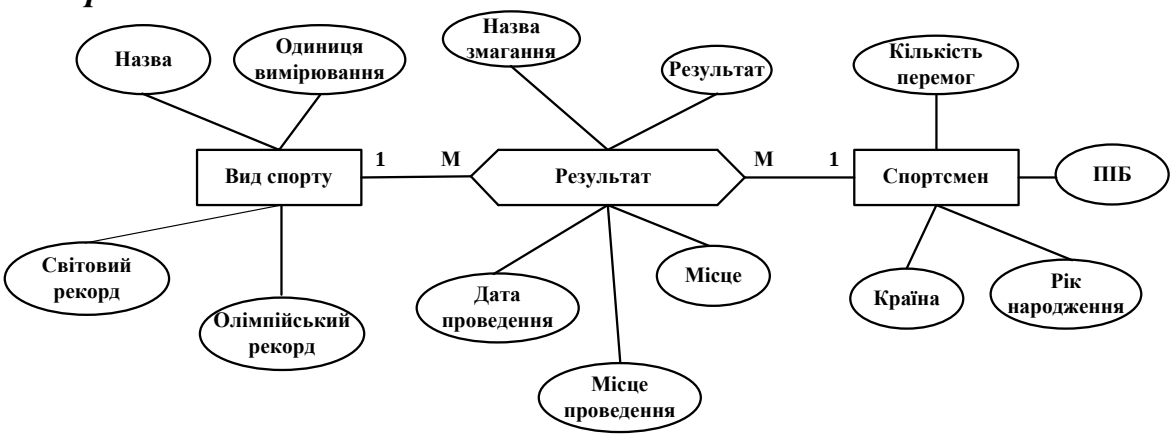
ER-діаграма:



Варіант 9

Предметна область: Спорт

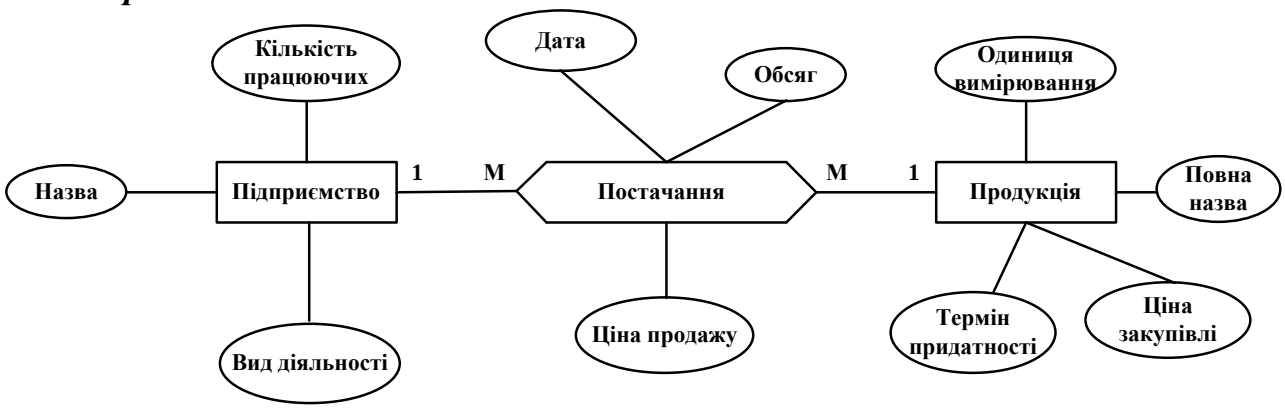
ER-діаграма:



Варіант 10

Предметна область: Сільскогосподарські постачання

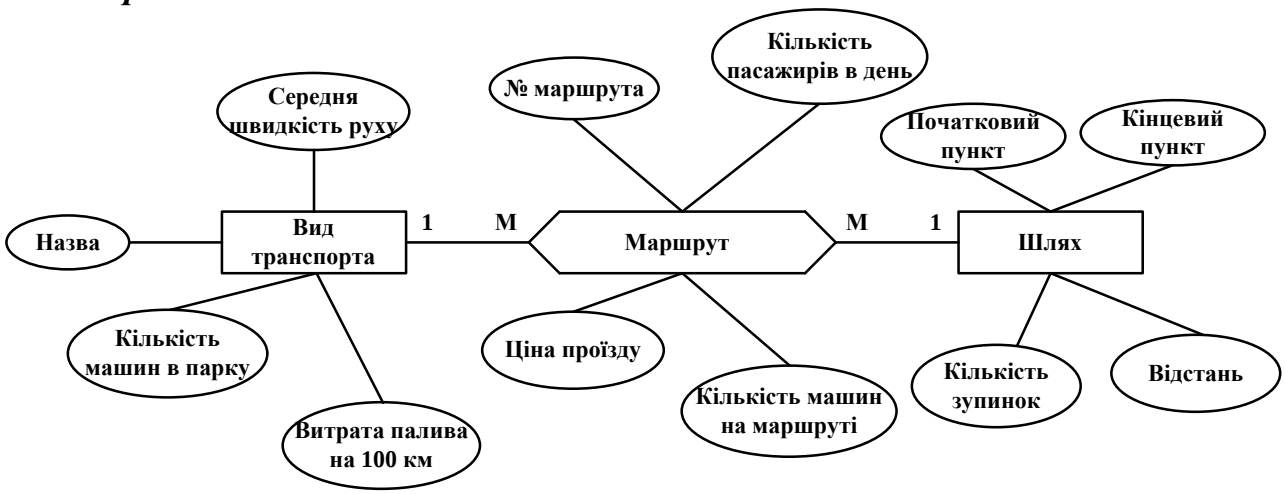
ER-діаграма:



Варіант 11

Предметна область: Міський транспорт

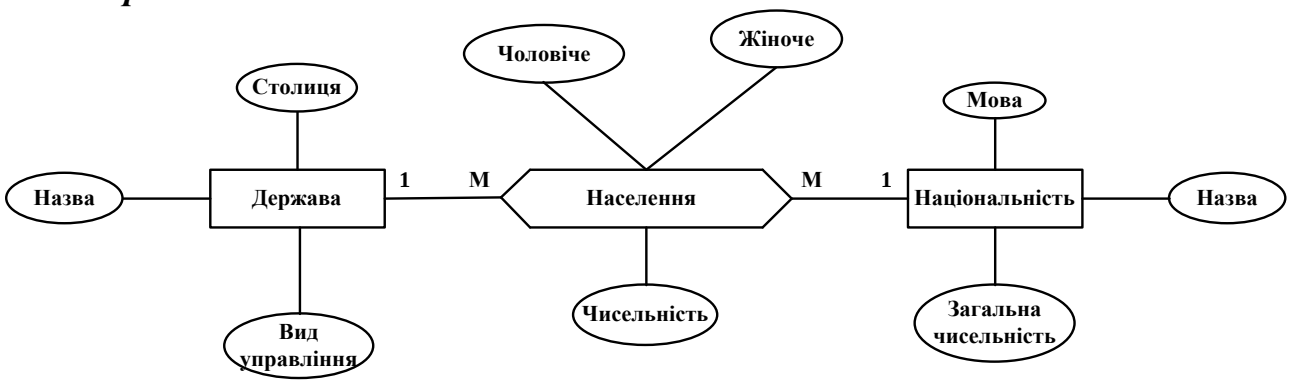
ER-діаграма:



Варіант 12

Предметна область: Географія

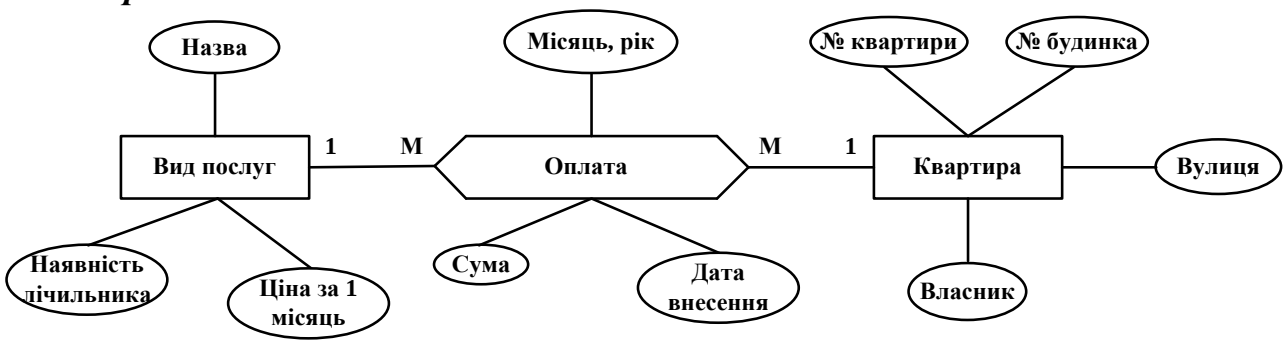
ER-діаграма:



Варіант 13

Предметна область: Управління будинком

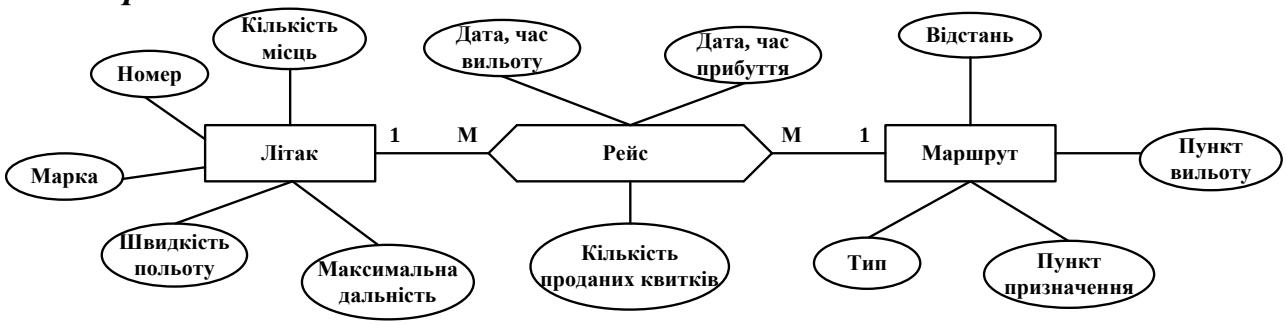
ER-діаграма:



Варіант 14

Предметна область: Аеропорт

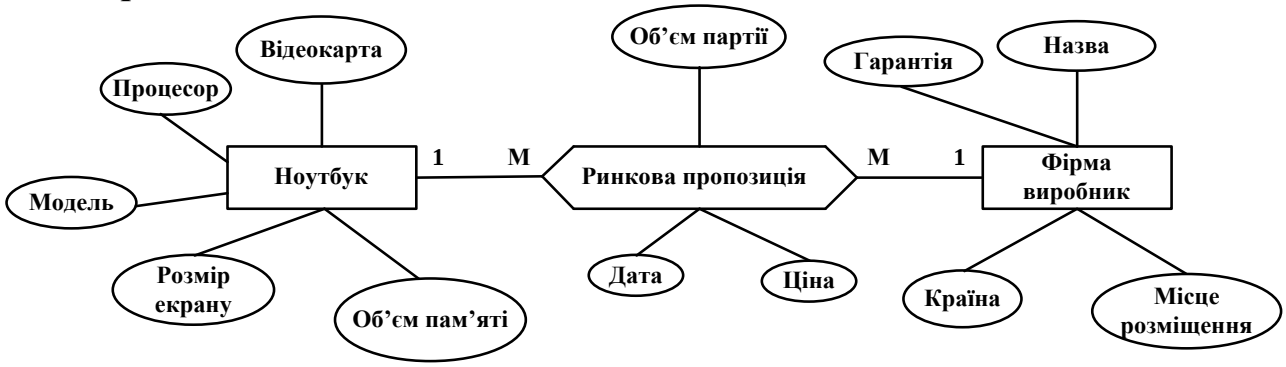
ER-діаграма:



Варіант 15

Предметна область: Ринок ПК

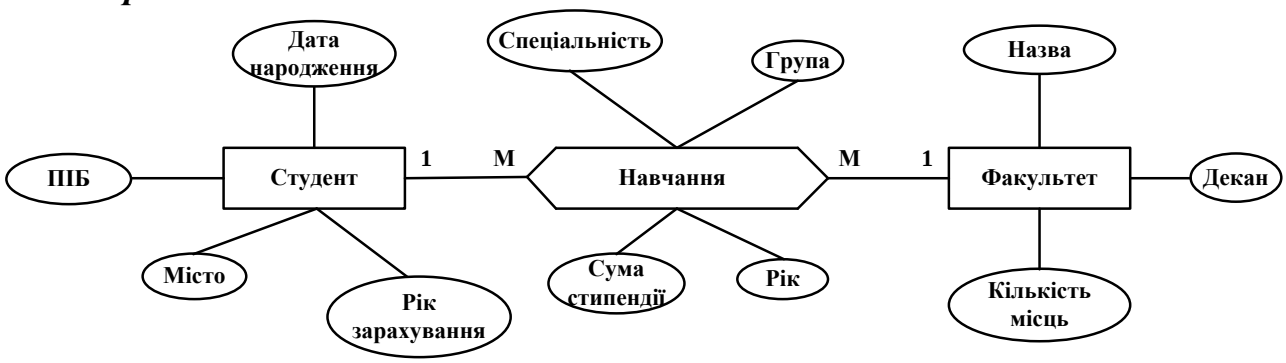
ER-діаграма:



Варіант 16

Предметна область: Деканат

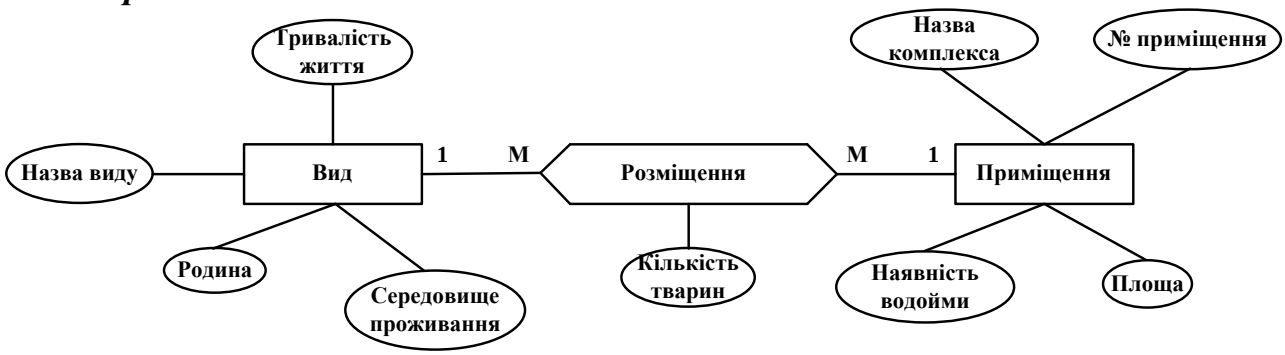
ER-діаграма:



Варіант 17

Предметна область: Зоопарк

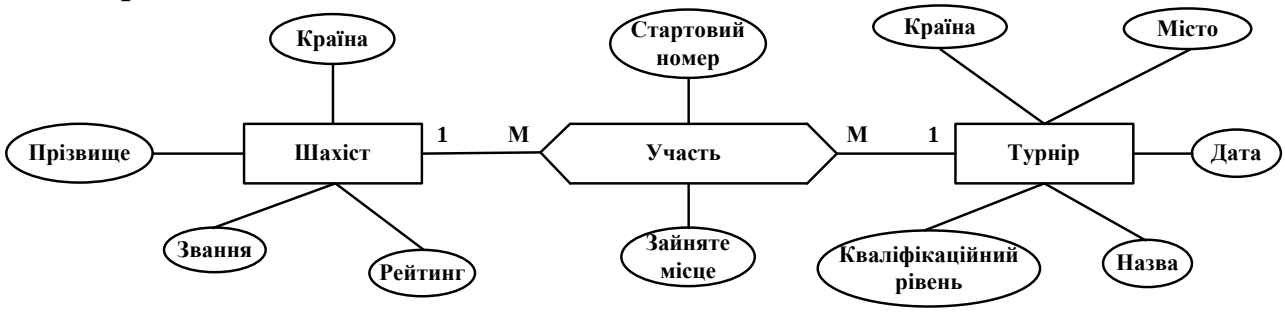
ER-діаграма:



Варіант 18

Предметна область: Шахи

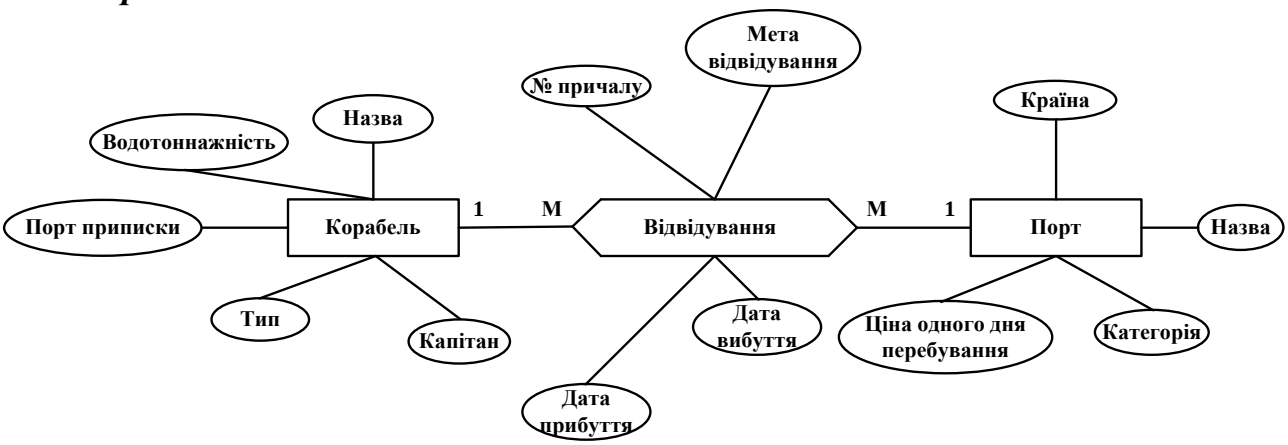
ER-діаграма:



Варіант 19

Предметна область: Судноплавство

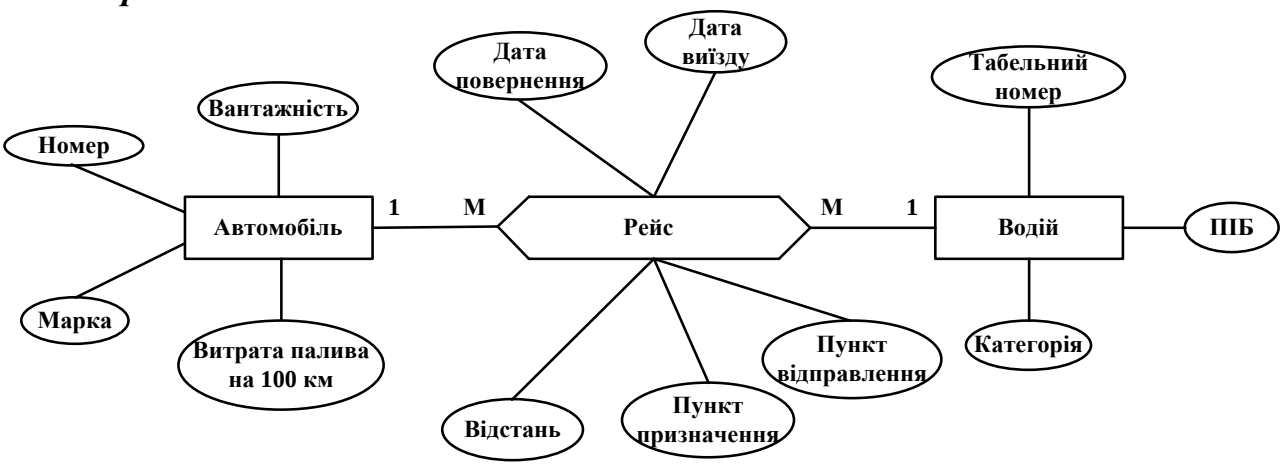
ER-діаграма:



Варіант 20

Предметна область: Автотранспортне підприємство

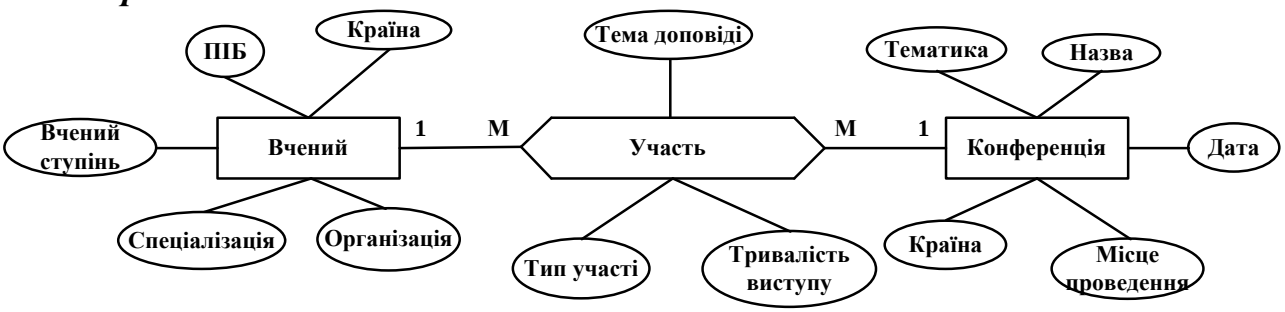
ER-діаграма:



Варіант 21

Предметна область: Наукові конференції

ER-діаграма:

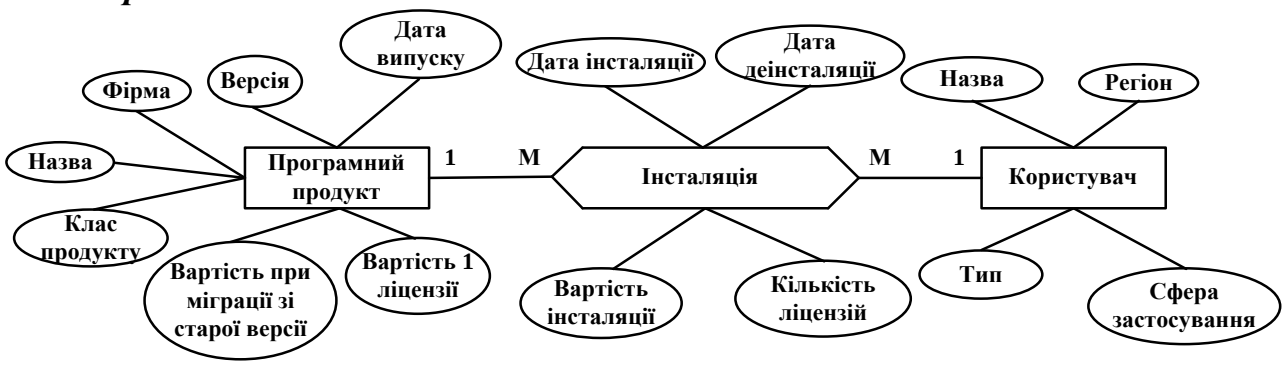


Тип участі: доповідь, повідомлення, стендова доповідь, оргкомітет тощо.

Варіант 22

Предметна область: Програмні продукти

ER-діаграма:



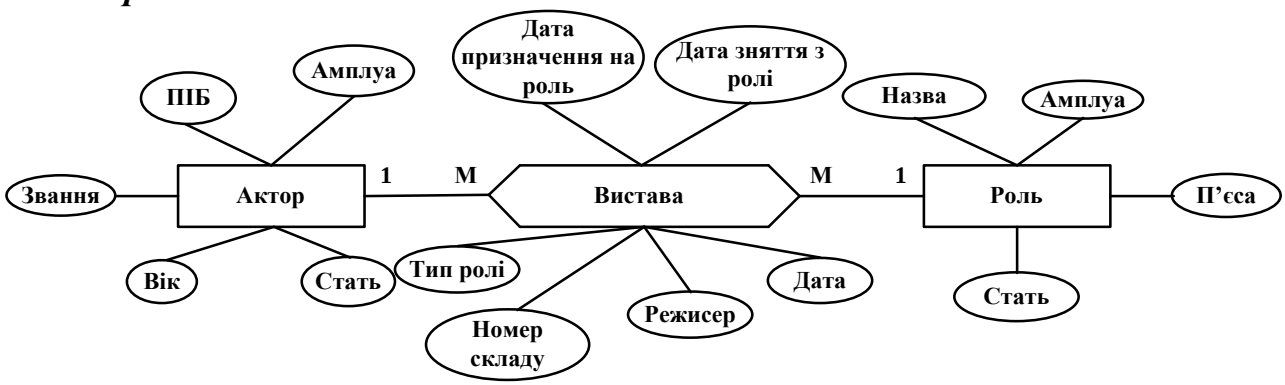
Клас: ОС, сервер додатків, СУБД, Web-сервер, система програмування тощо.

Тип користувача: індивідуальний, корпоративний, спільний, груповий тощо.

Варіант 23

Предметна область: Театр

ER-діаграма:



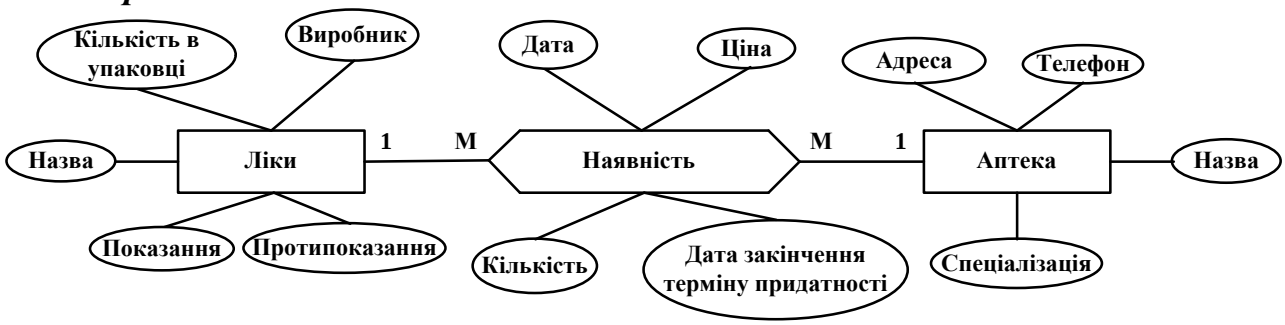
Амплуа: герой-коханець, інженю, злодій тощо.

Тип ролі: головна, друга, епізод, статист тощо.

Варіант 24

Предметна область: Довідникова аптек

ER-діаграма:

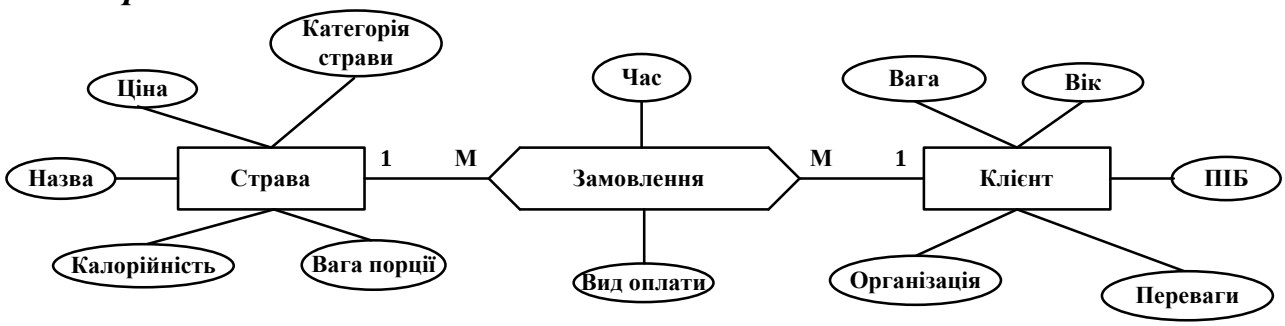


Тип: пігулки, мікстура, мазь тощо.

Варіант 25

Предметна область: Ресторан

ER-діаграма:

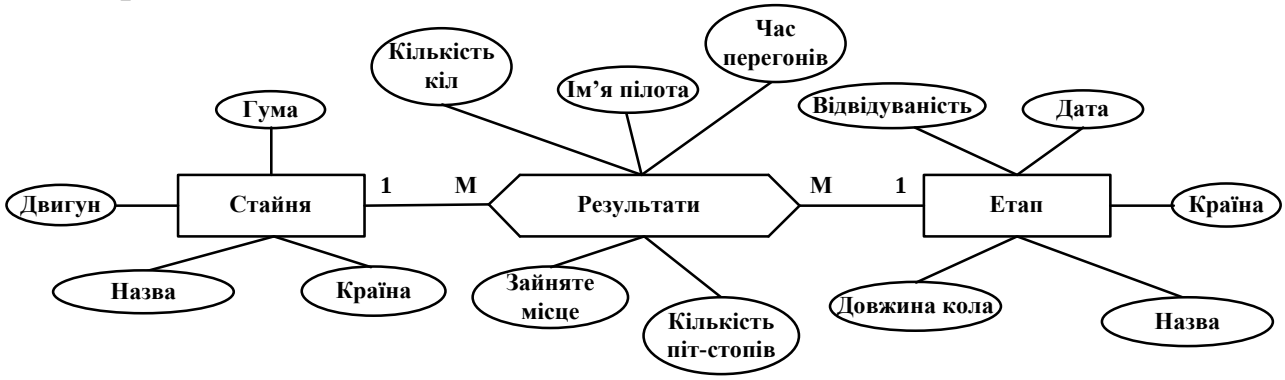


Категорія страви: перша, друга, гарнір, десерт тощо.

Варіант 26

Предметна область: Формула-1

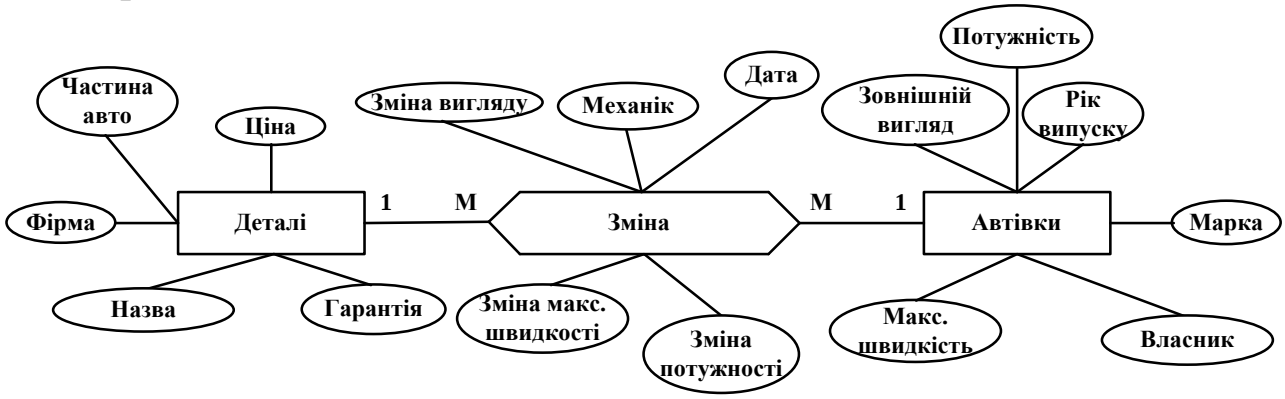
ER-діаграма:



Варіант 27

Предметна область: Тюнінг автомобілей

ER-діаграма:

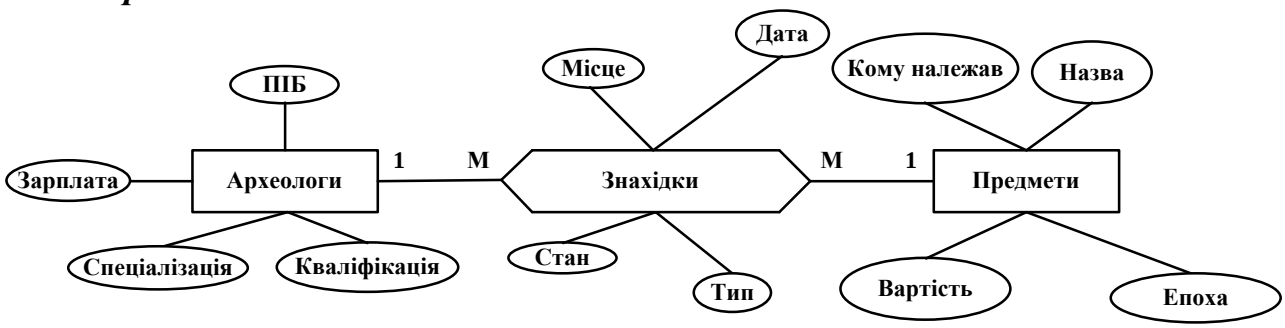


На автівці може встановлюватись лише одна деталь кожної назви.

Варіант 28

Предметна область: Археологія

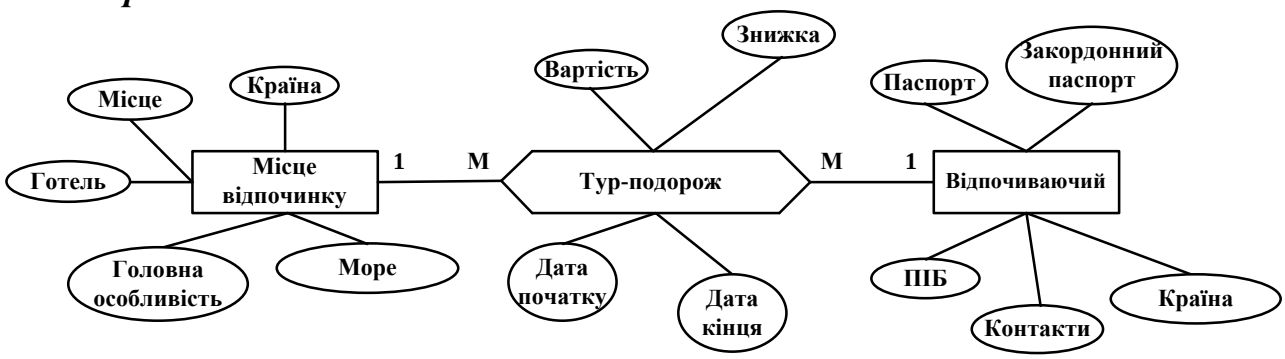
ER-діаграма:



Варіант 29

Предметна область: Тур-фірма

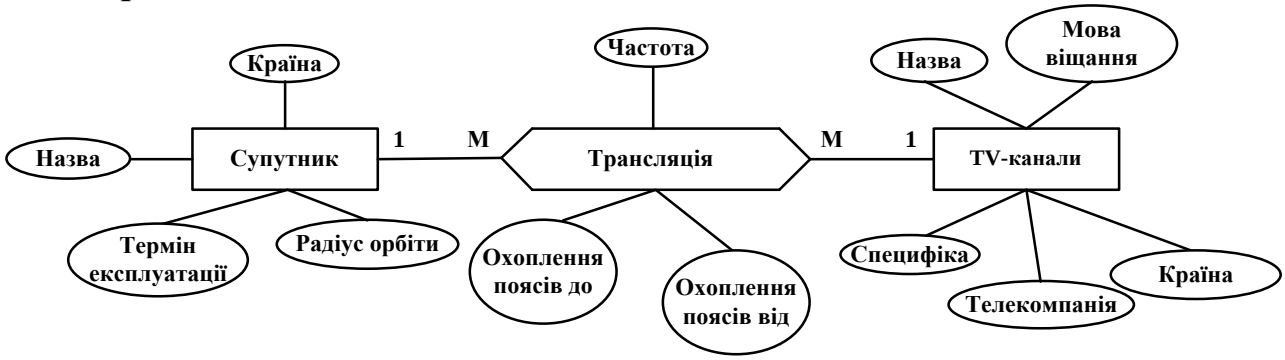
ER-діаграма:



Варіант 30

Предметна область: Телемовлення

ER-діаграма:



Рекомендовані ресурси

MSDN (Microsoft Developer Network) – ресурс підрозділу компанії Microsoft, відповідального за взаємодію з розробниками. Підрозділ працює як інформаційний сервіс.
www.msdn.com
MSDN: Microsoft SQL Server
https://www.microsoft.com/en-in/sql-server/sql-server-2019
https://www.microsoft.com/en-us/sql-server/sql-server-downloads

MSDN: SQL Management Studio
https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15

MSDN: Електронна документація по SQL Server
https://docs.microsoft.com/uk-ua/sql/?view=sql-server-ver15

Контрольні питання

1. Що таке СУБД?
2. Які категорії типів даних є в T-SQL?
3. В чому різниця між СУБД та РСУБД?
4. Що таке база даних?
5. Що таке представлення?
6. Що таке об’єкт користувач?

Комп'ютерний практикум 2

Запити. Маніпулювання даними

Мета роботи: знайомство з операторами INSERT, UPDATE, DELETE та SELECT мови запитів T-SQL.

Теоретичні відомості

- ***Мова SQL (Structured Query Language)*** – мова структурованих запитів, яка застосовується для створення, модифікації та управління даними в реляційній базі даних.
- ***Мова Transact-SQL (T-SQL)*** – це діалект мови SQL, який застосовується в СУБД Microsoft SQL Server. Всі додатки, які взаємодіють з екземпляром MS SQL Server, незалежно від їх реалізації та інтерфейсу користувача, надсилають серверу інструкції мовою T-SQL.
- ***Data Definition Language (DDL)*** – родина комп'ютерних мов програмування, які використовуються в базах даних для визначення схеми бази даних. Основні оператори мови DDL: CREATE, ALTER, DROP.
- ***Data Manipulation Language (DML)*** – родина комп'ютерних мов програмування, які використовуються в базах даних для управління даними: отримання, вставки, видалення, зміни. Мови DML поділяються на два типи:
  - процедурні (procedural DMLs) – для опису дій над даними;
  - декларативні (declarative DMLs) – для опису самих даних.
- Основні оператори мови DML: SELECT, INSERT, UPDATE, DELETE.
- ***Вибірка даних*** – це сукупність, яка виокремлена із загальної сукупності даних та володіюча оголошеними властивостями.
- Оператор SELECT дозволяє робити вибірку одного або декількох рядків/стовпців з однієї або декількох таблиць.
- Загальний формат оператора SELECT:
  SELECT [DISTINCT | ALL] {* | [column_expression [AS new_name]] [...]}
  FROM table_name [alias] [...]
  [WHERE condition]
  [GROUP BY column_list] [HAVING condition]
  [ORDER BY column_list]
  Тут параметр column_expression – ім'я стовпця або вираз з декількох імен; параметр table_name – ім'я існуючих в базі даних таблиці або представлення, до яких необхідно отримати доступ; параметр alias – це скорочення, яке встановлюється для імені таблиці table_name.

- Обробка елементів оператора SELECT виконується в наступній послідовності:
  - FROM – визначаються імена таблиці або декількох таблиць, які використовуються;
  - WHERE – виконується фільтрація рядків об'єкта відповідно до заданих умов;
  - GROUP BY – утворюються групи рядків, що мають одне й те саме значення в зазначеному стовпці;
  - HAVING – фільтруються групи рядків об'єкта відповідно до зазначеної умови;
  - SELECT – встановлюється, які стовпці повинні бути присутніми у вихідних даних;
  - ORDER BY – визначається впорядкованість результатів виконання оператора.
- Порядок речень і фраз в операторі SELECT не може бути змінений. Тільки два речення оператора – SELECT і FROM – є обов'язковими, всі інші речення і фрази можуть бути опущені.
- Операція SELECT є закритою: результат запиту до таблиці являє собою іншу таблицю.
- Оператор INSERT дозволяє додавати новий рядок до таблиці або представлення.
- Існує дві форми оператора INSERT:
  - для вставки єдиного рядка в зазначену таблицю з наступним форматом:
    INSERT INTO table_name [(column_list)]
    VALUES (data_value_list)
    Тут table_name – ім'я таблиці або оновлюваного представлення бази даних; column_list – список; data_value_list – список значень даних, який повинен наступним чином відповідати параметру column_list: кількість елементів в обох списках має бути однаковою, має існувати пряма відповідність між позицією одного й того ж елемента в обох списках, типи даних елементів списку data_value_list повинні бути сумісні з типом даних відповідних стовпців таблиці.
  - для копіювання множини рядків однієї таблиці в іншу таблицю з наступним форматом:
    INSERT INTO table_name [(column_list)]
    SELECT. . .
    Тут параметри table_name і column_list мають той самий формат і зміст, що і при вставці в таблицю одного рядка; речення SELECT може бути будь-яким допустимим оператором SELECT; рядки, що вставляються в зазначену таблицю, в точності відповідають рядкам

результуючої таблиці, створеної при виконанні вкладеного запиту; всі обмеження для першої форми оператора INSERT можуть бути застосовні і для другої.

- Оператор UPDATE дозволяє змінювати існуючі дані в таблиці або представленні.
- Формат оператора UPDATE:
  UPDATE table_name
  SET column_name1 = data_va1ue1 [, column_name2 = data_va1ue2 ...]
  [WHERE search_condition]
  Тут параметр table_name – ім'я таблиці бази даних або ім'я оновлюваного представлення; в реченні SET вказуються імена одного або більше стовпців, дані яких необхідно змінити; речення WHERE є необов'язковим (якщо воно опущено, значення зазначених стовпців будуть змінені у всіх рядках таблиці; якщо присутнє, то оновлені будуть тільки ті рядки, які задовольняють умові пошуку, заданій в параметрі search_condition); параметри data_va1ue представляють нові значення відповідних стовпців, які мають бути сумісні з ними за типом даних.
- Оператор DELETE дозволяє видаляти рядки з таблиць та представлень.
- Формат оператора DELETE:
  DELETE FROM table_name
  [WHERE search_condition]
  Тут параметр table_name – ім'я таблиці бази даних або ім'я оновлюваного представлення; параметр search_condition є необов'язковим (якщо він опущений, з таблиці будуть видалені всі існуючі в ній рядки, однак сама таблиця видалена не буде; якщо необхідно видалити не тільки вміст таблиці, але і її визначення, слід використовувати оператор DROP TABLE); якщо речення WHERE присутнє, з таблиці будуть видалені тільки ті рядки, які задовольняють умові відбору, заданій параметром search_condition.
- ***Директиви сценарію*** – це спеціальні команди, які використовуються лише в MS SQL Server. Дані команди допомагають серверу визначати правила роботи зі скриптом та транзакціями. До типових директив сценарію належать: GO, яка інформує програми сервера про закінчення пакету інструкцій T-SQL, та EXEC (EXECUTE), яка виконує процедуру або скалярну функцію.
- ***Оператор*** – це символ, який запускає операцію на виконання. Оператори SQL можна використовувати з іменами об’єктів, константами та змінними, якщо дозволяє вираз (таблиця 1).
- У виразах з багатьма операторами обчислення виконуються у відповідності до пріоритету (таблиця 2).

Таблиця 1 – Види операторів

Позначення	Значення	Позначення	Значення
+	Додавання	=	Оператор перевірки рівності
-	Віднімання	>	Більше
*	Множення	<	Менше
/	Ділення	>=	Більше дорівнює
%	Залишок від ділення	<=	Менше дорівнює
=	Присвоєння	<>	Не дорівнює
&	Побітова операція «AND»	!=	Не дорівнює
|	Побітова операція «OR»	!>	Не більше ніж
^	Побітова операція «XOR»	!<	Не менше ніж

Позначення	Значення
ALL	Істина, якщо кожне порівняння з набору істине
AND	Істина, якщо два булеві вирази істині
ANY	Істина, якщо будь-яке порівняння з набору істине
BETWEEN	Істина, якщо операнд знаходиться в середині діапазону
EXIST	Істина, якщо підлеглий запит містить будь-які рядки
IN	Істина, якщо операнд знаходиться в списку
LIKE	Істина, якщо операнд відповідає шаблону
NOT	Змінює значення булевих операторів на протилежні
OR	Істина, якщо будь-яка з пар мулевих виразів істина
SOME	Істина, якщо будь-яке порівняння з набору істине
+	Конкатенація рядків
-	Від'ємне число
~	Доповнення до одиниці

Таблиця 2 – Пріоритети обчислень в порядку спадання

№	Дія	Опис
1	+, -, ~	Оператори зміни знаку та доповнення до одиниці
2	*, /, %	Оператори множення і ділення
3	+, -, &	Оператори додавання, віднімання та побітового множення
4	=, >, <, >=, <=, <>, !=, !<, !>	Оператори порівняння
5	^, |	Оператори побітового додавання
6	NOT	Логічний оператор заперечення
7	AND	Логічний оператор множення
8	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME	Інші логічні оператори
9	=	Оператор присвоєння

Практичні приклади

```
USE DreamHomeDB; -- Вказуємо БД, яку будемо використовувати
GO

-----------------------------------------------------------------------
                        /* Створюємо таблицю з ім'ям Brunch*/
-----------------------------------------------------------------------
CREATE TABLE Brunch
```

```
(
	Branch_No smallint IDENTITY NOT NULL,
	Street Varchar(25) NOT NULL,
	Area Varchar(15) NULL,
	City Varchar(15) NOT NULL,
	Postcode char(8) NULL,
	Tel_No char(13) NOT NULL,
	Fax_No char(13) NULL
)
GO -- Кінець пакету інструкцій.

-------------------------------------------------------------------------
	-- Процедура sp_help повертає відомості про об'єкт бази даних.
-------------------------------------------------------------------------
EXEC sp_help Brunch

-------------------------------------------------------------------------
                              /* Оператор вставки INSERT */
-------------------------------------------------------------------------

-- Записуємо рядок в таблицю Brunch
INSERT INTO Brunch             -- Ключове слово INTO можно не використовувати.
(Street, Area, City, Postcode, Tel_No, Fax_No) -- Вказування порядку запису даних.
VALUES
('22 Deer Rd', 'Sidcup', 'London', 'SW1 4EH', '0171-886-1212', '0171-886-1214'); -- Дані, що
вводяться.
GO
-------------------------------------------------------------------------

-- Можна явно змінити порядок даних, які записуються.
INSERT INTO Brunch
(Area, Street, City, Postcode, Tel_No, Fax_No) -- Вказування порядку запису даних.
VALUES
('Dyce', '16 Argyll St', 'Aberden', 'AB2 3SU', '01224-67125', '01224-67111'); -- Дані, що
вводяться.
GO
-------------------------------------------------------------------------

INSERT INTO Brunch
 -- Вказування порядку запису даних відсутнє, тому використовується порядок за замовчуванням.
VALUES
('163 Main St', 'Partick', 'Glasgow', 'G11 9QX', '0141-339-2178', '0141-339-4439'); -- Дані, що
вводяться.
GO
-------------------------------------------------------------------------

INSERT INTO Brunch
 -- Вказування порядку запису даних відсутнє, тому використовується порядок за замовчуванням.
VALUES
('163 Main St', 'Glasgow', 'Partick', 'G11 9QX', '0141-339-2178', '0141-339-4439'); -- Дані, що
вводяться (Помилка черговості вводу!).
GO

SELECT * FROM Brunch
-------------------------------------------------------------------------

INSERT INTO Brunch
VALUES
 -- Неможна залишати порожніми поля, які відмічені обмеженням NOT NULL
(NULL, 'Partick', 'Glasgow', 'G11 9QX', '0141-339-2178', '0141-339-4439'); -- Дані, що вводяться.
GO
-------------------------------------------------------------------------

INSERT INTO Brunch
VALUES
-- Допускається залишати порожніми поля, які відмічені обмеженням NULL
('163 Main St', 'Partick', 'Glasgow', 'G11 9QX', '0141-339-2178', null); -- Дані, що вводяться.
GO
-------------------------------------------------------------------------

INSERT INTO Brunch
(Street, Area, City, Postcode, Tel_No, Fax_No)
VALUES -- Починаючи з 2008 SQL, допускається вводити декілька рядків одночасно
('32 Manse Rd', 'Leigh', 'Bristol', 'BS99 1NZ', '0117-916-1170', '0117-776-1114'),
('56 Clover Dr', null, 'London', 'NW10 6EU', '0181-963-1030', '0181-453-7992')
GO
-------------------------------------------------------------------------
```

```
INSERT INTO Brunch
-- Вказування порядку запису даних для декількох рядків не обов'язкове
VALUES
('32 Manse Rd', 'Leigh', 'Bristol', 'BS99 1NZ', '0117-916-1170', '0117-776-1114'),
('56 Clover Dr', null, 'London', 'NW10 6EU', '0181-963-1030', '0181-453-7992');
GO
-------------------------------------------------------------------------
USE DreamHomeDB;
GO
-------------------------------------------------------------------------
--                     Вибірка даних. Запит SELECT
-------------------------------------------------------------------------

-- Робимо вибірку всіх даних з таблиці Brunch.
SELECT * FROM Brunch;

-- Робимо вибірку даних із стовпця Street таблиці Brunch
SELECT Street FROM Brunch;

-- Робимо вибірку даних із стовпця City таблиці Brunch
SELECT City FROM Brunch;

SELECT Tel_No FROM Brunch;

-- Робимо вибірку даних із стовпців Street та City таблиці Brunch
SELECT Street, City FROM Brunch;

-- Робимо вибірку даних із стовпців City, Area, Postcode та Tel_No таблиці Brunch
SELECT City, Area, Postcode, Tel_No FROM Brunch;

-------------------------------------------------------------------------
USE AdventureWorks2019;
GO

/*
   Вибір з таблиці CreditCard всіх:
   типів кредитних карток,
   номерів карток,
   місяця, до якого дійсна картка, та
   року, до якого дійсна картка
*/

SELECT CardType, CardNumber, ExpMonth, ExpYear FROM Sales.CreditCard; -- Sales - схема

-------------------------------------------------------------------------
--                         Конструкція WHERE.
-------------------------------------------------------------------------

-- Робимо вибірку всіх даних з таблиці CreditCard, де значення комірок стовпця CreditCardID рівні
10.
SELECT * FROM Sales.CreditCard
WHERE CreditCardID = 10; -- В конструкції WHERE застосована операція порівняння (=)

-- Робимо вибірку всіх даних з таблиці CreditCard, де значення комірок стовпця CreditCardID менше
10.
SELECT * FROM Sales.CreditCard
WHERE CreditCardID < 10; -- В конструкції WHERE застосована операція порівняння (<)

-- Робимо вибірку всіх даних з таблиці CreditCard,
-- де значення комірок стовпця CreditCardID лежать в діапазоні від 1 до 3 (включно).
SELECT * FROM Sales.CreditCard
WHERE CreditCardID BETWEEN 1 AND 3 -- В конструкції WHERE застосована операція перевірки
діапазону.

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця ExpYear дорівнює 2005 або 2006.
SELECT * FROM Sales.CreditCard
WHERE ExpYear = 2005 OR ExpYear = 2007 -- Логічна операція "АБО"

-- Робимо вибірку всіх даних з таблиці CreditCard,
-- де значення комірок стовпця ExpYear лежать в діапазоні від 2005 до 2006 (включно).
SELECT * FROM Sales.CreditCard
WHERE ExpYear BETWEEN 2005 AND 2007 -- В конструкції WHERE застосована операція перевірки
діапазону.
-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
```

```
-- де значення комірок стовпця ExpYear дорівнює 2005 та значення комірок стовпця CardType
дорівнює 'Vista'.
SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE ExpYear = 2005 AND CardType = 'Vista' -- Логічна операція "І"

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця ExpYear не дорівнює 2006
SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE NOT ExpYear = 2006 -- Логічна операція "НЕ"

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця CardType відповідають шаблону 'ColonialVoice'
SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType LIKE 'ColonialVoice' -- LIKE - операція перевірки відповідності заданому шаблону -
'ColonialVoice'

SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType = 'ColonialVoice'

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця CardType відповідають шаблону 'Dis%'

SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType LIKE 'Dis%' -- Виведення всіх карток, ім'я типу яких починається на  'Dis',
                                        -- знак % означає довільну кількість символів
після Dis.

SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType = 'Dis%'  -- Вибірка порожня, оскільки такого значення як 'Dis%' немає в таблиці

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця CardType відповідають шаблону 'Vis__'
SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType LIKE 'Vis__' --_ - нижнє підкреслення визначає будь-який один символ після Vis

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця CardType відповідають шаблону 'Vis_'
SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType LIKE 'Vis_' -- чому вибірка порожня?

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця CardType відповідають шаблону 'Vis_a'
SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType LIKE 'Vis_a'

-- Робимо вибірку даних із стовпців CardType та ExpYear таблиці CreditCard,
-- де значення комірок стовпця CardType відповідають шаблону '%ist%'
SELECT CardType, ExpYear FROM Sales.CreditCard
WHERE CardType LIKE '%ist%'

-- Робимо вибірку всіх даних з таблиці CreditCard,
-- де значення комірок стовпця ExpMonth дорівнюють одному із значень ряду (1, 5, 11)
SELECT * FROM Sales.CreditCard WHERE ExpMonth IN (1, 5, 11); -- IN (1, 5, 11) визначає значення
ExpMonth, які рівні 1 або 5, або 11

SELECT * FROM Sales.CreditCard WHERE ExpMonth = 1 OR ExpMonth = 5 OR ExpMonth = 11;
-------------------------------------------------------------------------
USE AdventureWorks2019
GO

SELECT BusinessEntityID, FirstName, LastName, MiddleName, ModifiedDate FROM Person.Person

-------------------------------------------------------------------------
--                          Конструкція ORDER BY - ВПОРЯДКУВАТИ ЗА...
-------------------------------------------------------------------------

-- Робимо вибірку даних із стовпців BusinessEntityID, FirstName, LastName, MiddleName та
ModifiedDate,
-- де дані впорядковані за стовпцем FirstName таблиці Person.Person
SELECT BusinessEntityID, FirstName, LastName, MiddleName, ModifiedDate
FROM Person.Person
ORDER BY FirstName;  -- Впорядкувати за ім'ям.

-- ПРобимо вибірку даних із стовпців BusinessEntityID, FirstName, LastName, MiddleName та
ModifiedDate,
-- де дані впорядковані за стовпцем ModifiedDate таблиці Person.Person
SELECT BusinessEntityID, FirstName, LastName, MiddleName, ModifiedDate
```

```
FROM Person.Person
ORDER BY ModifiedDate;  -- Сортувати за ModifiedDate

-- Робимо вибірку даних із стовпців BusinessEntityID, FirstName, LastName, MiddleName та
ModifiedDate,
-- де дані впорядковані за стовпцем FirstName и LastName таблиці Person.Person
SELECT BusinessEntityID, FirstName, LastName, MiddleName, ModifiedDate
FROM Person.Person
ORDER BY FirstName, LastName;  -- Сортувати за Ім'ям та Прізвищем

-- Робимо вибірку даних із стовпців BusinessEntityID, FirstName, LastName, MiddleName та
ModifiedDate,
-- де дані впорядковані за стовпцем FirstName таблиці Person.Person
SELECT BusinessEntityID, FirstName, LastName, MiddleName, ModifiedDate
FROM Person.Person
ORDER BY FirstName ASC; -- Сортування за зростанням. ASC - за замовчуванням.

-- Робимо вибірку даних із стовпців BusinessEntityID, FirstName, LastName, MiddleName та
EmailAddress,
-- де дані впорядковані в порядку спадання за стовпцем FirstName таблиці Person.Contact
SELECT BusinessEntityID, FirstName, LastName, MiddleName, ModifiedDate
FROM Person.Person
ORDER BY FirstName DESC; -- Сортування за спаданням.
-------------------------------------------------------------------------
USE AdventureWorks2019
GO
-------------------------------------------------------------------------
--                          Агрегування даних. Конструкція GROUP BY
-------------------------------------------------------------------------

-- Робимо вибірку даних із стовпців SalesOrderID и OrderQty
-- таблиці Sales.SalesOrderDetail, де значення комірок стовпця SalesOrderID
-- дорівнюють одному із значень ряду (43666, 43660, 43664)
SELECT SalesOrderID, OrderQty
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN (43666, 43660, 43664);

-- Робимо вибірку даних із стовпця SalesOrderID
-- та повертаємого значення агрегованої функції SUM по стовпцю OrderQty
-- таблиці Sales.SalesOrderDetail, де значення комірок стовпця SalesOrderID
-- дорівнюють одному із значень ряду (43666, 43660, 43664),
-- при цьому виконується групування за значеннями стовпця SalesOrderID.
-- Агрегування - процес об'єднання елементів

SELECT SalesOrderID, SUM(OrderQty) -- Агрегована функція SUM()
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN (43666, 43660, 43664)
GROUP BY SalesOrderID;

SELECT SalesOrderID AS ID, SUM(OrderQty) AS Total -- Після ключового слова AS задаємо псевдонім
(alias) для результату функції SUM().
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN (43666, 43660, 43664)
GROUP BY SalesOrderID; -- Агрегування даних.

-- Робимо вибірку повертаємого значення агрегованої функції COUNT таблиці
HumanResources.Employee.
SELECT COUNT(*) AS Emp
FROM HumanResources.Employee; -- Агрегована функція COUNT() виводить кількість рядків в таблиці

-- Робимо вибірку даних із стовпця ProductID та повертаємого значення агрегованої функції COUNT
-- таблиці Sales.SalesOrderDetail, при цьому виконується групування за значеннями стовпця
ProductID.
SELECT ProductID AS Product, COUNT(*) AS [Count]
FROM Sales.SalesOrderDetail
GROUP BY ProductID;
-------------------------------------------------------------------------

-- Робимо вибірку даних із стовпця ProductID та повертаємого значення агрегованої функції COUNT
-- таблиці Sales.SalesOrderDetail, при цьому виконується групування за значеннями стовпця
ProductID,
-- де повертаєме значення агрегованої функції COUNT більше за 3300.
SELECT ProductID AS Product, COUNT(*) AS [Count]
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING COUNT(*) > 3300;       -- HAVING - має використовуватись сумісно з GROUP BY (HAVING
аналогічний WHERE).
```

```
                                                -- HAVING - умова, що застосовується лише до груп.
-------------------------------------------------------------------------

-- Помилка. HAVING - має використовуватись сумісно з GROUP BY.
SELECT ProductID AS Product
FROM Sales.SalesOrderDetail
HAVING ProductID > 10; -- Неправильне використання HAVING (без GROUP BY)

-- WHERE працює до групування, а HAVING працює разом з групуванням.
SELECT SalesOrderID, COUNT(*) AS TOTAL
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN (43666, 43660, 43664)
GROUP BY SalesOrderID
HAVING COUNT(*) > 5;
-------------------------------------------------------------------------
USE DreamHomeDB;
GO

INSERT Brunch
(Street, City, Tel_No)
VALUES ('16 Holhead', 'Aberdeen', '01224-196720');
GO

SELECT * FROM Brunch;

-------------------------------------------------------------------------
--            Оператор UPDATE (зміна (оновлення) даних в таблиці)
-------------------------------------------------------------------------
UPDATE Brunch
SET Tel_No = '01224-777777' -- змінити номер телефону того відділення,
WHERE Street = '16 Holhead'   -- яке знаходиться на вулиці '16 Holhead'
GO

SELECT * FROM Brunch;

-- Можливо змінювати значення одразу в декількох стовпцях
UPDATE Brunch
SET Area = 'Dee',
       Tel_No = '01224-777555'
WHERE Street = '16 Holhead'

SELECT * FROM Brunch;

UPDATE Brunch
SET Tel_No = '77777-777777' -- Якщо не задана конструкція WHERE, то зміниться весь стовпець на
вказане значення
GO

SELECT * FROM Brunch;

-------------------------------------------------------------------------
--               Оператор DELETE (видалення даних з таблиці)
-------------------------------------------------------------------------
-- Видалити всі відділення в місті Лондон
DELETE Brunch
WHERE City = 'London';
SELECT * FROM Brunch;

-- Видалення всіх даних з таблиці за допомогою DELETE.
DELETE Brunch;
SELECT * FROM Brunch;

-- Для видалення всіх даних з таблиці краще використовувати - TRUNCATE TABLE,
-- оскільки TRUNCATE видаляє інформацію з бази швидше за стандартний DELETE.
TRUNCATE TABLE Brunch;
SELECT * FROM Brunch;
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Створіть базу даних з ім'ям «MyDiet», в якій створіть таблицю «DietProduct» з полями: ProductId, Name, ProductNumber, Cost, Count, SellStartDate.
Таблицю Product заповніть 10 записами про продукти, які були придбані в період дієти.

Name	ProductNumber	Cost	Count	SellStsrtDate
Ананас	AH-77214	5$	5	09/25/2020
Авокадо	AK-76541	4.5$	12	09/15/2020
Лимон	LB-42789	3.8$	6	08/10/2020
Яблуко	DF-11198	1.7$	30	10/20/2020
Яблуко	DFN-1527	2.2$	40	09/09/2020
Банан	SS-14157	1.5$	15	10/21/2020
Груша	WE-12120	4$	10	10/24/2020
Томат	KK-77790	3.5$	45	10/15/2020
Огірок	OP-12126	2$	50	10/15/2020
Огірок	OP-12127	2$	50	10/18/2020

Завдання 3

Зробіть вибірку для продукції, кількість якої більше 10 од.
Зробіть вибірку для овочів, ціна якої більше 3$ та тієї, що поступила до продажу з 20/10/2020.

Завдання 4

Ананас та авокадо подорожчали на 50 центів. Змініть запис.

Завдання 5

Зайдіть на сайт MSDN.
Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп'ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Оператори маніпулювання даними:

1. Записати рядок в таблиці Книга, Читач, Видача за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Книга. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Адреса, Телефон таблиці Читач, де дані впорядковані по стовпцю ПІБ за зростанням.
4. Визначити читачів, у яких номер телефону починається з 095.

Варіант 2

Оператори маніпулювання даними:

1. Записати рядок в таблиці Викладач, Пара, Дисципліна за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Викладач. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Вид перевірки таблиці Дисципліна, де дані впорядковані по стовпцю Назва за спаданням.
4. Визначити викладачів з вченим званням «доцент».

Варіант 3

Оператори маніпулювання даними:

1. Записати рядок в таблиці Обладнання, Специфікація виробу, Матеріал за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Обладнання. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Ціна за одиницю таблиці Матеріал, де дані впорядковані по стовпцю Назва за зростанням.
4. Визначити матеріал, ціна за одиницю якого перевищує 200 грн.

Варіант 4

Оператори маніпулювання даними:

1. Записати рядок в таблиці Автівка, Замовлення, Автомеханік за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Автівка. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Стаж таблиці Автомеханік, де дані впорядковані по стовпцю Стаж за спаданням.
4. Визначити автомеханіків, стаж роботи яких перевищує 5 років.

Варіант 5

Оператори маніпулювання даними:

1. Записати рядок в таблиці Група, Сесія, Дисципліна за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Група. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Обсяг годин таблиці Дисципліна, де дані впорядковані по стовпцю Обсяг годин за зростанням.
4. Визначити дисципліну з кількістю годин, рівною 150.

Варіант 6

Оператори маніпулювання даними:

1. Записати рядок в таблиці Проєкти, Доручення, за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Проєкти. Вибрати Робітник дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Посада таблиці Робітник, де дані впорядковані по стовпцю Посада за спаданням.
4. Визначити проєкти, кінцевий термін яких стосується 2018, 2019 та 2020 років.

Варіант 7

Оператори маніпулювання даними:

1. Записати рядок в таблиці Пацієнт, Лікування, Лікар за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Пацієнт. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Стаж таблиці Лікар, де дані впорядковані по стовпцю Стаж за зростанням.
4. Визначити лікарів, стаж яких перевищує 10 років.

Варіант 8

Оператори маніпулювання даними:

1. Записати рядок в таблиці Оператор зв'язку, Підключення, Абонент за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Оператор зв'язку. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Адреса таблиці Абонент, де дані впорядковані по стовпцю ПІБ за спаданням.
4. Визначити оператора зв'язку, код якого починається з 067.

Варіант 9

Оператори маніпулювання даними:

1. Записати рядок в таблиці Вид спорту, Результат Спортсмен за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Вид спорту. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Кількість перемог таблиці Спортсмен, де дані впорядковані по стовпцю Кількість перемог за зростанням.
4. Визначити спортсмена, кількість перемог якого перевищує 5.

Варіант 10

Оператори маніпулювання даними:

1. Записати рядок в таблиці Підприємство, Постачання, Продукція за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Підприємство. Вибрати дані стовпця Повна назва.
3. Вибрати дані стовпців Повна назва, Ціна закупівлі таблиці Продукція, де дані впорядковані по стовпцю Ціна закупівлі за спаданням.
4. Визначити продукцію, ціна закупівлі якої перевищує 250 грн..

Варіант 11

Оператори маніпулювання даними:

1. Записати рядок в таблиці Вид транспорту, Маршрут, Шлях за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Вид транспорту. Вибрати дані стовпця Початковий пункт.
3. Вибрати дані стовпців Початковий пункт, Кінцевий пункт, Відстань таблиці Шлях, де дані впорядковані по стовпцю Відстань за зростанням.
4. Визначити вид транспорту, середня швидкість руху якого коливається від 45 км/год до 60 км/год.

Варіант 12

Оператори маніпулювання даними:

1. Записати рядок в таблиці Держава, Населення, Національність за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Держава. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Загальна чисельність таблиці Національність, де дані впорядковані по стовпцю Загальна чисельність за спаданням.
4. Визначити національність з загальною кількістю 4 млн..

Варіант 13

Оператори маніпулювання даними:

1. Записати рядок в таблиці Вид послуг, Оплата, Квартира за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Вид послуг. Вибрати дані стовпця Власник.
3. Вибрати дані стовпців Власник, Назва вулиці таблиці Квартира, де дані впорядковані по стовпцю Назва вулиці за зростанням.
4. Визначити квартири з номерами від 77 по 190.

Варіант 14

Оператори маніпулювання даними:

1. Записати рядок в таблиці Літак, Рейс, Маршрут за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Літак. Вибрати дані стовпця Пункт призначення.
3. Вибрати дані стовпців Пункт вильоту, Пункт призначення, Відстань таблиці Маршрут, де дані впорядковані по стовпцю Відстань за спаданням.
4. Визначити маршрут з відстанню, яка перевищує 6000 км.

Варіант 15

Оператори маніпулювання даними:

1. Записати рядок в таблиці Ноутбук, Ринкова пропозиція, Фірма виробник за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Ноутбук. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Країна таблиці Фірма виробник, де дані впорядковані по стовпцю Країна за зростанням.
4. Визначити ноутбук з максимальним об’ємом пам’яті.

Варіант 16

Оператори маніпулювання даними:

1. Записати рядок в таблиці Студент, Навчання, Факультет за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Студент. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Кількість місць таблиці Факультет, де дані впорядковані по стовпцю Кількість місць за спаданням.
4. Визначити факультет з кількістю місць, меншою за 60.

Варіант 17

Оператори маніпулювання даними:

1. Записати рядок в таблиці Вид, Розміщення, Приміщення за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Вид. Вибрати дані стовпця Назва комплекса.
3. Вибрати дані стовпців Назва комплекса, Площа таблиці Приміщення, де дані впорядковані по стовпцю Площа за зростанням.
4. Визначити вид з тривалістю життя від 80 до 100 років.

Варіант 18

Оператори маніпулювання даними:

1. Записати рядок в таблиці Шахіст, Участь, Турнір за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Шахіст. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Країна, Місто, Дата таблиці Турнір, де дані впорядковані по стовпцю Дата за спаданням.
4. Визначити турніри, які проводились з 2019 по 2020 роки.

Варіант 19

Оператори маніпулювання даними:

1. Записати рядок в таблиці Корабель, Відвідування, Порт за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Корабель. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Країна, Ціна одного дня перебування таблиці Порт, де дані впорядковані по стовпцю Ціна одного дня перебування за зростанням.
4. Визначити порт з ціною одного дня перебування, вищою за 200 $.

Варіант 20

Оператори маніпулювання даними:

1. Записати рядок в таблиці Автомобіль, Рейс, Водій за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Автомобіль. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Категорія таблиці Водій, де дані впорядковані по стовпцю Категорія за спаданням.
4. Визначити автівку з мінімальною витратою палива на 100 км.

Варіант 21

Оператори маніпулювання даними:

1. Записати рядок в таблиці Вчений, Участь, Конференція за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Вчений. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Країна, Тематика таблиці Конференція, де дані впорядковані по стовпцю Тематика за зростанням.
4. Визначити конференції, які проводились в 2019 році в містах Київ та Харків.

Варіант 22

Оператори маніпулювання даними:

1. Записати рядок в таблиці Програмний продукт, Інсталяція, Користувач за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Програмний продукт. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Тип таблиці Користувач, де дані впорядковані по стовпцю Тип за спаданням.
4. Визначити програмні продукти з класом ОС або СУБД.

Варіант 23

Оператори маніпулювання даними:

1. Записати рядок в таблиці Актор, Вистава, Роль за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Актор. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, П’єса таблиці Роль, де дані впорядковані по стовпцю П’єса за зростанням.
4. Визначити акторів, вік яких знаходиться в діапазоні від 25 до 30 років.

Варіант 24

Оператори маніпулювання даними:

1. Записати рядок в таблиці Ліки, Наявність, Аптека за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Ліки. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Адреса, Телефон таблиці Аптека, де дані впорядковані по стовпцю Телефон за спаданням.
4. Визначити аптеки, номер телефону яких починається з 044.

Варіант 25

Оператори маніпулювання даними:

1. Записати рядок в таблиці Страва, Замовлення, Клієнт за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Страва. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Вік таблиці Клієнт, де дані впорядковані по стовпцю Вік за зростанням.
4. Визначити страви, калорійність яких перевищує 150 ккал.

Варіант 26

Оператори маніпулювання даними:

1. Записати рядок в таблиці Стайня, Результати, Етап за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Стайня. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Довжина кола таблиці Етап, де дані впорядковані по стовпцю Довжина кола за спаданням.
4. Визначити етап з довжиною кола, більшою за 2000 м.

Варіант 27

Оператори маніпулювання даними:

1. Записати рядок в таблиці Деталі, Зміна, Автівки за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Деталі. Вибрати дані стовпця Марка.
3. Вибрати дані стовпців Марка, Рік випуску таблиці Автівки, де дані впорядковані по стовпцю Рік випуску за зростанням.
4. Визначити автівку з максимальною швидкістю.

Варіант 28

Оператори маніпулювання даними:

1. Записати рядок в таблиці Археологи, Знахідки, Предмети за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Археологи. Вибрати дані стовпця Назва.
3. Вибрати дані стовпців Назва, Вартість таблиці Предмети, де дані впорядковані по стовпцю Вартість за спаданням.
4. Визначити предмети, вартість яких перевищує 10 млн. дол..

Варіант 29

Оператори маніпулювання даними:

1. Записати рядок в таблиці Місце відпочинку, Тур-подорож, Відпочиваючий за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Місце відпочинку. Вибрати дані стовпця ПІБ.
3. Визначити тур-подорож, вартість якої перевищує 50 тис. дол..
4. Вибрати кількість днів відпочинку і кількість місць відпочинку в минулому році для кожного клієнта фірми.

Варіант 30

Оператори маніпулювання даними:

1. Записати рядок в таблиці Супутник, Трансляція, ТВ-канали за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Супутник. Вибрати дані стовпця Країна.
3. Вибрати дані стовпців Назва, Специфіка, Мова таблиці ТВ-канали, де дані впорядковані по стовпцю Мова за спаданням.
4. Визначити ТВ-канали з англійською мовою віщання.

Рекомендовані ресурси

T-SQL
https://msdn.microsoft.com/ru-ru/library/ms189826(v=sql.90).aspx

T-SQL: SELECT
https://msdn.microsoft.com/ru-ru/library/ms189499(v=sql.90).aspx

T-SQL: INSERT
https://msdn.microsoft.com/ru-ru/library/ms174335(v=sql.90).aspx

T-SQL: UPDATE
https://msdn.microsoft.com/ru-ru/library/ms177523(v=sql.90).aspx

T-SQL: DELETE
https://msdn.microsoft.com/ru-ru/library/ms189835(v=sql.90).aspx

Контрольні питання

1. Що таке SQL?
2. Що таке T-SQL?
3. Що таке DDL?
4. Що таке DML?
5. В чому полягає різниця між SQL та T-SQL?
6. Які існують різновиди DML?
7. Які існують різновиди операторів DML?
8. Які існують різновиди операторів DDL?
9. З яких операторів та параметрів побудовано загальний формат SELECT?
10. З яких операторів та параметрів побудовано загальний формат INSERT?
11. З яких операторів та параметрів побудовано загальний формат UPDATE?
12. З яких операторів та параметрів побудовано загальний формат DELETE?

Комп'ютерний практикум 3

Основи DDL

Мета роботи: знайомство з операторами для створення, зміни і видалення баз даних та таблиць.

Теоретичні відомості

- ***Data Definition Language (DDL)*** – родина комп'ютерних мов програмування, які використовуються в комп'ютерних програмах для опису структури бази даних.
- Основні оператори мови DML: CREATE, ALTER, DROP.
- Оператор CREATE використовується для визначення нових сутностей.
- Синтаксис оператора CREATE:
  CREATE DATABASE [IF NOT EXISTS] db_name
  Оператор CREATE DATABASE створює базу даних з вказаним ім'ям. Якщо база даних вже існує і не вказано ключовий параметр IF NOT EXISTS, то виникає помилка виконання команди.
- Оператор ALTER використовується для зміни визначень існуючих сутностей.
- Синтаксис оператора ALTER:
  ALTER [IGNORE] TABLE tbl_name alter_spec [, alter_spec ...]

  alter_specification:
  ADD [COLUMN] create_definition [FIRST | AFTER column_name]
  або ADD [COLUMN] (create_definition, create_definition, ...)
  або ADD INDEX [index_name] (index_col_name, ...)
  або ADD PRIMARY KEY (index_col_name, ...)
  або ADD UNIQUE [index_name] (index_col_name, ...)
  або ADD FULLTEXT [index_name] (index_col_name, ...)
  або ADD [CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name, ...)
  [Reference_definition]
  або ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  або CHANGE [COLUMN] old_col_name create_definition
  [FIRST | AFTER column_name]
  або MODIFY [COLUMN] create_definition [FIRST | AFTER column_name]
  або DROP [COLUMN] col_name
  або DROP PRIMARY KEY
  або DROP INDEX index_name
  або DISABLE KEYS

або ENABLE KEYS
або RENAME [TO] new_tbl_name
або ORDER BY col
або table_options

Оператор ALTER TABLE забезпечує можливість змінювати структуру існуючої таблиці.

- Оператор DROP використовується для видалення існуючих сутностей.
- Синтаксис оператора DROP:
  DROP DATABASE [IF EXISTS] db_name
  Оператор DROP DATABASE видаляє всі таблиці в зазначеній базі даних і саму базу. Якщо ви виконуєте DROP DATABASE на базі даних, що символічно пов'язана з іншою, то видаляється як посилання, так і оригінальна база даних. Будьте дуже уважні при роботі з цією командою!
- ***Зв'язок*** – це асоціація між двома сутностями.
- ***Ключ*** – це елемент даних, за допомогою якого окремі екземпляри деякого типу сутності унікально ідентифікуються.
- ***Потенційний ключ*** – атрибут або набір атрибутів, за допомогою якого окремі екземпляри деякого типу сутності унікально ідентифікуються.
- ***Складовий ключ*** – потенційний ключ, який складається з двох або більше атрибутів.
- ***Первинний ключ (Primary Key)*** – це атрибут або група атрибутів, які однозначно ідентифікують екземпляр сутності. На діаграмі первинні ключі розміщуються вище горизонтальної лінії. Ключ може бути складним, тобто складатись з декількох атрибутів.
- ***Зовнішні ключі (Foreign Key)*** створюються автоматично, коли сутності з'єднуються зв'язком (міграція ключа). Зв'язки між таблицями реляційної БД ідентифікуються однаковими ключами в таблицях (зовнішніми ключами).
- ***Діаграми «сутність-зв'язок» (Entity-Relationship)*** призначені для розробки моделей даних і забезпечують стандартний спосіб визначення даних та відношень між ними.

Практичні приклади

```
-------------------------------------------------------------------------
                                    /* Створення Бази Даних*/
-------------------------------------------------------------------------

CREATE DATABASE ShopDB
ON
(
        NAME = 'ShopDB',
        FILENAME = 'D:\ShopDB.mdf',
        SIZE = 10MB,
        MAXSIZE = 100MB,
        FILEGROWTH = 10MB
)
LOG ON
```

```
(
	NAME = 'LogShopDB',
	FILENAME = 'D:\LogShopDB.ldf',
	SIZE = 5MB,
	MAXSIZE = 50MB,
	FILEGROWTH = 5MB
)
COLLATE Cyrillic_General_CI_AS -- Задаємо кодування для бази даних за замовчуванням

-------------------------------------------------------------------------
                                    /* Видалення Бази Даних*/
-------------------------------------------------------------------------

-- Перед видаленням певної бази даних, рекомендується
-- переключитись на базу master.
USE master;
GO

-- Оператор DROP DATABASE видаляє Базу Даних
DROP DATABASE  ShopDB;
GO

-------------------------------------------------------------------------
                                    /* Зміна Бази Даних*/
-------------------------------------------------------------------------

EXEC sp_helpdb ShopDB

ALTER DATABASE ShopDB   -- Оператор ALTER DATABASE змінює БД
MODIFY FILE                     -- Зміна розміру БД
( NAME = 'ShopDB',
  SIZE = 100MB )
GO

 ALTER DATABASE ShopDB-- Оператор ALTER DATABASE змінює БД
 MODIFY FILE
( NAME = 'ShopDB',
  MAXSIZE = 1000MB,                 -- Зміна максимального розміру БД
  FILEGROWTH = 40% )       -- та параметра збільшення розміру БД
  -- Якщо FILEGROWTH задається у відсотках, то відсоток рахується від поточного розміру БД
(FILEGROWTH = 0,4*SIZE).
GO

ALTER DATABASE ShopDB
 MODIFY FILE
(
  NAME = 'ShopDB',
  MAXSIZE = 10MB               -- Змінений максимальний розмір не може бути менший за поточний
розмір БД
)
GO

ALTER DATABASE ShopDB
MODIFY FILE
( NAME = 'ShopDB',
  SIZE = 100MB )               -- Не дозволяється задавати розмір БД, який менший або рівний
поточному.
GO

EXEC sp_helpdb ShopDB

-------------------------------------------------------------------------
USE ShopDB
GO

-------------------------------------------------------------------------
                                        /* Створення таблиці */
-------------------------------------------------------------------------

CREATE TABLE Customers
(	-- Ключове слово IDENTITY створює стовпець ідентифікації.(seed = 1, step = 2)
	CustomerNo int IDENTITY(1, 2) NOT NULL,
	CustomerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	Address2 varchar(25) DEFAULT 'Unknown', -- Ключове слово DEFAULT присвоює значення за
замовчуванням.
	City varchar(15) NOT NULL,
```

```
        [State] char(2) NOT NULL,
        Zip varchar(10) NOT NULL,
        Contact varchar(25) NOT NULL,
        Phone char(10) NOT NULL,
        FedIDNo  varchar(10) NOT NULL,
        DateInSystem smalldatetime NOT NULL
);
GO

DROP TABLE Customers; -- DROP TABLE - оператор видалення таблиці
GO

CREATE TABLE Customers
(       -- Ключове слово IDENTITY створює стовпець ідентифікації.(seed = 1, step = 2)
        CustomerNo int IDENTITY(1, 2) NOT NULL,
        CustomerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        Address2 varchar(25) DEFAULT 'Unknown', -- Ключове слово DEFAULT присвоює значення за
замовчуванням.
        City varchar(15) NOT NULL,
        [State] char(2) NOT NULL,
        Zip varchar(10) NOT NULL,
        Contact varchar(25) NOT NULL,
        Phone char(10) NOT NULL,
        FedIDNo  varchar(10) NOT NULL,
        DateInSystem smalldatetime NOT NULL
)
WITH (DATA_COMPRESSION = ROW); -- таблиця із стисненням по рядках (Не працює в SQL Express)
GO

INSERT INTO Customers
(CustomerName, Address1, City, [State], Zip, Contact, Phone, FedIDNo, DateInSystem)
VALUES
('Jeremy', 'new str', 'NewCity', 'ds', 'newZIP', 'newContact', '0567878932', 'NewIDNO',
GETDATE());
GO

SELECT * FROM Customers;
GO

------------------------------------------------------------------------
                                                /* Зміна таблиці */
------------------------------------------------------------------------

-- Змінюємо таблицю Customers,
-- додавши стовпець NewFild

ALTER TABLE Customers -- Оператор ALTER TABLE змінює таблицю
ADD                   -- ADD - додає
NewFild int NULL      -- стовпець.
GO


SELECT * FROM Customers;
GO

-- Змінюємо таблицю Customers,
-- видаливши стовпець NewFild

ALTER TABLE Customers -- Оператор ALTER TABLE змінює таблицю
DROP COLUMN           -- DROP COLUMN - видаляє
NewFild               -- стовпець.
GO

SELECT * FROM Customers;
GO

ALTER TABLE Customers
ADD
NewFild2 varchar(10) NOT NULL
-- Неможливо додати колонку з обмеженням NOT NULL, не встановивши значення за замовчуванням.
GO

ALTER TABLE Customers
ADD
NewFild2 varchar(10) NOT NULL
DEFAULT 'Unknown'
```

```
GO

SELECT * FROM Customers
GO

-------------------------------------------------------------------------

USE ShopDB
GO

DROP TABLE Customers;
GO
-------------------------------------------------------------------------
--                  Створення первинного ключа
-------------------------------------------------------------------------
-- PRIMARY KEY (первинний ключ, обмеження первинного ключа)
-- Значення комірок стовпця(-ів) з обмеженням PRIMARY KEY
-- однозначно визначає (-ють) кожен рядок в таблиці.

CREATE TABLE Customers
(
       CustomerNO int NOT NULL PRIMARY KEY, -- На стовпці CustomerNO  задаємо первинний ключ
       CustomerName varchar(25) NOT NULL,
       Address1 varchar(25) NOT NULL,
       City     varchar(15) NOT NULL,
       Contact  varchar(25) NOT NULL,
       Phone char(12)
)
GO

INSERT INTO Customers

(CustomerNO, CustomerName, Address1, City, Contact, Phone)
VALUES
(1,'John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
(2,'Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');
GO

-- Спроба порушити унікальність.
INSERT INTO Customers

(CustomerNO, CustomerName, Address1, City, Contact, Phone)
VALUES
(1,'Aline Stewart', '5 Tarbot Rd', 'Aberdeen', 'AS@gmail.com', '0141-8481825'); -- Помилка!

SELECT * FROM Customers;

DROP TABLE Customers;
GO

-- Якщо первинний ключ складається з двох та більше стовпців,
-- його називають складовим первинним ключем.
-- Спроба вставити другий Первинний ключ в таблицю призводить до помилки.
-- Первинний ключ в таблиці може бути лише один (простий або складовий)
CREATE TABLE Customers
(
       CustomerNo int NOT NULL,
       CustomerName varchar(25) NOT NULL,
       Address1 varchar(25) NOT NULL,
       City      varchar(15) NOT NULL,
       Contact varchar(25) NOT NULL,
       Phone char(12) NOT NULL,
       PRIMARY KEY (CustomerNo, CustomerName) -- задаємо складовий первинний ключ на стовпцях
CustomerNo, CustomerName
)
GO

INSERT INTO Customers

(CustomerNO, CustomerName, Address1, City, Contact, Phone)
VALUES
(1,'John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
(2,'Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');
GO

INSERT INTO Customers
```

```
(CustomerNO, CustomerName, Address1, City, Contact, Phone)
VALUES
(1,'Aline Stewart', '5 Tarbot Rd', 'Aberdeen', 'AS@gmail.com', '0141-8481825');

INSERT INTO Customers

(CustomerNO, CustomerName, Address1, City, Contact, Phone)
VALUES
(2,'Mike Ritchie', '18 Tain St', 'Aberdeen', 'Mike@i.com', '0171-7777777'); -- Помилка!

SELECT * FROM Customers;

-- Створення обмеження первинного ключа на існуючій таблиці
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
        CustomerNO int IDENTITY NOT NULL,
        CustomerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        City     varchar(15) NOT NULL,
        Contact  varchar(25) NOT NULL,
        Phone char(12)
)
GO

-- Змінюємо таблицю   Customers, задавши обмеження первинного ключа на стовпці   CustomerNo
ALTER TABLE Customers
ADD CONSTRAINT PK_Customers
PRIMARY KEY (CustomerNo)

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
('Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');
GO

SELECT * FROM Customers;

DROP TABLE Customers;
GO

CREATE TABLE Customers
(
        CustomerNO int NOT NULL,
        CustomerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        City     varchar(15) NOT NULL,
        Contact  varchar(25) NOT NULL,
        Phone char(12)
)
GO

ALTER TABLE Customers
ADD  -- Ім'я обмеження не є обов'язковим.
PRIMARY KEY (CustomerNo,Address1)

INSERT INTO Customers

(CustomerNO, CustomerName, Address1, City, Contact, Phone)
VALUES
(1,'John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
(2,'Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');
GO

INSERT INTO Customers

(CustomerNO, CustomerName, Address1, City, Contact, Phone)
VALUES
(1,'Aline Stewart', '5 Tarbot Rd', 'Aberdeen', 'AS@gmail.com', '0141-8481825');

INSERT INTO Customers

(CustomerNO, CustomerName, Address1, City, Contact, Phone)
```

```
VALUES
(2,'Mike Ritchie', '18 Tain St', 'Aberdeen', 'Mike@i.com', '0171-7777777');
SELECT * FROM Customers;

-------------------------------------------------------------------------

USE ShopDB
GO

DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustomerNO int IDENTITY NOT NULL
		PRIMARY KEY,
	CustomerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	City     varchar(15) NOT NULL,
	Contact  varchar(25) NOT NULL,
	Phone char(12)
)
GO
-------------------------------------------------------------------------
--                  Створення зовнішнього ключа
-------------------------------------------------------------------------
-- FOREIGN KEY (зовнішній ключ, обмеження зовнішнього ключа)
DROP TABLE Orders;
GO

-- Значення комірок стовпця дочірньої таблиці з обмеженням FOREIGN KEY
-- повинно співпадати з одним з існуючих значень в батьківській таблиці.

CREATE TABLE Orders
(
	OrderID int IDENTITY NOT NULL
		PRIMARY KEY,
	CustomerNo int NOT NULL                                                    -- Стовпець
дочірньої таблиці, для якої заданий FOREIGN KEY
	FOREIGN KEY REFERENCES Customers(CustomerNo), -- Батьківська таблиця Customers та її
стовпець CustomerNo.
	OrderDate date NOT NULL,
	Goods varchar(30) NOT NULL
)
GO

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
('Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');

INSERT INTO Orders
(CustomerNo, OrderDate,       Goods)
VALUES
(1, GETDATE(), 'KeyBoard'),
(2, GETDATE(), 'Mouse'),
(2, GETDATE(), 'WebCam'),
(1, GETDATE(), 'Mouse');
SELECT CustomerNO, CustomerName, Address1, City FROM Customers;
SELECT * FROM Orders;

-- Не допускається запис в стовпець-посилання (стовпець з FOREIGN KEY) дочірньої таблиці
-- значень неіснуючих в стовпці-посиланні (стовпець з PRIMARY KEY) батьківської таблиці.

INSERT INTO Orders
(CustomerNo, OrderDate,       Goods)
VALUES
(3, GETDATE(), 'Mouse'); -- Помилка!

DROP TABLE Orders;
GO

-- Створення обмеження зовнішнього ключа на існуючій таблиці
CREATE TABLE Orders
(
```

```
        OrderID int IDENTITY NOT NULL
                PRIMARY KEY,
        CustomerNo int NOT NULL,
        OrderDate date NOT NULL,
        Goods varchar(30) NOT NULL
)

ALTER TABLE Orders
ADD CONSTRAINT FK_CustomersCustomerNo
FOREIGN KEY (CustomerNo) REFERENCES Customers(CustomerNo);

DROP TABLE Customers -- Неможливо видалити таблицю, на яку посилається дочірня таблиця.

-- Першими видаляються дочірні таблиці, після чого видаляються батьківські таблиці.

DROP TABLE Orders;
DROP TABLE Customers;


USE ShopDB
GO

CREATE TABLE Customers
(
        CustomerNO int IDENTITY NOT NULL
                PRIMARY KEY,
        CustomerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        City     varchar(15) NOT NULL,
        Contact  varchar(25) NOT NULL,
        Phone char(12)
)
GO

CREATE TABLE Orders
(
        OrderID int IDENTITY NOT NULL
                PRIMARY KEY,
        CustomerNo int NOT NULL,
        OrderDate date NOT NULL,
        Goods varchar(30) NOT NULL
)
GO

ALTER TABLE Orders
ADD CONSTRAINT FK_CustomersCustomerNo
FOREIGN KEY (CustomerNo) REFERENCES Customers(CustomerNo);

GO

-- Видалення обмеження зовнішнього. FK_CustomersCustomerNo - ім'я за замовчуванням
ALTER TABLE Orders
DROP CONSTRAINT FK_CustomersCustomerNo;
GO
DROP TABLE Customers; -- Таблиця Customers видалена.
GO

-------------------------------------------------------------------------
USE ShopDB
GO

DROP TABLE Customers;

DROP TABLE Orders;

CREATE TABLE Customers
(
        CustomerNO int IDENTITY NOT NULL
                PRIMARY KEY,
        CustomerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        City     varchar(15) NOT NULL,
        Contact  varchar(25) NOT NULL,
        Phone char(12)
)
GO
CREATE TABLE Orders
```

```
(
	OrderID int IDENTITY NOT NULL
		PRIMARY KEY,
	CustomerNo int NULL DEFAULT 2 ,
	OrderDate date NOT NULL,
	EmployeeId int NULL
)
GO

ALTER TABLE Orders
ADD CONSTRAINT FK_CustomersCustomerNo
FOREIGN KEY (CustomerNo) REFERENCES Customers(CustomerNo);

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
('Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');

INSERT INTO Orders
(CustomerNo, OrderDate,	EmployeeId)
VALUES
(1, GETDATE(), 1);

SELECT * FROM Customers;
SELECT * FROM Orders;

-- Неможливо видалити запис з батьківської таблиці, оскільки на неї є посилання з дочірньої
таблиці.
DELETE Customers
WHERE CustomerName = 'John Kay';
-------------------------------------------------------------------------
--		Обмеження цілісності посилання
-------------------------------------------------------------------------
-- За допомогою обмеження цілісності посилання (ON DELETE, ON UPDATE)
-- можна визначити дії, які SQL Server буде робити
-- над рядками дочірньої таблиці, коли користувач спробує
-- видалити або оновити ключовий стовпець батьківської таблиці.

-- Дії:

-- 1. CASCADE -	при UPDATE, DELETE в батьківській таблиці ключового значення,
--			в дочірній таблиці також виконується  UPDATE на нове значення,
--			або DELETE	рядків з дочірньої таблиці відповідно
ALTER TABLE Orders
DROP CONSTRAINT FK_CustomersCustomerNo;
GO

ALTER TABLE Orders
ADD CONSTRAINT FK_CustomersCustomerNo
FOREIGN KEY (CustomerNo) REFERENCES Customers(CustomerNo)
	ON DELETE CASCADE
GO

DELETE Customers
WHERE CustomerName = 'John Kay';
SELECT * FROM Customers;
SELECT * FROM Orders;

DBCC CHECKIDENT("Customers",RESEED,0); -- Оновлення значення IDENTITY = 0
GO

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632')

DBCC CHECKIDENT("Customers",RESEED,2) -- Оновлення значення IDENTITY = 2
GO
DBCC CHECKIDENT("Orders",RESEED,0); -- Оновлення значення IDENTITY = 0
GO

INSERT INTO Orders
(CustomerNo, OrderDate,	EmployeeId)
VALUES (1,GETDATE(),1);
```

```
GO

SELECT * FROM Customers;
SELECT * FROM Orders;

-- 2. SET NULL - при UPDATE, DELETE в батьківській таблиці ключового значення,
--                           в дочірній таблиці всі значення, які складають ці зовнішні ключі,
--                           будуть змінені на NULL

ALTER TABLE Orders
DROP CONSTRAINT FK_CustomersCustomerNo;
GO

ALTER TABLE Orders
ADD CONSTRAINT FK_CustomersCustomerNo
FOREIGN KEY (CustomerNo) REFERENCES Customers(CustomerNo)
       ON DELETE SET NULL
GO

DELETE Customers
WHERE CustomerName = 'John Kay';
GO

SELECT * FROM Customers;
SELECT * FROM Orders;

DBCC CHECKIDENT("Customers",RESEED,0); -- Оновлення значення IDENTITY = 0
GO

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632')

DBCC CHECKIDENT("Customers",RESEED,2) -- Оновлення значення IDENTITY = 2
GO

DELETE Orders WHERE OrderID = 1;
GO

DBCC CHECKIDENT("Orders",RESEED,0); -- Оновлення значення IDENTITY = 0
GO

INSERT INTO Orders
(CustomerNo, OrderDate,       EmployeeId)
VALUES (1,GETDATE(),1);
GO

SELECT * FROM Customers;
SELECT * FROM Orders;

-- 3. SET DEFAULT - при UPDATE, DELETE в батьківській таблиці ключового значення,
--                                  в дочірній таблиці всі значення, які складають ці зовнішні
ключі,
--                                  будуть змінені на значення за замовчуванням.
ALTER TABLE Orders
DROP CONSTRAINT FK_CustomersCustomerNo;
GO
ALTER TABLE Orders
ADD CONSTRAINT FK_CustomersCustomerNo
FOREIGN KEY (CustomerNo) REFERENCES Customers(CustomerNo)
       ON DELETE SET DEFAULT
GO

DELETE Customers
WHERE CustomerName = 'John Kay';
GO

SELECT * FROM Customers;
SELECT * FROM Orders;

DBCC CHECKIDENT("Customers",RESEED,0); -- Оновлення значення IDENTITY = 0
GO

INSERT INTO Customers
```

```
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632')

DBCC CHECKIDENT("Customers",RESEED,2) -- Оновлення значення IDENTITY = 2
GO

DELETE Orders WHERE OrderID = 1;
GO

DBCC CHECKIDENT("Orders",RESEED,0); -- Оновлення значення IDENTITY = 0
GO

INSERT INTO Orders
(CustomerNo, OrderDate,       EmployeeId)
VALUES (1,GETDATE(),1);
GO

SELECT * FROM Customers;
SELECT * FROM Orders;

-- 4. NO ACTION (за замовчуванням) - забороняє виконання UPDATE, DELETE!
ALTER TABLE Orders
DROP CONSTRAINT FK_CustomersCustomerNo;
GO

ALTER TABLE Orders
ADD CONSTRAINT FK_CustomersCustomerNo
FOREIGN KEY (CustomerNo) REFERENCES Customers(CustomerNo)
       ON DELETE NO ACTION
GO

DELETE Customers
WHERE CustomerName = 'John Kay';

SELECT * FROM Customers;
SELECT * FROM Orders;

-------------------------------------------------------------------------

USE ShopDB
GO

-------------------------------------------------------------------------
                                    /*Обмеження користувача*/
-------------------------------------------------------------------------

DROP TABLE Orders;
GO
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
       CustomerNO int IDENTITY NOT NULL,
       CustomerName varchar(25) NOT NULL,
       Address1 varchar(25) NOT NULL,
       City     varchar(15) NOT NULL,
       Contact  varchar(25) NOT NULL,
       -- Обмеження користувача
       Phone char(12) CHECK (Phone LIKE '[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-
9]')
)
GO

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632');

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-*******'); -- Помилка.

SELECT * FROM Customers;
```

```
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustomerNO int IDENTITY NOT NULL,
	CustomerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	City	varchar(15) NOT NULL,
	Contact  varchar(25) NOT NULL,
	Phone char(12) UNIQUE -- UNIQUE (альтернативний ключ), значення в комірці має бути
унікальним
)
GO

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-*******');

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632');

SELECT * FROM Customers;

DROP TABLE Customers;
GO

-- Можливо задати одразу декілька обмежень в довільному порядку
CREATE TABLE Customers
(
	CustomerNO int IDENTITY NOT NULL,
	CustomerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	City	varchar(15) NOT NULL,
	Contact  varchar(25) NOT NULL,
	Phone char(12) UNIQUE CHECK (Phone LIKE '[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-
9][0-9]')
)
GO


INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632');

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-*******'); -- Помилка (Не відповідає
шаблону).

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'); -- Помилка (Номер
повторюється).
SELECT * FROM Customers;

DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustomerNO int IDENTITY NOT NULL,
	CustomerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	City	varchar(15) NOT NULL,
	Contact  varchar(25) NOT NULL,
	Phone char(12)
)
GO
```

```
-- Створення обмеження користувача на існуючій таблиці
ALTER TABLE Customers
ADD CONSTRAINT CN_CustomersPhoneNo
-- Обмеження CHECK
CHECK (Phone LIKE '[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
GO

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632');

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-*******'); -- Помилка (Не відповідає
шаблону).

-------------------------------------------------------------------------

ALTER TABLE Customers
NOCHECK CONSTRAINT CN_CustomersPhoneNo;             -- Відключення роботи обмеження CHECK

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-*******');

-------------------------------------------------------------------------

ALTER TABLE Customers
CHECK CONSTRAINT CN_CustomersPhoneNo;               --  Включення роботи обмеження CHECK

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-*******'); -- Помилка (Не відповідає
шаблону).

-------------------------------------------------------------------------
ALTER TABLE Customers
DROP CONSTRAINT       CN_CustomersPhoneNo;          --  Видалення обмеження CHECK

INSERT INTO Customers
(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-*******');

--  Після видалення не можливе повернення обмеження CHECK
ALTER TABLE Customers
CHECK CONSTRAINT CN_CustomersPhoneNo;
-------------------------------------------------------------------------
--                 Зв'язок Один до Одного
-------------------------------------------------------------------------
USE ShopDB
GO

DROP TABLE Orders;
GO
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
        CustomerNO int IDENTITY NOT NULL
                PRIMARY KEY,
        CustomerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        City     varchar(15) NOT NULL,
        Contact  varchar(25) NOT NULL,
        Phone char(12)
)
GO

CREATE TABLE Orders
(
        OrderID int IDENTITY NOT NULL
```

```
                PRIMARY KEY,
        CustomerNo int NOT NULL       UNIQUE
        FOREIGN KEY REFERENCES Customers(CustomerNo), -- Зв'язок Один до Одного.
        OrderDate date NOT NULL,
        Goods varchar(30) NOT NULL
)
GO

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
('Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');

INSERT INTO Orders
(CustomerNo, OrderDate,       Goods)
VALUES
(1, GETDATE(), 'KeyBoard'),
(2, GETDATE(), 'Mouse');

SELECT CustomerNO, CustomerName, Address1, City FROM Customers;
SELECT * FROM Orders;

INSERT INTO Orders
(CustomerNo, OrderDate,       Goods)
VALUES
(2, GETDATE(), 'WebCam'),
(1, GETDATE(), 'Mouse');  -- Помилка
-------------------------------------------------------------------------
--                 Зв'язок Один до Багатьох
-------------------------------------------------------------------------
DROP TABLE Orders;
GO
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
        CustomerNO int IDENTITY NOT NULL
                PRIMARY KEY,
        CustomerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        City     varchar(15) NOT NULL,
        Contact  varchar(25) NOT NULL,
        Phone char(12)
)
GO
CREATE TABLE Orders
(
        OrderID int IDENTITY NOT NULL
                PRIMARY KEY,
        CustomerNo int NOT NULL
        FOREIGN KEY REFERENCES Customers(CustomerNo), -- Зв'язок Один до Багатьох.
        OrderDate date NOT NULL,
        Goods varchar(30) NOT NULL
)
GO

INSERT INTO Customers

(CustomerName, Address1, City, Contact, Phone)
VALUES
('John Kay', '56 Hight St', 'London', 'JK@gmail.com', '0171-7745632'),
('Mike Ritchie', '64 Fern Rd', 'Glasgow', 'MR@gmail.com', '01475-392178');

INSERT INTO Orders
(CustomerNo, OrderDate,       Goods)
VALUES
(1, GETDATE(), 'KeyBoard'),
(2, GETDATE(), 'Mouse'),
(2, GETDATE(), 'WebCam'),
(1, GETDATE(), 'Mouse');
SELECT CustomerNO, CustomerName, Address1, City FROM Customers;
SELECT * FROM Orders;

-------------------------------------------------------------------------
```

```
--                Зв'язок Багато ко Багатьох
-------------------------------------------------------------------------
DROP TABLE Students;
DROP TABLE StudentsCourses;
DROP TABLE Courses;

CREATE TABLE Students
(
	StudentId int IDENTITY NOT NULL
		PRIMARY KEY,
	FName varchar(50) NOT NULL,
	LName varchar(50) NOT NULL,
	Email varchar(50) NOT NULL,
	Phone varchar(50) NOT NULL
)

CREATE TABLE Courses
(
	CourseId int IDENTITY NOT NULL
		PRIMARY KEY,
	CourseName varchar(50) NOT NULL,
	Price money Not Null
)

CREATE TABLE StudentsCourses
(
	 StudentId int NOT NULL
		FOREIGN KEY REFERENCES Students(StudentId),
	 CourseId int NOT NULL
		FOREIGN KEY REFERENCES Courses(CourseId),
	 PRIMARY KEY(StudentId, CourseId)
)

INSERT INTO Students

(FName, LName, Email, Phone)
VALUES
('John', 'Kay', 'JK@gmail.com', '0171-7745632'),
('Aline', 'Stewart', 'AS@gmail.com', '0141-8481825'),
('Mike', 'Ritchie', 'MR@gmail.com', '0147-3921787');

INSERT INTO Courses
(CourseName, Price)
VALUES
('C#', 400.00),
('Java ', 300.00),
('Python', 350.00),
('Scala', 450.00);


INSERT INTO StudentsCourses
(StudentId, CourseId)
VALUES
(1,1),
(2,1),
(3,3),
(3,1),
(2,2);

SELECT * FROM Students
SELECT * FROM StudentsCourses
SELECT * FROM Courses
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Створити базу даних максимальної розмірності 100 МБ, за умови, що буде використовуватись близько 30 МБ. Введіть всі необхідні налаштування. Журнал транзакцій повинен бути розміщений на іншому фізичному носії (якщо такий є).

Завдання 3

Спроєктувати БД, яка відображає інформацію про угоди, що укладені менеджерами з продажу товару. Поля таблиць продумати самостійно.

Завдання 4

Створити БД ShopDB, що складається з таблиць Employees, Customers та Orders. Задати зв'язки між таблицями. Видалити таблиці Employees, Customers та Orders.

Завдання 5

Зайдіть на сайт MSDN.

Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп'ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Бібліотека».
2. Встановити зв'язки між таблицями.

Варіант 2

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Університет».
2. Встановити зв'язки між таблицями.

Варіант 3

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Виробництво».
2. Встановити зв'язки між таблицями.

Варіант 4

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Автомайстерня».
2. Встановити зв'язки між таблицями.

Варіант 5

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Сесія».
2. Встановити зв'язки між таблицями.

Варіант 6

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Керування проєктом».
2. Встановити зв'язки між таблицями.

Варіант 7

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Поліклініка».
2. Встановити зв'язки між таблицями.

Варіант 8

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Телефонізація».
2. Встановити зв'язки між таблицями.

Варіант 9

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Спорт».
2. Встановити зв'язки між таблицями.

Варіант 10

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Сільскогосподарські постачання».
2. Встановити зв'язки між таблицями.

Варіант 11

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Міський транспорт».
2. Встановити зв'язки між таблицями.

Варіант 12

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Географія».
2. Встановити зв'язки між таблицями.

Варіант 13

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Управління будинком».
2. Встановити зв'язки між таблицями.

Варіант 14

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Аеропорт».
2. Встановити зв'язки між таблицями.

Варіант 15

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Ринок ПК».
2. Встановити зв'язки між таблицями.

Варіант 16

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Деканат».
2. Встановити зв'язки між таблицями.

Варіант 17

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Зоопарк».
2. Встановити зв'язки між таблицями.

Варіант 18

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Шахи».
2. Встановити зв'язки між таблицями.

Варіант 19

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Судноплавство».
2. Встановити зв'язки між таблицями.

Варіант 20

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Автотранспортне підприємство».
2. Встановити зв'язки між таблицями.

Варіант 21

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Наукові конференції».
2. Встановити зв'язки між таблицями.

Варіант 22

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Програмні продукти».
2. Встановити зв'язки між таблицями.

Варіант 23

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Театр».
2. Встановити зв'язки між таблицями.

Варіант 24

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Довідникова аптек».
2. Встановити зв'язки між таблицями.

Варіант 25

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Ресторан».
2. Встановити зв'язки між таблицями.

Варіант 26

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Формула-1».
2. Встановити зв'язки між таблицями.

Варіант 27

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Тюнінг автомобілей».
2. Встановити зв'язки між таблицями.

Варіант 28

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Археологія».
2. Встановити зв'язки між таблицями.

Варіант 29

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Тур-фірма».
2. Встановити зв'язки між таблицями.

Варіант 30

Цілісність даних:

1. Забезпечити доменну цілісність, цілісність сутностей та цілісність посилань в БД, що розробляється за напрямком «Телемовлення».
2. Встановити зв'язки між таблицями.

Рекомендовані ресурси

MSDN: Primary key
https://msdn.microsoft.com/ru-ru/library/ms191236.aspx

MSDN: Foreign key
https://msdn.microsoft.com/ru-ru/library/ms175464.aspx

MSDN: Tables
https://docs.microsoft.com/ru-ru/previous-versions/sql/sql-server-2005/ms175010(v=sql.90)

MSDN: DDL
https://msdn.microsoft.com/ru-ru/library/ff848799.aspx

Контрольні питання

1. Що таке зв'язок?
2. Що таке зовнішній ключ?
3. Що таке первинний ключ?
4. Які типи зв'язків використовують в базах даних?
5. За допомогою якої діаграми надається модель даних?

Комп'ютерний практикум 4
Проєктування бази даних

Мета роботи: знайомство з принципами проєктування баз даних та поняттями нормалізації і денормалізації.

Теоретичні відомості

- ***Нормалізація*** – це метод створення набору відношень з заданими властивостями на базі вимог до даних, які встановлюються для конкретної предметної області.
- ***Функціональна залежність*** описує зв'язок між атрибутами і є одним з основних понять нормалізації.
- ***Детермінант функціональної залежності*** – це атрибут або група атрибутів, що розташованім на діаграмі функціональної залежності ліворуч від символу стрілки (A→B).
- ***Нормалізація*** – це формальний метод аналізу відношень на базі їх первинного ключа (або потенційних ключів) та існуючих функціональних залежностей.
- ***Нормалізація*** – це метод створення набору відношень з заданими властивостями на базі вимог до даних, які встановлюються для конкретної предметної області.
- ***Нормалізація таблиць*** – це формальний апарат обмежень на формування таблиць, який дозволяє усунути дублювання даних, забезпечує несуперечливість даних, що зберігаються в базі, зменшує трудовитрати на ведення бази даних (введення та коригування даних).
- ***Ненормалізована форма*** – це таблиця, яка містить одну або декілька груп даних, які повторюються.
- ***Перша нормальна форма (1NF)*** – це відсутність даних, що повторюються; будь-яке поле будь-якого запису зберігає лише одне значення.
- ***Повна функціональна залежність*** – це функціональна залежність A→B, в якій видалення будь-якого атрибуту з A призводить до втрати цієї залежності.
- ***Часткова функціональна залежність*** – це функціональна залежність A→B, яка в A містить деякій атрибут, при видаленні якого ця залежність зберігається.
- ***Друга нормальна форма (2NF)*** – задовольняє першій нормальній формі; кожен стовпець повинен залежати від всього ключа.

- ***Транзитивна залежність*** – опис такого типу функціональної залежності, яка виникає при наявності наступних функціональних залежностей між атрибутами A, B, C: A→B і B→C.
- ***Третя нормальна форма (3NF)*** – задовольняє другій нормальній формі; в жодному неключовому стовпці не може бути залежності від іншого неключового стовпця; наявність в таблиці похідних даних не допускається.
- ***Денормалізація таблиць*** – це процес зниження нормальної форми. Здійснюється якщо приведена вища форма призводить до погіршення практичного використання.
- ***Нормальна форма Бойса-Кодда (NFBC)*** – відношення, в якому кожен детермінант є потенційним ключем.
- ***Багатозначна залежність*** – це залежність між атрибутами A, B та C деякого відношення, при якій для кожного значення атрибута A існують відповідні набори значень атрибутів B та C, причому обидва цих набори не залежать один від одного.
- ***Четверта нормальна форма (4NF)*** – відношення, яке знаходиться в нормальній формі Бойса-Кодда та не містить нетривіальних багатозначних залежностей.
- ***Залежність з'єднання*** – це така ситуація, при якій декомпозиція відношення може супроводжуватися генерацією помилкових рядків при зворотному з'єднанні декомпозованих відношень за допомогою операції природного з'єднання.
- ***П'ята нормальна форма (5NF)*** – відношення, яке не містить залежностей з'єднання.

Практичні приклади

```
USE master
DROP DATABASE ShopDB;
CREATE DATABASE ShopDB;
USE ShopDB
CREATE TABLE Orders
(
	Customer varchar(200),
	OrderDate date,
	OrderDetails varchar(100),
	Employee varchar(150),
)

INSERT Orders
VALUES
('John Riley Kay; London, 56 Hight St; 0171-7745632;',
 '2019-12-12',
 'JN754 Jeans, 35€, 7, 245€; BL58 Belt, 20€, 6, 120€; SH14 Shoes, 55€, 5, 275€;',
 'Mike Oliver Ritchie'),

 ('Tony Daniel Smith; Glasgow, 64 Fern Rd; 0147-3921789;',
 '2019-05-17',
 'HT23 Hat, 28€, 5, 140€; TS777 T-shirt, 30€, 10, 300€;',
 'Aline Agnes Stewart'),

 ('John Robert Walker; Aberdeen, 5 Tarbot Rd; 0141-8481825;',
 '2019-05-19',
 'SH14 Shoes, 55€, 10, 550€; HT23 Hat, 28€, 10, 280€;',
```

```
 'Aline Agnes Stewart')

SELECT * FROM Orders
 ------------------------------------------------------------------------
--                                                                 Перша НФ
------------------------------------------------------------------------
-- Перша нормальна форма (1NF) - вимагає відсутність в таблиці рядків,
-- які повторюються, а також розбиття складних значень атомарних даних
-- в комірці на простіші атомарні  дані.
DROP TABLE Orders
CREATE TABLE Orders
(
        CustFName varchar(15) NOT NULL,
        CustMName varchar(15) NOT NULL,
        CustLName varchar(15) NOT NULL,
        CustomerCity varchar(15),
        CustomerAddress varchar(25),
        Phone varchar(12) NOT NULL,
        OrderDate date NOT NULL,
        ProductID char(5) NOT NULL,
        ProductDescription varchar(15),
        UnitPrice smallmoney NOT NULL,
        Qty int NOT NULL,
        TotalPrice money,
        EmpFName varchar(15) NOT NULL,
        EmpMName varchar(15) NOT NULL,
        EmpLName varchar(15) NOT NULL,
        CHECK (Phone LIKE '[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
)

INSERT Orders
VALUES
('John', 'Riley', 'Kay', 'London', '56 Hight St', '0171-7745632',
 '2019-12-12',
 'JN754', 'Jeans', 35, 7, 245,
 'Mike', 'Oliver', 'Ritchie'),

('John', 'Riley', 'Kay', 'London', '56 Hight St', '0171-7745632',
 '2019-12-12',
 'BL58', 'Belt', 20, 6, 120,
 'Mike', 'Oliver', 'Ritchie'),

 ('John', 'Riley', 'Kay', 'London', '56 Hight St', '0171-7745632',
 '2019-12-12',
 'SH14', 'Shoes', 55, 5, 275,
 'Mike', 'Oliver', 'Ritchie'),

 ('Tony', 'Daniel', 'Smith', 'Glasgow', '64 Fern Rd', '0147-3921789',
 '2019-05-17',
 'HT23', 'Hat', 28, 5, 140,
 'Aline', 'Agnes', 'Stewart'),

 ('Tony', 'Daniel', 'Smith', 'Glasgow', '64 Fern Rd', '0147-3921789',
 '2019-05-17',
 'TS777', 'T-shirt', 30, 10, 300,
 'Aline', 'Agnes', 'Stewart'),

 ('John', 'Robert', 'Walker', 'Aberdeen', '5 Tarbot Rd', '0141-8481825',
 '2019-05-19',
 'SH14', 'Shoes', 55, 10, 550,
 'Aline', 'Agnes', 'Stewart'),
 ('John', 'Robert', 'Walker', 'Aberdeen', '5 Tarbot Rd', '0141-8481825',
 '2019-05-19',
 'HT23', 'Hat', 28, 10, 280,
 'Aline', 'Agnes', 'Stewart')
SELECT * FROM Orders
-- Перед тим, як переходити до розгляду другої та третьої нормальних форм, необхідно забезпечити
-- сутнісну цілісність для таблиці Orders(визначити первинний ключ)
DROP TABLE Orders
CREATE TABLE Orders
(
        OrderID int NOT NULL,
        LineItem int NOT NULL,
        OrderDate date NOT NULL,
        CustFName varchar(15) NOT NULL,
        CustMName varchar(15) NOT NULL,
        CustLName varchar(15) NOT NULL,
```

```
        CustomerCity varchar(15),
        CustomerAddress varchar(25),
        Phone varchar(12) NOT NULL,
        ProductID char(5) NOT NULL,
        ProductDescription varchar(15),
        UnitPrice smallmoney NOT NULL,
        Qty int NOT NULL,
        TotalPrice money,
        EmpFName varchar(15) NOT NULL,
        EmpMName varchar(15) NOT NULL,
        EmpLName varchar(15) NOT NULL,

        CHECK (Phone LIKE '[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
        PRIMARY KEY (OrderId, LineItem)
)

INSERT Orders
VALUES

(1, 1, '2019-12-12',
 'John', 'Riley', 'Kay', 'London', '56 Hight St', '0171-7745632',
 'JN754', 'Jeans', 35, 7, 245,
 'Mike', 'Oliver', 'Ritchie'),

(1, 2, '2019-12-12',
 'John', 'Riley', 'Kay', 'London', '56 Hight St', '0171-7745632',
  'BL58', 'Belt', 20, 6, 120,
 'Mike', 'Oliver', 'Ritchie'),

(1, 3, '2019-12-12',
 'John', 'Riley', 'Kay', 'London', '56 Hight St', '0171-7745632',
 'SH14', 'Shoes', 55, 5, 275,
 'Mike', 'Oliver', 'Ritchie'),

(2, 1, '2019-05-17',
 'Tony', 'Daniel', 'Smith', 'Glasgow', '64 Fern Rd', '0147-3921789',
 'HT23', 'Hat', 28, 5, 140,
 'Aline', 'Agnes', 'Stewart'),

(2, 2,  '2019-05-17',
 'Tony', 'Daniel', 'Smith', 'Glasgow', '64 Fern Rd', '0147-3921789',
 'TS777', 'T-shirt', 30, 10, 300,
 'Aline', 'Agnes', 'Stewart'),

(3, 1, '2019-05-19',
 'John', 'Robert', 'Walker', 'Aberdeen', '5 Tarbot Rd', '0141-8481825',
 'SH14', 'Shoes', 55, 10, 550,
 'Aline', 'Agnes', 'Stewart'),

(3, 2, '2019-05-19',
 'John', 'Robert', 'Walker', 'Aberdeen', '5 Tarbot Rd', '0141-8481825',
 'HT23', 'Hat', 28, 10, 280,
 'Aline', 'Agnes', 'Stewart')

SELECT * FROM Orders
-------------------------------------------------------------------------
--                                                                Друга НФ
-------------------------------------------------------------------------
-- Друга нормальна форма (2NF) - задовольняє першій нормальній формі,
-- а також кожне поле повинно залежити від всього ключа.
-- В таблиці Orders всі поля, окрім ProductID, ProductDescription, UnitPrice, Qty, TotalPricез,
-- залежать не від всього ключа.
DROP TABLE Orders
DROP TABLE Employees
DROP TABLE Customers
DROP TABLE OrderDetails
CREATE TABLE Employees
       (
              EmployeeID int NOT NULL IDENTITY
                     PRIMARY KEY,
              FName nvarchar(15) NOT NULL,
              LName nvarchar(15) NOT NULL,
              MName nvarchar(15) NOT NULL,
              Salary money NOT NULL,
              PriorSalary money NOT NULL,
              HireDate date NOT NULL,
              TerminationDate date NULL,
```

```
                ManagerEmpID int NULL
        )
GO
CREATE TABLE Customers
        (
        CustomerNo int NOT NULL IDENTITY
                PRIMARY KEY,
        FName nvarchar(15) NOT NULL,
        LName nvarchar(15) NOT NULL,
        MName nvarchar(15) NULL,
        Address1 nvarchar(50) NOT NULL,
        Address2 nvarchar(50) NULL,
        City nchar(10) NOT NULL,
        Phone char(12) NOT NULL UNIQUE,
        DateInSystem date NULL,

        CHECK (Phone LIKE '[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
        )
GO
CREATE TABLE Orders
(
        OrderID int NOT NULL IDENTITY PRIMARY KEY,
        OrderDate date NOT NULL,
        CustomerNo int,
        EmployeeID int,

        FOREIGN KEY(CustomerNo) REFERENCES Customers(CustomerNo),
        FOREIGN KEY(EmployeeID) REFERENCES Employees(EmployeeID),
)
CREATE TABLE OrderDetails
(
        OrderID int NOT NULL,
        LineItem int NOT NULL,
        ProductID char(5) NOT NULL,
        ProductDescription varchar(15),
        UnitPrice smallmoney NOT NULL,
        Qty int NOT NULL,
        TotalPrice as Qty * UnitPrice,
    FOREIGN KEY(OrderID) REFERENCES Orders(OrderID),
        PRIMARY KEY (OrderId, LineItem)
)

INSERT Employees
(FName, MName, LName, Salary, PriorSalary, HireDate, TerminationDate, ManagerEmpID)
VALUES
('Mike', 'Oliver', 'Ritchie', 8000, 500, CAST('2017-10-25' AS Date), NULL, NULL),
('Aline', 'Agnes', 'Stewart', 5000, 0, CAST('2018-10-22' AS Date), NULL, 2),
('Charlie', 'Oliver', 'Benson', 10000, 800, '2012-11-20', NULL, 1),
('Jane', 'Agata', 'Evans', 900, 0, '2018-11-20', NULL, 2);
GO
INSERT Customers
(LName, FName, MName, Address1, Address2, City, Phone,DateInSystem)
VALUES
('Kay', 'John', 'Riley','56 Hight St',NULL,'London','0171-7745632','2019-12-12'),
('Smith', 'Tony', 'Daniel','64 Fern Rd',NULL,'Glasgow','0147-3921789','2019-05-17'),
('Walker', 'John', 'Robert','5 Tarbot Rd',NULL,'Aberdeen','0141-8481825','2019-09-19'),
('Fulton','Daniel','Harry','32 Manse Rd','56 Clover Dr','Bristol','0149-3221786','2019-05-14'),
('Morrison','Noah','Matthew','5 Novar Dr','6 Lawrence St','Glasgow','0147-7887789','2019-11-19');
GO
INSERT Orders
VALUES
( '2019-12-12', 1, 2),
( '2019-05-17', 3, 4),
( '2019-09-19', 5, 4)
INSERT OrderDetails
VALUES
( 1, 1, 'JN754', 'Jeans', 35, 7),
( 1, 2, 'BL58', 'Belt', 20, 6 ),
( 1, 3, 'SH14', 'Shoes', 55, 5 ),
( 2, 1, 'HT23', 'Hat', 28, 5 ),
( 2, 2, 'TS777', 'T-shirt', 30, 10 ),
( 3, 1, 'SH14', 'Shoes', 55, 10 ),
( 3, 2, 'HT23', 'Hat', 28, 10 )
SELECT * FROM Customers
SELECT * FROM Employees
SELECT * FROM Orders
SELECT * FROM OrderDetails
```

```
-------------------------------------------------------------------------
--                                                              Третя НФ
-------------------------------------------------------------------------
-- Третя нормальна форма (3NF) - задовольняє 2NF, а також в жодному
-- неключовому полі не може бути залежності від іншого неключового
-- поля. Наявність в таблиці похідних даних не допускається.
DROP TABLE OrderDetails
DROP TABLE Products
CREATE TABLE Products
(
        ProductID char(5) NOT NULL PRIMARY KEY,
        [Description] varchar(15),
        UnitPrice smallmoney NOT NULL,
        [Weight] numeric(18, 0) NULL
)

CREATE TABLE OrderDetails
(
        OrderID int NOT NULL,
        LineItem int NOT NULL,
        ProductID char(5) NOT NULL,
        Qty int NOT NULL,

    FOREIGN KEY(OrderID) REFERENCES Orders(OrderID),
        FOREIGN KEY(ProductID) REFERENCES Products(ProductID),
        PRIMARY KEY (OrderId, LineItem)
)

INSERT Products
VALUES
( 'JN754', 'Jeans', 35, 2),
( 'BL58', 'Belt', 20,  1),
( 'TS777', 'T-shirt', 30,  2),
( 'SH14', 'Shoes', 55,  2),
( 'HT23', 'Hat', 28,  1)

INSERT OrderDetails
VALUES
( 1, 1, 'JN754', 7 ),
( 1, 2, 'BL58', 6 ),
( 1, 3, 'SH14', 5 ),
( 2, 1, 'HT23', 5 ),
( 2, 2, 'TS777', 10 ),
( 3, 1, 'SH14', 10 ),
( 3, 2, 'HT23', 10 )

SELECT * FROM Customers
SELECT * FROM Employees
SELECT * FROM Orders
SELECT * FROM OrderDetails
SELECT * FROM Products
-------------------------------------------------------------------------
--                                                        Денормалізація
-------------------------------------------------------------------------
-- Денормалізація - процес зниження нормальної форми.
-- Здійснюється тоді, коли приведена вища форма призводить
-- до погіршення практичного використання

DROP TABLE OrderDetails
CREATE TABLE OrderDetails
(
        OrderID int NOT NULL,
        LineItem int NOT NULL,
        ProductID char(5) NOT NULL,
        Qty int NOT NULL,
        TotalPrice money,
    FOREIGN KEY(OrderID) REFERENCES Orders(OrderID),
        FOREIGN KEY(ProductID) REFERENCES Products(ProductID),
        PRIMARY KEY (OrderId, LineItem)
)
INSERT OrderDetails
VALUES
( 1, 1, 'JN754', 7, 7 * (select UnitPrice FROM Products where ProductID = 'JN754')),
( 1, 2, 'BL58', 6, 6 * (select UnitPrice FROM Products where ProductID = 'BL58')),
( 1, 3, 'SH14', 5, 5 * (select UnitPrice FROM Products where ProductID = 'SH14')),
( 2, 1, 'HT23', 5, 5 * (select UnitPrice FROM Products where ProductID = 'HT23') ),
( 2, 2, 'TS777', 10, 10 * (select UnitPrice FROM Products where ProductID = 'TS777') ),
```

```
( 3, 1, 'SH14', 10, 10 * (select UnitPrice FROM Products where ProductID = 'SH14') ),
( 3, 2, 'HT23', 10, 10 * (select UnitPrice FROM Products where ProductID = 'HT23') )
SELECT * FROM OrderDetails
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Провести нормалізацію всіх таблиць бази даних DreamHomeDB. Надати розгорнуті пояснення.

Завдання 3

Нормалізувати таблицю, наведену нижче. Надати розгорнуті пояснення.

Завдання 4

Зайдіть на сайт MSDN.

Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп'ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Бібліотека», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов'язані з цим кроком.

Варіант 2

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Університет», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов'язані з цим кроком.

Варіант 3

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Виробництво», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов'язані з цим кроком.

Варіант 4

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Автомайстерня», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов'язані з цим кроком.

Варіант 5

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Сесія», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 6

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Керування проєктом», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 7

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Поліклініка», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 8

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Телефонізація», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 9

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Спорт», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 10

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Сільскогосподарські постачання», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 11

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Міський транспорт», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 12

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Географія», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 13

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Управління будинком», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 14

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Аеропорт», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 15

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Ринок ПК», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 16

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Деканат», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 17

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Зоопарк», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 18

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Шахи», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 19

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Судноплавство», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 20

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Автотранспортне підприємство», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 21

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Наукові конференції», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 22

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Програмні продукти», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 23

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Театр», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 24

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Довідникова аптек», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 25

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Ресторан», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 26

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Формула-1», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 27

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Тюнінг автомобілей», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 28

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Археологія», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов’язані з цим кроком.

Варіант 29

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Тур-фірма», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов'язані з цим кроком.

Варіант 30

Нормалізація таблиць:

1. Провести нормалізацію таблиць БД, що розробляється за напрямком «Телемовлення», до 3NF. Надати розгорнуті пояснення.
2. Провести денормалізацію та пояснити причини, пов'язані з цим кроком.

Рекомендовані ресурси

MSDN: Primary key
https://msdn.microsoft.com/ru-ru/library/ms191236.aspx

MSDN: Foreign key
https://msdn.microsoft.com/ru-ru/library/ms175464.aspx

MSDN: Tables
msdn.microsoft.com/ru-ru/library/ms189104.aspx

Контрольні питання

1. Що таке нормалізація таблиць?
2. Що таке денормалізація таблиць?
3. Які обмеження існують для першої нормальної форми?
4. Які обмеження існують для другої нормальної форми?
5. Які обмеження існують для третьої нормальної форми?

Комп'ютерний практикум 5
Операція з'єднання

Мета роботи: розгляд варіацій з'єднань (Join's).

Теоретичні відомості

- ***Реляційна алгебра*** – це теоретична мова операцій, які на основі одного або декількох відношень дозволяють створювати інше відношення без зміни вихідних відношень.
- ***Операція вибірки*** (selection) проводиться над кортежами одного відношення. Результат вибірки – нове відношення, яке складається з кортежів вихідного відношення, що задовольняють заданій умові.
- ***Операція проєкції*** (projection) проводиться над кортежами одного відношення. Результат проєкції – нове відношення, яке містить лише задані атрибути вихідного відношення.
- ***Декартовий добуток*** (cartesian product) – операція над двома відношеннями, в результаті якої отримується нове відношення, що складається з усіх можливих кортежів, які є попарними поєднанням кортежів вихідних відношень.
- ***Декартовий добуток*** двох таблиць – це інша таблиця, яка складається з усіх можливих пар рядків, що входять до складу обох таблиць. Набір стовпців результуючої таблиці – це всі стовпці першої таблиці, за якими слідують всі стовпці другої таблиці.
- ***Об'єднання*** (union) – операція над двома відношеннями, в результаті якої отримується нове відношення, що складається з усіх кортежів вихідних відношень. Спільні для вихідних відношень кортежі в новому відношенні зустрічаються лише по одному разу.
- ***Різниця*** (set difference) – операція над двома відношеннями, в результаті якої отримується нове відношення, що складається з кортежів, які належать першому відношенню і не належать другому.
- ***Поєднання*** (join) – операція над двома відношеннями, які мають спільні атрибути, в результаті якої отримується нове відношення, що складається з усіх атрибутів вихідних відношень і об'єднує лише ті кортежі вихідних відношень, в яких значення спільних атрибутів співпадають.
- ***Перетин*** (intersection) – операція над двома відношеннями, в результаті якої отримується нове відношення, що складається з кортежів, які належать обом вихідним відношенням.
- Результатом ***ділення*** (division) відношення A(A1, A2, …, An) на відношення B(B1, B2, …, Bn) по відношенню C(A1, A2, …, An, B1, B2, …, Bn) називається нове відношення, що

містить всі такі кортежі *x* з відношення A, для яких виконується умова: для будь-якого *y*, що належить множині кортежів B, відношення C містить кортеж *x*:*y*.

- ***Операція з'єднання*** в мові SQL використовується для об'єднання інформації з двох таблиць за допомогою утворення пар пов'язаних рядків, обраних з кожної таблиці. Пари рядків, які розміщуються в об'єднану таблицю, складаються за рівністю значень зазначених стовпців, що входять до них.
- Якщо необхідно отримати інформацію більш ніж з однієї таблиці, то можна або застосувати підзапит, або виконати з'єднання таблиць. Якщо результуюча таблиця запиту повинна містити стовпці з різних вихідних таблиць, то доцільно використовувати механізм з'єднання таблиць.
- ***З'єднання*** є підмножиною більш загальної комбінації даних двох таблиць, що називається їх декартовим добутком.
- ***Внутрішнє з'єднання*** (INNER JOIN) – з'єднання, при якому в запиті всі записи з таблиці на лівому і правому боці операції INNER JOIN додаються до результуючого набору записів, за відповідності умові значень в зв'язаних полях.
- ***Ліве зовнішнє з'єднання*** (LEFT OUTER JOIN) – зовнішнє з'єднання, при якому в запиті всі записи з таблиці на лівому боці операції LEFT OUTER JOIN в інструкції SQL додаються до результуючого набору записів, навіть якщо в таблиці на правому боці відсутні співпадаючі значення в зв'язаних полях.
- ***Праве зовнішнє з'єднання*** (RIGHT OUTER JOIN) – зовнішнє з'єднання, при якому в запиті всі записи з таблиці на правому боці операції RIGHT OUTER JOIN в інструкції SQL додаються до результуючого набору записів, навіть якщо в таблиці на лівому боці відсутні співпадаючі значення в зв'язаних полях.
- ***Повне з'єднання*** (FULL OUTER JOIN) – зовнішнє з'єднання, при якому в запиті всі записи з таблиці на лівому та правому боці операції FULL OUTER JOIN додаються до результуючого набору записів, за відповідності умові значень в зв'язаних полях, а також:
    1) значення з правої таблиці, що не мають відповідностей в лівій таблиці;
    2) значення з лівої таблиці, що не мають відповідностей в правій таблиці.
- ***Перехресне з'єднання*** (CROSS JOIN) – виконує декартовий добуток таблиць, що залучені до з'єднання. В CROSS JOIN не використовується конструкція ON.

Практичні приклади

```
USE ShopDB
GO
```

```
CREATE TABLE JoinTest1
(id_jt1 int,
 name varchar(50));
GO

CREATE TABLE JoinTest2
(id_jt2 int,
 name varchar(50));
GO

INSERT JoinTest1
VALUES  (1,'one'),
               (2,'two'),
               (3,'three'),
               (4,'four'),
               (5,'five'),
               (9,'nine'),
               (10,'ten');

INSERT JoinTest2
VALUES  (1,'one'),
               (2,'two'),
               (3,'three'),
               (4,'four'),
               (5,'five'),
               (6,'six'),
               (7,'seven'),
               (8,'eight');
select * from JoinTest1;
select * from JoinTest2;



-------------------------------------------------------------------------
--                              INNER JOIN
-------------------------------------------------------------------------

-- INNER JOIN (внутрішнє з'єднання) - з'єднання, при якому
--   в запиті всі записи з таблиці на лівому та правому боці операції
--   INNER JOIN додаються в результуючий набір записів, за відповідності
--   умові значень в зв'язаних полях.

-- Виконуємо вибірку всіх даних з об'єднання таблиць JoinTest1 та JoinTest2
-- за зв'язуючими полями id_jt1 та id_jt2.

SELECT * FROM
                        JoinTest2      -- Ліва таблиця (Таблиця JoinTest2)
                        INNER JOIN  -- Оператор з'єднання
                        JoinTest1      -- Права таблиця (Таблиця JoinTest1)
ON id_jt1 = id_jt2;             -- Умова з'єднання, за якої значення в комірках, які порінюються,
повинні співпадати.
GO

-------------------------------------------------------------------------
--                LEFT OUTER JOIN, RIGHT OUTER JOIN
-------------------------------------------------------------------------

-- LEFT OUTER JOIN (ліве зовнішнє з'єднання) - зовнішнє з'єднання, при якому
--   в запиті всі записи з таблиці на лівому боці операції LEFT JOIN в
--   інструкції SQL додаються в результуючий набір записів, навіть якщо в
--   таблиці на правому боці відсутні співпадаючі значення в зв'язаних полях.

-- Виконуємо вибірку всіх даних з результуючого набору даних лівого зовнішнього
-- з'єднання таблиць JoinTest1 та JoinTest2 за зв'язуючими полями id_jt1 та id_jt2.

SELECT * FROM JoinTest2        -- Ліва таблиця JoinTest2
    LEFT OUTER JOIN JoinTest1  -- LEFT JOIN
ON id_jt1=id_jt2;
GO


-- RIGHT OUTER JOIN (праве зовнішнє з'єднання) - зовнішнє з'єднання, при якому
--   в запиті всі записи з таблиці на правому боці операції RIGHT JOIN в
--   інструкції SQL додаються в результуючий набір записів, навіть якщо в
--   таблиці на лівому боці відсутні співпадаючі значення в зв'язаних полях.

-- Виконуємо вибірку всіх даних з результуючого набору даних правого зовнішнього
```

```
-- з'єднання таблиць JoinTest1 та JoinTest2 за зв'язуючими полями id_jt1 та id_jt2.
SELECT * FROM JoinTest2
   RIGHT OUTER JOIN JoinTest1 -- Права таблиця JoinTest1
ON id_jt1 = id_jt2;
GO


--------------------------------------------------------------------------
--                        FULL OUTER JOIN
--------------------------------------------------------------------------

-- FULL OUTER JOIN (повне з'єднання) - зовнішнє з'єднання, при якому
--   в запиті всі записи з таблиці на лівому та правому боці операції
--   FULL JOIN додаються в результуючий набір записів, за відповідності
--   умові значень в зв'язаних полях, а також:
--             - значення з правої таблиці, що не мають відповідностей в лівій таблиці;
--             - значення з лівої таблиці, що не мають відповідностей в правій таблиці.

SELECT *
FROM JoinTest2
FULL OUTER JOIN JoinTest1      --FULL JOIN
ON id_jt1 = id_jt2;
GO


--------------------------------------------------------------------------
--                           CROSS JOIN
--------------------------------------------------------------------------

-- CROSS JOIN (перехресне з'єднання) - виконує декартовий добуток таблиць,
-- які залучені до з'єднання. В CROSS JOIN не використовується конструкція ON.

-- Виконуємо вибірку всіх даних з результуючого набору даних перехресного
-- з'єднання таблиць JoinTest1 та JoinTest2.

SELECT * FROM JoinTest1
   CROSS JOIN JoinTest2 -- CROSS JOIN
-- ON - не використовується
GO
--------------------------------------------------------------------------
USE ShopDB
GO
--------------------------------------------------------------------------
--                             UNION
--------------------------------------------------------------------------

-- UNION об'єднує результати двох запитів SELECT в єдину результуючу таблицю.

-- Якщо результати обох запитів містять рядки з співпадаючими значеннями комірок,
-- то операція UNION вносить в результуючу таблицю лише один такий рядок.

-- Якщо в результаті одного з запитів виходять рядки з унікальними значеннями,
-- які не співпадають з жодним з рядків результата іншого запиту, то цей рядок
-- також вноситься в результуючу таблицю.

-- Операція UNION вимагає використання таких запитів, кожен з яких повертає
-- вибірку в табличному наданні, при цьому, типи та кількість стовпців повинні співпадати.

SELECT * FROM JoinTest1
UNION
SELECT * FROM JoinTest2
--------------------------------------------------------------------------
--                             UNION ALL
--------------------------------------------------------------------------

-- UNION ALL об'єднує результати двох запитів SELECT в єдину результуючу таблицю.

-- Якщо результати обох запитів містять рядки з співпадаючими значеннями комірок,
-- то операція UNION ALL вносить в результуючу таблицю всі рядки, що дублюються.

-- Якщо в результаті одного з запитів виходять рядки з унікальними значеннями,
-- які не співпадають з жодним з рядків результата іншого запиту, то цей рядок
-- також вноситься в результуючу таблицю.

-- Операція UNION ALL вимагає використання таких запитів, кожен з яких повертає
-- вибірку в табличному наданні, при цьому, типи та кількість стовпців повинні співпадати.
```

```
SELECT * FROM JoinTest1
UNION ALL
SELECT * FROM JoinTest2

-------------------------------------------------------------------------
--                             EXCEPT
-------------------------------------------------------------------------

-- EXCEPT виключає результати правого запиту.

-- Якщо результат лівого запиту операції EXCEPT містить унікальні рядки, які не співпадають
-- з жодним з рядків правого запиту, то тільки такі рядки вносяться в результуючу таблицю.
-- Унікальні рядки правого запиту операції EXCEPT, ніколи не входять в результуючу таблицю.

-- Якщо результати обох запитів містять співпадаючі рядки, то операція EXCEPT їх ігнорує.

-- Операція EXCEPT вимагає використання таких запитів, кожен з яких повертає
-- вибірку в табличному наданні, при цьому, типи та кількість стовпців повинні співпадати.

SELECT * FROM JoinTest1
EXCEPT
SELECT * FROM JoinTest2

SELECT * FROM JoinTest2
EXCEPT
SELECT * FROM JoinTest1

-------------------------------------------------------------------------
--                              INTERSECT
-------------------------------------------------------------------------

-- INTERSECT об'єднує результати двох запитів SELECT в єдину результуючу таблицю.

-- Якщо результати обох запитів містять рядки з співпадаючими значеннями комірок,
-- то, операція INTERSECT вносить в результуючу таблицю лише один такий рядок.

-- Якщо в результаті одного з запитів виходять унікальні рядки,
-- які не співпадають з жодним з рядків результата іншого запиту,
-- то такі рядки ігноруються операцією INTERSECT.

-- Операція INTERSECT вимагає використання таких запитів, кожен з яких повертає
-- вибірку в табличному наданні, при цьому, типи та кількість стовпців повинні співпадати.

SELECT * FROM JoinTest1
INTERSECT
SELECT * FROM JoinTest2
-------------------------------------------------------------------------
DROP DATABASE ShopDB

CREATE DATABASE ShopDB
ON
(
        NAME = 'ShopDB',
        FILENAME = 'D:\ShopDB.mdf',
        SIZE = 10MB,
        MAXSIZE = 100MB,
        FILEGROWTH = 10MB
)
LOG ON
(
        NAME = 'LogShopDB',
        FILENAME = 'D:\LogShopDB.ldf',
        SIZE = 5MB,
        MAXSIZE = 50MB,
        FILEGROWTH = 5MB
)
COLLATE Cyrillic_General_CI_AS

USE ShopDB;
GO

CREATE TABLE Employees
        (
                EmployeeID int NOT NULL,
                FName nvarchar(15) NOT NULL,
                LName nvarchar(15) NOT NULL,
                MName nvarchar(15) NOT NULL,
```

```
                Salary money NOT NULL,
                PriorSalary money NOT NULL,
                HireDate date NOT NULL,
                TerminationDate date NULL,
                ManagerEmpID int NULL
        )
GO

ALTER TABLE Employees ADD
        CONSTRAINT PK_Employees PRIMARY KEY(EmployeeID)
GO

ALTER TABLE Employees ADD CONSTRAINT
        FK_Employees_Employees FOREIGN KEY(ManagerEmpID)
        REFERENCES Employees(EmployeeID)
GO

CREATE TABLE Customers
        (
        CustomerNo int NOT NULL IDENTITY,
        FName nvarchar(15) NOT NULL,
        LName nvarchar(15) NOT NULL,
        MName nvarchar(15) NULL,
        Address1 nvarchar(50) NOT NULL,
        Address2 nvarchar(50) NULL,
        City nchar(10) NOT NULL,
        Phone char(12) NOT NULL,
        DateInSystem date NULL
        )
GO

ALTER TABLE Customers ADD
        CONSTRAINT PK_Customers PRIMARY KEY(CustomerNo)
GO


CREATE TABLE Orders
        (
        OrderID int NOT NULL IDENTITY,
        CustomerNo int NULL,
        OrderDate date NOT NULL,
        EmployeeID int NULL
        )
GO

ALTER TABLE Orders ADD
        CONSTRAINT PK_Orders PRIMARY KEY (OrderID)

GO

ALTER TABLE Orders ADD CONSTRAINT
        FK_Orders_Customers FOREIGN KEY(CustomerNo)
        REFERENCES Customers(CustomerNo)
                ON UPDATE  CASCADE
                ON DELETE  SET NULL

GO

ALTER TABLE Orders ADD CONSTRAINT
        FK_Orders_Employees FOREIGN KEY(EmployeeID)
        REFERENCES Employees(EmployeeID)
                ON UPDATE  CASCADE
                ON DELETE  SET NULL
GO
---
CREATE TABLE Products
        (
        ProdID int NOT NULL IDENTITY,
        Description nchar(50) NOT NULL,
        UnitPrice money NULL,
        Weight numeric(18, 0) NULL
        )
GO

ALTER TABLE Products ADD CONSTRAINT
        PK_Products PRIMARY KEY (ProdID)
GO
```

```
CREATE TABLE OrderDetails
        (
        OrderID int NOT NULL,
        LineItem int NOT NULL,
        ProdID int NULL,
        Qty int NOT NULL,
        UnitPrice money NOT NULL,
        TotalPrice as Qty*UnitPrice
        )
GO


ALTER TABLE OrderDetails ADD CONSTRAINT
        PK_OrderDetails PRIMARY KEY
        (
        OrderID,
        LineItem
        )
GO


ALTER TABLE OrderDetails ADD CONSTRAINT
        FK_OrderDetails_Products FOREIGN KEY(ProdID)
                REFERENCES Products(ProdID)
                ON UPDATE  NO ACTION
                ON DELETE  SET NULL
GO


ALTER TABLE OrderDetails ADD CONSTRAINT
        FK_OrderDetails_Orders FOREIGN KEY(OrderID)
        REFERENCES Orders(OrderID)
         ON UPDATE  CASCADE
         ON DELETE  CASCADE
GO


INSERT Employees
(EmployeeID, FName, MName, LName, Salary, PriorSalary, HireDate, TerminationDate, ManagerEmpID)
VALUES
(1,'Mike', 'Oliver', 'Ritchie', 8000, 500, CAST('2017-10-25' AS Date), NULL, NULL),
(2,'Aline', 'Agnes', 'Stewart', 5000, 0, CAST('2018-10-22' AS Date), NULL, 1),
(3,'Charlie', 'Oliver', 'Benson', 10000, 800, '2012-11-20', NULL, 2),
(4,'Jane', 'Agata', 'Evans', 900, 0, '2018-11-20', NULL, 2);
GO

INSERT Customers
(LName, FName, MName, Address1, Address2, City, Phone,DateInSystem)
VALUES
('Kay', 'John', 'Riley','56 Hight St',NULL,'London','0171-7745632','2019-12-12'),
('Smith', 'Tony', 'Daniel','64 Fern Rd',NULL,'Glasgow','0147-3921789','2019-05-17'),
('Walker', 'John', 'Robert','5 Tarbot Rd',NULL,'Aberdeen','0141-8481825','2019-09-19'),
('Fulton','Daniel','Harry','32 Manse Rd','56 Clover Dr','Bristol','0149-3221786','2019-05-14'),
('Morrison','Noah','Matthew','5 Novar Dr','6 Lawrence St','Glasgow','0147-7887789','2019-11-19');
GO

INSERT Products
( Description, UnitPrice, Weight )
VALUES
('JN754 Jeans',35,0.9),
('TS777 T-shirt',20,0.3),
('J555 Jeans',48,0.7),
('BL58 Belt',20,0.5),
('SH14 Shoes',55,1),
('HT23 Hat',28,0.35)
GO

INSERT Orders
(CustomerNo, OrderDate, EmployeeID)
VALUES
(1,'2019-03-17',2),
(3,'2019-06-22',4),
(5,'2019-11-19',4)
GO

INSERT OrderDetails
(OrderID, LineItem, ProdID, Qty, UnitPrice)
```

```
VALUES
(1,1,1,5,85),
(1,2,4,5,125),
(1,3,5,5,40),
(2,1,6,10,28),
(2,2,2,15,12),
(3,1,5,20,58),
(3,2,6,18,28)
GO

SELECT * FROM Employees;
SELECT * FROM OrderDetails;
SELECT * FROM Products;
SELECT * FROM Orders;
SELECT * FROM Customers;
-----------------------------------------------------------------------
USE ShopDB
GO

-- Показати статистику за кількістю проданих одиниць товару.
SELECT Products.ProdID, [Description], SUM(Qty) AS Qty, SUM(TotalPrice) AS TotalSold -- Помилки
немає.
          FROM Products
INNER JOIN OrderDetails
      ON Products.ProdID = OrderDetails.ProdID
         GROUP BY Products.ProdID, [Description]


-- Якщо в таблицях, які використовуються, є стовпці з однаковими іменами,
-- то потрібно явно вказати до якої таблиці належить даний стовпець.
SELECT ProdID, [Description], Qty, TotalPrice FROM Products  -- Помилка!
                        JOIN OrderDetails
                        ON Products.ProdID = OrderDetails.ProdID


-- Вивести загальну суму продажів по співробітникам.
SELECT FName, LName, MName, SUM(TotalPrice) AS [Total Sold] FROM Employees -- Аліас [Total Sold]
             JOIN Orders
                        ON Employees.EmployeeID       = Orders.EmployeeID
             JOIN OrderDetails
                        ON Orders.OrderID = OrderDetails.OrderID
GROUP BY  Employees.FName,
                 Employees.LName,
                 Employees.MName

-- Вивести загальну суму продажів по кожному із співробітників

SELECT FName, LName, MName, SUM(TotalPrice) AS [Total Sold] FROM Employees -- Аліас [Total Sold]
             LEFT JOIN Orders
                        ON Employees.EmployeeID       = Orders.EmployeeID
             LEFT JOIN OrderDetails
                        ON Orders.OrderID = OrderDetails.OrderID
GROUP BY  Employees.FName,
                 Employees.LName,
                 Employees.MName

-- Вивести план підлеглості співробітників
SELECT Emp1.FName, Emp1.MName, Emp1.LName, Emp2.FName, Emp2.MName, Emp2.LName
       FROM Employees Emp1 -- Аліас Emp1
                JOIN Employees       Emp2 -- Аліас Emp2
                ON Emp1.EmployeeID = Emp2.ManagerEmpID
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Створити БД DreamHomeDB, яка складається з 5 таблиць: Staff, Property_for_Rent, Renter, Qwner, Viewing.

Branch	(Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff	(Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent	(Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter	(Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner	(Ono, FName, LName, Address, Tel_No)
Viewing	(Rno, Pno, Date, Comment)

Завдання 3

Створити вибірку за допомогою JOIN's для наступних завдань:

1) Скласти список імен всіх клієнтів, які вже оглянули хоча б один об'єкт, що здається в оренду, і повідомили свою думку з цього приводу.

2) Перелічити відділення компанії і об'єкти, що здаються в оренду, які розташовані в одному і тому ж місті, а також інші відділення компанії.

3) Перелічити відділення компанії і об'єкти, що здаються в оренду, які розташовані в одному і тому ж місті, а також всі інші об'єкти.

4) Перелічити відділення компанії і об'єкти, що здаються в оренду, розташовані в одному і тому ж місті, а також всі інші відділення і об'єкти.

Завдання 4

Зайдіть на сайт MSDN.

Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп'ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Вибірки з використанням з'єднань:

1. Вибрати книгу з фізики, яка найбільш популярна у читачів (максмальне число видач за рік).
2. Вибрати читачів, які мають заборгованість більше 4 місяців.
3. Визначити книгу, яка була найбільш популярною взимку 2020 року.
4. Визначити читачів, у яких на руках знаходяться книги видавництва «Освіта і Наука».

Варіант 2

Вибірки з використанням з'єднань:

1. Вибрати викладачів, у яких найбільша кількість пар восени 2020 року.
2. Визначити можливі «накладки» у викладачів в розкладі.
3. Вивести розклад занять викладача «Добролюбова М.В.» на листопад 2020 року.
4. Визначити для кожної групи кількість іспитів і заліків.

Варіант 3

Вибірки з використанням з'єднань:

1. Визначити виріб, з найбільшою тривалістю виробництва.
2. Вивести список устаткування, яке потребує заміни в найближчі півроку.
3. Вивести список виробів, яких виходить найбільше з одиниці об'єму сталі.
4. Вивести середній термін придатності обладнання.

Варіант 4

Вибірки з використанням з'єднань:

1. Вибрати прізвище того механіка, який частіше за всіх працює з новими автомобілями.
2. Вибрати прізвище господаря, що найчастіше відвідує майстерню і має автомобіль марки «Мерседес».
3. Визначити тих власників автомобілів, яких завжди обслуговує один і той самий механік. Вивести прізвища механіка і його постійного клієнта.
4. Для кожної марки автомобіля вибрати механіка, найчастіше обслуговуючого цю марку.

Варіант 5

Вибірки з використанням з'єднань:

1. Вибрати групу з найбільшою тривалістю сесії.
2. Визначити викладача, який в сесію приймає заліки у найбільшого числа студентів.
3. Визначити, відсоток іспитів, що здаються групою «ПА-01мп», від загального числа предметів для кожної категорії.
4. Визначити, чи не здає будь-яка група два іспити або заліки в один день.

Варіант 6

Вибірки з використанням з'єднань:

1. Визначити ті роботи в 2020 році, за якими дата завершення перевищує крайній термін, тобто проєкт прострочений.
2. Визначити максимальну кількість проєктів у одного працівника в 2019 році.
3. Визначити ті роботи, які до дати завершення були виконані не більше, ніж на 50 %.
4. Визначити 5 співробітників, що виконують найбільший обсяг робіт.

Варіант 7

Вибірки з використанням з'єднань:

1. Визначити ті випадки, коли пацієнт лікувався менше місяця.
2. Вивести імена тих лікарів, які працюють виключно з інвалідами.
3. Визначити відсоток смертності від захворювання «каріес».
4. Вивести імена тих пацієнтів, які лікувалися у лікарів усіх спеціальностей.

Варіант 8

Вибірки з використанням з'єднань:

1. Вибрати найбільш популярний тариф для кожного оператора.
2. Визначити сумарну заборгованість по кожному з тарифів.
3. Підрахувати, скільки є незареєстрованих номерів у кожного з операторів.
4. Вибрати список абонентів оператора «Київстар», що мають заборгованість більше 3000 грн.

Варіант 9

Вибірки з використанням з'єднань:

1. Вивести таблицю розподілу місць в змаганні «відкритий чемпіонат» в місті «Васюки» з «шахів» в 2021 році.
2. Визначити спортсменів, які мають як олімпійський, так і світовий рекорд.
3. Вивести список спортсменів, що беруть участь більш ніж у 8 змаганнях на рік в порядку зростання середнього місця на цих змаганнях.
4. Визначити найкращий показник спортсмена «Фуркад» у виді спорту «біатлон» і різницю зі світовим і олімпійським рекордами.

Варіант 10

Вибірки з використанням з'єднань:

1. Вивести підприємства, які провели в поточному році не більше однієї операції.
2. Визначити підприємства, вид діяльності якого є провідним в поставці товару «банан».
3. Визначити випадки, коли були продані товари із терміном придатності не більше тижня.
4. Вивести список продуктів, для яких закупівельна ціна, як правило, нижче собівартості виробника.

Варіант 11

Вибірки з використанням з'єднань:

1. Визначити найприбутковіший засіб пересування (відношення середня плата за проїзд / середні витрати на паливо).
2. Визначити середнє відношення числа тролейбусів на маршруті до числа зупинок.
3. Вивести маршрути машини «Газель» в порядку спадання їх довжини.
4. Визначити очікуваний прибуток фірми за день.

Варіант 12

Вибірки з використанням з'єднань:

1. Розташувати країни за багатонаціональністю в порядку спадання.
2. Підрахувати чисельність населення монархій.
3. Визначити тип управління тієї країни, де проживає найбільше представників національності «вірмени».
4. Вибрати список національностей, які проживають в країнах з анархічним ладом.

Варіант 13

Вибірки з використанням з'єднань:

1. Визначити суму місячної оплати для квартири № 50 в будинку 302 по Садовій вулиці.
2. Визначити заборгованість по оплаті «доставка сміття» Константинова Романа Вікторовича.
3. Визначити загальну суму виплат для квартир по вулиці Боткіна.
4. Вибрати список власників, які мають заборгованості понад рік.

Варіант 14

Вибірки з використанням з'єднань:

1. Визначити середньорозрахунковий час польоту для літака міжнародних перевезень «Boing-24».
2. Вибрати марку літака, яка найчастіше використовується на внутрішніх рейсах.
3. Вибрати маршрут / маршрути, за якими найчастіше літають рейси, заповнені менш, ніж на 70 %.
4. Визначити наявність вільних місць на рейс №354 23 серпня 2019 року.

Варіант 15

Вибірки з використанням з'єднань:

1. Визначити фірму, яка надає найбільшу гарантію на ноутбуки на базі процесора «Pentium Centrino».
2. Вибрати модель з найбільшим розміром екрану, яка випускається в місті «Дніпро».
3. Визначити фірму, що здійснила обсяг продажів на найбільшу суму за 2020 рік.
4. Вибрати міста, в яких випускаються ноутбуки на базі «ASUS».

Варіант 16

Вибірки з використанням з'єднань:

1. Вибрати кількість студентів на кожному факультеті.
2. Вибрати середню величину стипендії для кожної групи факультету «ПБФ».
3. Для кожного факультету вибрати список груп із зазначенням чисельності студентів в кожній групі.
4. Вибрати прізвища і величину стипендії студентів з міста «Житомир».

Варіант 17

Вибірки з використанням з'єднань:

1. Визначити середню площу, що припадає на одну особину в вольєрі «мавпятник».
2. Вибрати випадки розміщення тваринного виду «крокодил» в приміщенні з водоймою, але площею, менше 3 квадратних метрів.
3. Визначити загальну чисельність представників сімейства «вовчі» в зоопарку.
4. Вивести середню тривалість життя для мешканців «тераріуму».

Варіант 18

Вибірки з використанням з'єднань:

1. Вибрати турнір з найбільшою кількістю гросмейстерів.
2. Вибрати ті турніри, де жодне призове місце не зайняв представник країни-господаря турніру.
3. Вибрати тих шахістів, які виграли не менше трьох турнірів протягом 2020.
4. Визначити турніри, в яких учасник з найвищим рейтингом посів останнє місце.

Варіант 19

Вибірки з використанням з'єднань:

1. Вибрати імена капітанів, що приводили кораблі в місто «Варна» взимку 2020/21 років.
2. Визначити, коли і з якою метою судно «Електрон» відвідувало порт «Трансе».
3. Визначити, який прибуток отримав порт міста Миколаїв від кораблів з Туреччини.
4. Визначити, з якою метою найчастіше заходять кораблі в порт «Тортуга».

Варіант 20

Вибірки з використанням з'єднань:

1. Визначити водія, який найчастіше виконує далекі перевезення (> 500 км).
2. Визначити середню вантажопідйомність рейсів з Києва.
3. Визначити загальну витрату палива у перевезеннях, що здійснені у 2019.
4. Визначити середній час поїздки для водія «Шевченко».

Варіант 21

Вибірки з використанням з'єднань:

1. Вибрати вчених, сумарна тривалість виступів яких на конференціях в Україні перевищує 5 годин.
2. Вибрати конференцію, тематика якої не збігається зі спеціалізацією більшості її учасників.
3. Визначити, в якій конференції брало участь більше всього докторів наук.
4. Визначити середній відсоток іноземних вчених на конференціях, що проходять в місті «Київ».

Варіант 22

Вибірки з використанням з'єднань:

1. Визначити сферу застосування, в якій інстальовано найбільшу кількість продуктів.
2. Визначити витрати на придбання / модифікацію програмного забезпечення, зроблені покупцем «UkrEnergo» в 2020 році.
3. Визначити тих покупців, для яких сумарна вартість всіх модифікацій продукту перевищила вартість ліцензії на продукт останньої версії.
4. Вибрати список продуктів типу «серверні операційні системи», в порядку спадання їх популярності.

Варіант 23

Вибірки з використанням з'єднань:

1. Визначити роль актриси, яка найчастіше грає роль «технічки-шпигунки».
2. Підрахувати кількість п'єс для кожного режисера.
3. Вибрати список п'єс, в яких стать виконавців ролей змінювалась.
4. Вибрати список акторів, які зайняті більш ніж в трьох постановках одночасно.

Варіант 24

Вибірки з використанням з'єднань:

1. Визначити, якою хворобою найчастіше страждають клієнти аптеки «Еко-аптека».
2. Визначити, яких збитків зазнає аптека 999, якщо протягом місяця не реалізує всі ліки, у яких закінчується термін придатності.
3. Визначити, в яких аптеках найнижча середня ціна на болезаспокійливі.
4. Вибрати кількість таблеток, на яких доведеться змінити дату закінчення терміну придатності, якщо вони не будуть реалізовані протягом тижня.

Варіант 25

Вибірки з використанням з'єднань:

1. Для клієнта «Петренко» вибрати меню з чотирьох страв (перше, друге, закуска і компот) так, щоб калорійність його була від 600 до 800 кілокалорій.
2. Вибрати найдешевше блюдо, яке задовольняє переваги клієнта «Олена По».
3. Вибрати середній щоденний дохід їдальні.
4. Вибрати найбільш споживану страву в березні.

Варіант 26

Вибірки з використанням з'єднань:

1. Визначити для кожного пілота кількість етапів, кількість набраних очок за системою (10-8-6-5-4-3-2-1) і розташувати в порядку спадання.
2. Визначити для кожної команди середній відсоток проходження етапу її пілотами (відношення пройдених кіл до загальної довжини гонки).
3. Вибрати етапи, на яких більшість очок набирали пілоти на гумі «Michelin».
4. Для кожного пілота визначити середню швидкість на етапі за рік.

Варіант 27

Вибірки з використанням з'єднань:

1. Визначити фірму, яка надає найбільший спектр деталей, кількість цих деталей і середня гарантія.
2. Визначити 10 деталей, які дають найбільшу зміну швидкості.
3. Для кожного власника визначити витрати на тюнінг автомобілів.
4. Визначити найбільш оздоблені автомобілі для кожної марки і їх параметри.

Варіант 28

Вибірки з використанням з'єднань:

1. Вибрати список археологів, їх кваліфікацію, кількість знахідок за останній рік з епохи середньовіччя.
2. Вибрати назву предмета, епоху, кількість знахідок таких предметів, археолога, який переважно знаходив ці предмети.
3. Вивести повну інформацію про предмети, знайдені в районі «Каїра» епохи «Древньоєгипетська».
4. Вивести середню вартість знахідок для кожного археолога за час їх роботи.

Варіант 29

Вибірки з використанням з'єднань:

1. Вибрати відпочиваючого, який найбільше число раз відвідав «Єгипет», вивести цю кількість і його витрати на поїздки.
2. Вибрати моря і назви готелів, які відвідав «Коновалов» в минулому році.
3. Вибрати всі можливі варіанти не більше двох тур-поїздок на суму 3000 $, так щоб відпочинок пройшов на «Середземному» морі.
4. Вибрати кількість днів відпочинку і кількість місць відпочинку в минулому році для кожного клієнта фірми.

Варіант 30

Вибірки з використанням з'єднань:

1. Вибрати канали, які транслюються на найбільшу територію і за специфікою є новинними.
2. Вибрати супутники, радіус їх орбіт і розташувати в порядку спадання за кількістю каналів на них.
3. Вибрати супутники, що не ведуть мовлення жодного каналу своєї країни.
4. Вибрати телекомпанії, що ведуть мовлення на Україну каналів розважальної специфіки.

Рекомендовані ресурси

MSDN: Joins (SQL Server)
https://docs.microsoft.com/en-us/sql/relational-databases/performance/joins?view=sql-server-ver15

MSDN: Set Operators - EXCEPT and INTERSECT
https://docs.microsoft.com/en-us/sql/t-sql/language-elements/set-operators-except-and-intersect-transact-sql?view=sql-server-ver15

MSDN: Set Operators - UNION (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/language-elements/set-operators-union-transact-sql?view=sql-server-ver15

Контрольні питання

1. Як виводяться дані при внутрішньому з'єднанні?
2. Як виводяться дані при зовнішньому з'єднанні?
3. Які існують види зовнішніх з'єднань?
4. Для чого використовується операція UNION?
5. Для чого використовується операція UNION ALL?
6. Для чого використовується операція EXCEPT?
7. Для чого використовується операція INTERSECT?
8. Які рядки ігноруються операцією INTERSECT?
9. Використання яких запитів вимагає операція EXCEPT?
10. Використання яких запитів вимагає операція UNION?
11. Використання яких запитів вимагає операція UNION ALL?
12. Використання яких запитів вимагає операція INTERSECT?
13. Чому при зовнішньому з'єднанні CROSS JOIN не використовується конструкція ON?
14. Що таке операція вибірки?
15. Що таке операція проєкції?
16. Що таке декартовий добуток?
17. Що таке операція різниці?
18. Що таке операція поєднання?
19. Що таке операція об'єднання?
20. Що таке операція перетину?
21. Що таке операція ділення?
22. Який оператор відповідає операції декартового добутку?

Комп'ютерний практикум 6

Вкладені запити

Мета роботи: розгляд підзапитів та курсорів.

Теоретичні відомості

- ***Підзапити*** (вкладені запити, внутрішні запити, внутрішні операції вибору) – це запити, які використовуються в інструкціях SELECT, UPDATE, INSERT, DELETE або в іншому підзапиті. Вкладені запити можуть бути використані всюди, де дозволені вирази.
- ***Зовнішні запити*** (зовнішня операція вибору) – інструкції, які містять вкладені запити.
- ***Зв'язаний підзапит*** – це запит, в якому (в реченнях WHERE, HAVING) вказане поле таблиці зовнішнього запиту.
- Розрізняють наступні ***типи підзапитів***: скалярний, рядковий, табличний.
- ***Скалярний підзапит*** повертає значення, що вибирається з перетину одного стовпця з одним рядком, тобто єдине значення. Скалярний підзапит може використовуватися скрізь, де потрібно вказати єдине значення.
- ***Рядковий підзапит*** повертає значення декількох стовпців таблиці, але у вигляді єдиного рядка. Рядковий підзапит може використовуватися скрізь, де застосовується конструктор рядкових значень – зазвичай це предикати.
- ***Табличний підзапит*** повертає значення одного або більше стовпців таблиці, що розміщені в більш ніж одному рядку. Табличний підзапит може використовуватися скрізь, де допускається вказувати таблицю – наприклад, як операнд предиката IN.
- Багато інструкцій мови T-SQL, що включають вкладені запити, можна записати у вигляді з'єднань. Але деякі запити можуть бути виконані тільки за допомогою вкладених запитів. Зазвичай в T-SQL немає різниці між інструкціями з вкладеними запитами та семантично еквівалентними інструкціями без них щодо продуктивності. Однак в деяких випадках з'єднання показують кращу продуктивність.
- Вкладені запити більш вигідно використовувати, коли потрібно обчислити агрегатне значення "на льоту" і використовувати його в іншому запиті для порівняння або поля, яке виводиться.
- З'єднання більш вигідно використовувати, коли список вибору інструкції SELECT в запиті містить стовпці більш ніж з однієї таблиці.
- ***Тимчасові таблиці*** – це таблиці, які зберігаються в базі даних tempdb та автоматично видаляються, коли зникає необхідність в їх використанні.

- Розрізняють ***локальні*** та ***глобальні*** тимчасові таблиці, які відрізняються одна від одної іменами, видимістю та доступністю. Імена локальних тимчасових таблиць починаються з одного символу (#); вони видимі лише поточному з'єднанню користувача і видаляються, коли користувач відключається від екземпляра Microsoft SQL Server. Імена глобальних таблиць починаються з двох символів номера (##); їх видно будь-якому користувачеві; вони видаляються, коли всі користувачі, які на них посилаються, відключаються від екземпляра Microsoft SQL Server. Локальна тимчасова таблиця, створена зберігаємою процедурою, видаляється автоматично при завершенні процедури.
- ***Узагальнені табличні вирази*** – це тимчасові результуючі набори, які визначені в області виконання поодиноких інструкцій SELECT, INSERT, UPDATE. Узагальнені табличні вирази, як і похідні таблиці, не зберігаються в базі даних у вигляді об'єктів, а час їх життя обмежений тривалістю запиту. Застосування узагальнених табличних виразів дозволяє значно підвищити зрозумілість коду і спростити роботу зі складними запитами, розбивши їх на окремі логічні будівельні блоки. З них можна складати більш складні проміжні узагальнені табличні вирази для формування кінцевого результуючого набору. Узагальнені табличні вирази можуть бути визначені в призначених для користувача підпрограма (функціях, зберігаємих процедурах, тригерах, представленнях).
- Інструкції Microsoft SQL Server створюють повний результуючий набір, але бувають випадки, коли результати зручніше обробляти порядково. Для цого використовують курсори.
- Курсор (current set of record) – тимчасовий набір рядків, які можна перебирати послідовно, з першого до останнього.

Практичні приклади

```
USE ShopDB
GO

DROP TABLE SubTest1;
DROP TABLE SubTest2;

CREATE TABLE SubTest1
(id1 int,
 name varchar(50));
 GO

CREATE TABLE SubTest2
(id2 int,
 name varchar(50));
GO

INSERT SubTest1
VALUES  (1,'one'),
               (2,'two'),
               (3,'three'),
               (4,'four'),
               (5,'five'),
               (9,'nine'),
               (10,'ten');
```

```
GO

INSERT SubTest2
VALUES  (1,'one'),
                (2,'two'),
                (3,'three'),
                (4,'four'),
                (5,'five'),
                (6,'six'),
                (7,'seven'),
                (8,'eight');
GO
SELECT * FROM SubTest1;
SELECT * FROM SubTest2;
-------------------------------------------------------------------------------------------
--                                                                  Вкладені запити
-------------------------------------------------------------------------------------------
-- Вкладений запит - внутрішня складова частина загального запиту (запит всередині запиту)

SELECT * FROM SubTest1
WHERE id1 IN
                        (SELECT id2 FROM SubTest2); -- вкладений запит

-- Якщо вкладений запит повертає більше одного значення, то використовується умова порівняння IN;
-- Якщо вкладений запит повертає одне значення, то використовується умова порівняння =;

SELECT * FROM SubTest1
WHERE id1 =       -- ПОМИЛКА
                        (SELECT id2 FROM SubTest2); -- вкладений запит

-- Вкладений запит може також містити конструкцію WHERE

SELECT * FROM SubTest1
WHERE id1 =
                        (SELECT id2 FROM SubTest2 WHERE name = 'four');
-------------------------------------------------------------------------------------------
USE AdventureWorks2019;
GO

SELECT FirstName + ' ' + LastName as Name, BirthDate
FROM Person.Person as pc    -- Person - схема (аналог простору імен)
JOIN HumanResources.Employee as he
ON pc.BusinessEntityID = he.BusinessEntityID
WHERE BirthDate = '1977-02-03'

-- Вкладені запити можна використовувати разом з JOIN's

SELECT FirstName + ' ' + LastName as Name, BirthDate
FROM Person.Person as pc
JOIN HumanResources.Employee as he
ON pc.BusinessEntityID = he.BusinessEntityID
WHERE BirthDate = (SELECT MIN(BirthDate) FROM HumanResources.Employee );

-------------------------------------------------------------------------------------------

USE ShopDB
GO
-----------------------------------------------------------------------
--                                  Зв'язані вкладені запити
-----------------------------------------------------------------------
--  Зв'язані вкладені запити
--      1. Зовнішній запит отримує рядок і він передається у внутрішній запит.
--      2. Внутрішній запит виконується з врахуванням отриманих значень.
--      3. Внутрішній запит передає у зовнішній результуюче значення.
--      4. Зовнішній запит використовує ці значення для завершення наміченої обробки.

SELECT * FROM SubTest1 AS ST1
WHERE name /* 4 */ = /* 3 */(SELECT name                                    --|(2)
                                                        FROM SubTest2 AS ST2                    --
|(2)
                                                        WHERE ST1.id1 = ST2.id2); -- (1)  --|(2)

SELECT * FROM SubTest1 AS ST1
WHERE name=(SELECT name
                        FROM SubTest2 AS ST2
                        WHERE ST1.id1 = ST2.id2);
```

```
-- Швидше виконується наступним чином:

-- EXISTS - повертає true, якщо вибірка повертає хоча б одне значення.
-- після знаходження потрібного значення EXISTS, не продовжує пошук по таблиці.
SELECT * FROM SubTest1  AS ST1
WHERE EXISTS
    (SELECT * FROM SubTest2   ST2
     WHERE ST1.id1 = ST2.id2);


--  Зв'язаний вкладений запит в списку вибірки
-- , (кома)
SELECT * ,(SELECT nameFROM SubTest2 AS ST2 WHERE ST1.id1 = ST2.id2) AS Name2
FROM SubTest1 AS ST1;

SELECT *,
          (SELECT id2 FROM SubTest2 AS ST2  WHERE ST1.id1 = ST2.id2) AS Id2,
          (SELECT name       FROM SubTest2 AS ST2 WHERE ST1.id1 = ST2.id2) AS Name2
FROM SubTest1 AS ST1;

SELECT *,
          (SELECT id2 FROM SubTest2 AS ST2  WHERE ST1.id1 = ST2.id2) AS Id2,
          (SELECT name       FROM SubTest2 AS ST2 WHERE ST1.id1 = ST2.id2) AS Name2
FROM SubTest1 AS ST1
WHERE ST1.id1 = (SELECT id2 FROM SubTest2 AS ST2
                 WHERE ST1.id1 = ST2.id2);

-- Показати статистику по кількості проданих одиниць товару.

SELECT Products.ProductID, [Description], Qty, TotalPrice
         FROM Products
INNER JOIN OrderDetails
      ON Products.ProductID = OrderDetails.ProductID

-----------------------------------------------------------------------

SELECT (SELECT  ProductID FROM Products
           WHERE Products.ProductID = OrderDetails.ProductID) AS ProdID,
          (SELECT  [Description] FROM Products
           WHERE Products.ProductID = OrderDetails.ProductID) AS [Description], Qty, TotalPrice
FROM OrderDetails

-- Вивести загальну суму продажу по співробітниках

SELECT FName, LName, MName, SUM(TotalPrice) AS [Total Sold]
              FROM Employees
             JOIN Orders
                       ON Employees.EmployeeID       = Orders.EmployeeID
             JOIN OrderDetails
                       ON Orders.OrderID = OrderDetails.OrderID
GROUP BY  Employees.FName,
                Employees.LName,
                Employees.MName

-- Створення тимчасової таблиці CREATE TABLE #TableName:
-- #TableName  - локальна таблиця, даною таблицею может користуватись лише
--               поточний користувач
-- ##TableName - глобальна таблиця, даною таблицею может користуватись будь-який
--                             користувач
CREATE TABLE #TmpTable
(FName varchar(50),
 LName varchar(50),
 MName varchar(50),
 TotalPrice money);
 GO

SELECT (SELECT FName FROM Employees
              WHERE EmployeeID = (SELECT EmployeeID FROM Orders
                                                    WHERE Orders.OrderID = OrderDetails.OrderID)
              ) AS FName,
            (SELECT LName FROM Employees
                WHERE EmployeeID = (SELECT EmployeeID FROM Orders
                                                      WHERE Orders.OrderID = OrderDetails.OrderID)
              ) AS LName,
              (SELECT MName FROM Employees
               WHERE EmployeeID = (SELECT EmployeeID FROM Orders
                                                    WHERE Orders.OrderID = OrderDetails.OrderID)
```

```
                ) AS MName,
                TotalPrice
INTO TmpTable
FROM OrderDetails;
GO

SELECT FName, LName, MName, SUM(TotalPrice) AS [Total Sold] FROM TmpTable
GROUP BY FName, LName, MName;
GO

-- WITH ... AS - Узагальнені табличні вирази - використовуються замість тимчасових таблиць,
-- що запобігає зберіганню даних, які дублюються.

WITH Managers (FName, LName, MName, TotalPrice) AS  -- не аліас!
(
SELECT (
        SELECT FName FROM Employees
               WHERE EmployeeID = (
                                   SELECT EmployeeID FROM Orders
                                                     WHERE Orders.OrderID = OrderDetails.OrderID
                                                     )
               ),
               (
                SELECT LName FROM Employees
                WHERE EmployeeID = (
                                    SELECT EmployeeID FROM Orders
                                                      WHERE Orders.OrderID = OrderDetails.OrderID
                                                      )
               ),
               (
                SELECT MName FROM Employees
                WHERE EmployeeID = (
                                    SELECT EmployeeID FROM Orders
                                                      WHERE Orders.OrderID = OrderDetails.OrderID
                                                      )
               ),
               TotalPrice
FROM OrderDetails
)
         SELECT FName, LName, MName, SUM(TotalPrice) AS [Total Sold]
         FROM Managers GROUP BY FName, LName, MName;
GO

-----------------------------------------------------------------------

-- Увага!!! Використовуйте курсори тільки за крайньої потреби!
-- Курсор - набір записів разом з вказівником, який ідентифікує поточний рядок.

USE AdventureWorks2019;
GO

--------------------------- Визначення курсора -------------------------------------

DECLARE contact_cursor CURSOR           -- оголошення курсора
FOR
               SELECT * FROM Person.Person   -- відбір рядків для курсора

OPEN contact_cursor                      -- відкриття курсора
CLOSE contact_cursor                    -- закриття курсора
DEALLOCATE contact_cursor               -- видалення курсора
GO

-- FETCH отримує дані з курсора

DECLARE contact_cursor CURSOR
       SCROLL              -- встановлює обмеження, SCROLL (дозволяє рухатись курсору в будь-
якому напрямку)
       FOR
       SELECT FirstName, LastName FROM Person.Person

OPEN contact_cursor


DECLARE @FirstName nvarchar(50),
               @LastName nvarchar(50);
PRINT '1 NEXT'
FETCH NEXT FROM contact_cursor       -- NEXT - отримати рядок
```

```
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName


DECLARE @FirstName nvarchar(50),
                @LastName nvarchar(50);
PRINT '2 NEXT'
FETCH NEXT FROM contact_cursor       -- NEXT - отримати рядок
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName


DECLARE @FirstName nvarchar(50),
                @LastName nvarchar(50);
PRINT '3 PRIOR'
FETCH PRIOR FROM contact_cursor      -- PRIOR - отримати попередній рядок
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName


DECLARE @FirstName nvarchar(50),
                @LastName nvarchar(50);
PRINT '4 LAST'
FETCH LAST FROM contact_cursor       -- LAST - отримати останній рядок
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName


DECLARE @FirstName nvarchar(50),
                @LastName nvarchar(50);
PRINT '5 FIRST'
FETCH FIRST FROM contact_cursor      -- FIRST - отримати перший рядок
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName


DECLARE @FirstName nvarchar(50),
                @LastName nvarchar(50);
PRINT '6 ABSOLUTE'
FETCH ABSOLUTE 5 FROM contact_cursor -- ABSOLUTE n - отримати рядок під номером n
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName


DECLARE @FirstName nvarchar(50),
                @LastName nvarchar(50);
PRINT '7 RELATIVE'
FETCH RELATIVE 5 FROM contact_cursor -- RELATIVE n - отримати n-нтий рядок після поточного
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName

CLOSE contact_cursor
DEALLOCATE contact_cursor
GO

-- LOCAL - дозволяє автоматично закривати та видаляти курсор після виконання скрипта.

DECLARE contact_cursor CURSOR
        LOCAL             -- встановлюємо обмеження LOCAL
        FOR
        SELECT FirstName, LastName FROM Person.Person

OPEN contact_cursor

DECLARE @FirstName nvarchar(50), @LastName nvarchar(50);

FETCH NEXT FROM contact_cursor       -- NEXT - отримати рядок
        INTO @FirstName, @LastName
PRINT @FirstName + ' ' + @LastName
GO

DEALLOCATE contact_cursor -- Помилка, оскільки LOCAL видалив курсор.
GO
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Створити БД DreamHomeDB, яка складається з 5 таблиць: Staff, Property_for_Rent, Renter, Qwner, Viewing.

Branch	(Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff	(Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent	(Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter	(Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner	(Ono, FName, LName, Address, Tel_No)
Viewing	(Rno, Pno, Date, Comment)

Завдання 3

Створити вибірку за допомогою вкладених запитів для наступних завдань:

1) Скласти список персоналу, який працює у відділенні компанії, розташованому на вулиці 'Main St' в будинку 163 (*рекомендується використання підзапиту з перевіркою на рівність*).

2) Скласти список всіх співробітників, що мають зарплату вище середньої, вказавши те, наскільки їх зарплата перевищує середню зарплату по підприємству (*рекомендується використання підзапитів з узагальнюючою функцією*).

3) Скласти список об'єктів, що здаються в оренду, за які відповідають працівники відділення компанії, розташованого на вулиці 'Main st' в будинку 168 (*рекомендується використання предиката IN, який визначає, чи збігається вказане значення з одним зі значень, що містяться у вкладеному запиті або списку*).

4) Знайти всіх працівників, чия зарплата перевищує зарплату хоча б одного працівника відділення компанії під номером 'B3' (*рекомендується використання ключових слів ANY і SOME, які порівнює скалярне значення з набором значень, що складається з одного стовпця*).

5) Знайти всіх працівників, чия заробітна плата більше заробітної плати будь-якого працівника відділення компанії під номером 'B3' (*рекомендується використання ключового слова ALL, яке порівнює скалярне значення з набором значень, що складається з одного стовпця*).

Завдання 4

Зайдіть на сайт MSDN.

Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп'ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Вибірки з використанням вкладених запитів:

1. Вибрати книгу з фізики, яка найбільш популярна у читачів (максмальне число видач за рік).
2. Вибрати читачів, які мають заборгованість більше 4 місяців.
3. Визначити книгу, яка була найбільш популярною взимку 2020 року.
4. Визначити читачів, у яких на руках знаходяться книги видавництва «Освіта і

Варіант 2

Вибірки з використанням вкладених запитів:

1. Вибрати викладачів, у яких найбільша кількість пар восени 2020 року.
2. Визначити можливі «накладки» у викладачів в розкладі.
3. Вивести розклад занять викладача «Добролюбова М.В.» на листопад 2020 року.
4. Визначити для кожної групи кількість іспитів і заліків.

Варіант 3

Вибірки з використанням вкладених запитів:

1. Визначити виріб, з найбільшою тривалістю виробництва.
2. Вивести список устаткування, яке потребує заміни в найближчі півроку.
3. Вивести список виробів, яких виходить найбільше з одиниці об'єму сталі.
4. Вивести середній термін придатності обладнання.

Варіант 4

Вибірки з використанням вкладених запитів:

1. Вибрати прізвище того механіка, який частіше за всіх працює з новими автомобілями.
2. Вибрати прізвище господаря, що найчастіше відвідує майстерню і має автомобіль марки «Мерседес».
3. Визначити тих власників автомобілів, яких завжди обслуговує один і той самий механік. Вивести прізвища механіка і його постійного клієнта.
4. Для кожної марки автомобіля вибрати механіка, найчастіше обслуговуючого цю марку.

Варіант 5

Вибірки з використанням вкладених запитів:

1. Вибрати групу з найбільшою тривалістю сесії.
2. Визначити викладача, який в сесію приймає заліки у найбільшого числа студентів.
3. Визначити, відсоток іспитів, що здаються групою «ПА-01мп», від загального числа предметів для кожної категорії.
4. Визначити, чи не здає будь-яка група два іспити або заліки в один день.

Варіант 6

Вибірки з використанням вкладених запитів:

1. Визначити ті роботи в 2020 році, за якими дата завершення перевищує крайній термін, тобто проєкт прострочений.
2. Визначити максимальну кількість проєктів у одного працівника в 2019 році.
3. Визначити ті роботи, які до дати завершення були виконані не більше, ніж на 50 %.
4. Визначити 5 співробітників, що виконують найбільший обсяг робіт.

Варіант 7

Вибірки з використанням вкладених запитів:

1. Визначити ті випадки, коли пацієнт лікувався менше місяця.
2. Вивести імена тих лікарів, які працюють виключно з інвалідами.
3. Визначити відсоток смертності від захворювання «карієс».
4. Вивести імена тих пацієнтів, які лікувалися у лікарів усіх спеціальностей.

Варіант 8

Вибірки з використанням вкладених запитів:

1. Вибрати найбільш популярний тариф для кожного оператора.
2. Визначити сумарну заборгованість по кожному з тарифів.
3. Підрахувати, скільки є незареєстрованих номерів у кожного з операторів.
4. Вибрати список абонентів оператора «Київстар», що мають заборгованість більше 3000 грн.

Варіант 9

Вибірки з використанням вкладених запитів:

1. Вивести таблицю розподілу місць в змаганні «відкритий чемпіонат» в місті «Васюки» з «шахів» в 2021 році.
2. Визначити спортсменів, які мають як олімпійський, так і світовий рекорд.
3. Вивести список спортсменів, що беруть участь більш ніж у 8 змаганнях на рік в порядку зростання середнього місця на цих змаганнях.
4. Визначити найкращий показник спортсмена «Фуркад» у виді спорту «біатлон» і різницю зі світовим і олімпійським рекордами.

Варіант 10

Вибірки з використанням вкладених запитів:

1. Вивести підприємства, які провели в поточному році не більше однієї операції.
2. Визначити підприємства, вид діяльності якого є провідним в поставці товару «банан».
3. Визначити випадки, коли були продані товари із терміном придатності не більше тижня.
4. Вивести список продуктів, для яких закупівельна ціна, як правило, нижче собівартості виробника.

Варіант 11

Вибірки з використанням вкладених запитів:

1. Визначити найприбутковіший засіб пересування (відношення середня плата за проїзд / середні витрати на паливо).
2. Визначити середнє відношення числа тролейбусів на маршруті до числа зупинок.
3. Вивести маршрути машини «Газель» в порядку спадання їх довжини.
4. Визначити очікуваний прибуток фірми за день.

Варіант 12

Вибірки з використанням вкладених запитів:

1. Розташувати країни за багатонаціональністю в порядку спадання.
2. Підрахувати чисельність населення монархій.
3. Визначити тип управління тієї країни, де проживає найбільше представників національності «вірмени».
4. Вибрати список національностей в країнах з анархічним ладом.

Варіант 13

Вибірки з використанням вкладених запитів:

1. Визначити суму місячної оплати для квартири № 50 в будинку 302 по Садовій вулиці.
2. Визначити заборгованість по оплаті «доставка сміття» Кота Романа Вікторовича.
3. Визначити загальну суму виплат для квартир по вулиці Боткіна.
4. Вибрати список власників, які мають заборгованості понад рік.

Варіант 14

Вибірки з використанням вкладених запитів:

1. Визначити середньорозрахунковий час польоту для літака міжнародних перевезень «Boing-24».
2. Вибрати марку літака, яка найчастіше літає на внутрішніх рейсах.
3. Вибрати маршрут / маршрути, за якими найчастіше літають рейси, заповнені менш, ніж на 70 %.
4. Визначити наявність вільних місць на рейс №354 23 серпня 2019 року.

Варіант 15

Вибірки з використанням вкладених запитів:

1. Визначити фірму, яка надає найбільшу гарантію на ноутбуки на базі процесора «Pentium Centrino».
2. Вибрати модель з найбільшим розміром екрану, яка випускається в місті «Дніпро».
3. Визначити фірму, що здійснила обсяг продажів на найбільшу суму за 2020.
4. Вибрати міста, в яких випускаються ноутбуки на базі «ASUS».

Варіант 16

Вибірки з використанням вкладених запитів:

1. Вибрати кількість студентів на кожному факультеті.
2. Вибрати середню величину стипендії для кожної групи факультету «ПБФ».
3. Для кожного факультету вибрати список груп із зазначенням чисельності студентів в кожній групі.
4. Вибрати прізвища і величину стипендії студентів з міста «Житомир».

Варіант 17

Вибірки з використанням вкладених запитів:

1. Визначити середню площу, що припадає на одну особину в вольєрі «мавпятник».
2. Вибрати випадки розміщення тваринного виду «крокодил» в приміщенні з водоймою, але площею, менше 3 квадратних метрів.
3. Визначити загальну чисельність представників сімейства «вовчі» в зоопарку.
4. Вивести середню тривалість життя для мешканців «тераріуму».

Варіант 18

Вибірки з використанням вкладених запитів:

1. Вибрати турнір з найбільшою кількістю гросмейстерів.
2. Вибрати ті турніри, де жодне призове місце не зайняв представник країни-господаря турніру.
3. Вибрати тих шахістів, які виграли не менше трьох турнірів протягом 2020 року.
4. Визначити турніри, в яких учасник з найвищим рейтингом посів останнє місце.

Варіант 19

Вибірки з використанням вкладених запитів:

1. Вибрати імена капітанів, що приводили кораблі в місто «Варна» взимку 2020/21 років.
2. Визначити, коли і з якою метою судно «Електрон» відвідувало порт «Трансе».
3. Визначити, який прибуток отримав порт міста Миколаїв від кораблів з Туреччини.
4. Визначити, з якою метою найчастіше заходять кораблі в порт «Тортуга».

Варіант 20

Вибірки з використанням вкладених запитів:

1. Визначити водія, який найчастіше виконує далекі перевезення (> 500 км).
2. Визначити середню вантажопідйомність рейсів з Києва.
3. Визначити загальну витрату палива у перевезеннях, що здійснені у 2019.
4. Визначити середній час поїздки для водія «Шевченко».

Варіант 21

Вибірки з використанням вкладених запитів:

1. Вибрати вчених, сумарна тривалість виступів яких на конференціях в Україні перевищує 5 годин.
2. Вибрати конференцію, тематика якої не збігається зі спеціалізацією більшості її учасників.
3. Визначити, в якій конференції брало участь більше всього докторів наук.
4. Визначити середній відсоток іноземних вчених на конференціях, що проходять в місті «Київ».

Варіант 22

Вибірки з використанням вкладених запитів:

1. Визначити сферу застосування, в якій інстальовано найбільшу кількість продуктів.
2. Визначити витрати на придбання / модифікацію програмного забезпечення, зроблені покупцем «UkrEnergo» в 2020 році.
3. Визначити тих покупців, для яких сумарна вартість всіх модифікацій продукту перевищила вартість ліцензії на продукт останньої версії.
4. Вибрати список продуктів типу «серверні операційні системи», в порядку спадання їх популярності.

Варіант 23

Вибірки з використанням вкладених запитів:

1. Визначити роль актриси, яка найчастіше грає роль «технічки-шпигунки».
2. Підрахувати кількість п'єс для кожного режисера.
3. Вибрати список п'єс, в яких стать виконавців ролей змінювалась.
4. Вибрати список акторів, які зайняті більш ніж в трьох постановках одночасно.

Варіант 24

Вибірки з використанням вкладених запитів:

1. Визначити, якою хворобою найчастіше страждають клієнти аптеки «Еко-аптека».
2. Визначити, яких збитків зазнає аптека 999, якщо протягом місяця не реалізує всі ліки, у яких закінчується термін придатності.
3. Визначити, в яких аптеках найнижча середня ціна на болезаспокійливі.
4. Вибрати кількість таблеток, на яких доведеться змінити дату закінчення терміну придатності, якщо вони не будуть реалізовані протягом тижня.

Варіант 25

Вибірки з використанням вкладених запитів:

1. Для клієнта «Петренко» вибрати меню з чотирьох страв (перше, друге, закуска і компот) так, щоб калорійність його була від 600 до 800 кілокалорій.
2. Вибрати найдешевше блюдо, яке задовольняє переваги клієнта «Олена Юсупова».
3. Вибрати середній щоденний дохід їдальні.
4. Вибрати найбільш споживану страву в березні.

Варіант 26

Вибірки з використанням вкладених запитів:

1. Визначити для кожного пілота кількість етапів, кількість набраних очок за системою (10-8-6-5-4-3-2-1) і розташувати в порядку спадання.
2. Визначити для кожної команди середній відсоток проходження етапу її пілотами (відношення пройдених кіл до загальної довжини гонки).
3. Вибрати етапи, на яких більшість очок набирали пілоти на гумі «Michelin».
4. Для кожного пілота визначити середню швидкість на етапі за рік.

Варіант 27

Вибірки з використанням вкладених запитів:

1. Визначити фірму, яка надає найбільший спектр деталей, кількість цих деталей і середня гарантія.
2. Визначити 10 деталей, які дають найбільшу зміну швидкості.
3. Для кожного власника визначити витрати на тюнінг автомобілів.
4. Визначити найбільш оздоблені автомобілі для кожної марки і їх параметри.

Варіант 28

Вибірки з використанням вкладених запитів:

1. Вибрати список археологів, їх кваліфікацію, кількість знахідок за останній рік з епохи середньовіччя.
2. Вибрати назву предмета, епоху, кількість знахідок таких предметів, археолога, який переважно знаходив ці предмети.
3. Вивести повну інформацію про предмети, знайдені в районі «Каїра» епохи «Древньоєгипетська».
4. Вивести середню вартість знахідок для кожного археолога за час їх роботи.

Варіант 29

Вибірки з використанням вкладених запитів:

1. Вибрати відпочиваючого, який найбільше число раз відвідав «Єгипет», вивести цю кількість і його витрати на поїздки.
2. Вибрати моря і назви готелів, які відвідав «Коновалов» в минулому році.
3. Вибрати всі можливі варіанти не більше двох тур-поїздок на суму 3000 $, так щоб відпочинок пройшов на «Середземному» морі.
4. Вибрати кількість днів відпочинку і кількість місць відпочинку в минулому році для кожного клієнта фірми.

Варіант 30

Вибірки з використанням вкладених запитів:

1. Вибрати канали, які транслюються на найбільшу територію і за специфікою є новинними.
2. Вибрати супутники, радіус їх орбіт і розташувати в порядку спадання за кількістю каналів на них.
3. Вибрати супутники, що не ведуть мовлення жодного каналу своєї країни.
4. Вибрати телекомпанії, що ведуть мовлення на Україну каналів розважальної специфіки.

Рекомендовані ресурси

MSDN: Вкладені запити (SQL Server)
https://docs.microsoft.com/ru-ru/sql/relational-databases/performance/subqueries?view=sql-server-ver15

MSDN: Тимчасові таблиці
https://docs.microsoft.com/ru-ru/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-temporary

MSDN: Курсори
https://docs.microsoft.com/RU-RU/sql/relational-databases/cursors?view=sql-server-ver15

Контрольні питання

1. Що таке вкладений запит?
2. Для чого використовуються вкладені запити?
3. В яких випадках краще використовувати вкладені запити, а не з’єднання?
4. Що таке зв’язані підзапити?
5. Що таке тимчасові таблиці?
6. Що таке курсори?
7. З якою метою використовують курсори?

Комп'ютерний практикум 7

Індексування та представлення

Мета роботи: розгляд питань теорії збалансованих дерев, індексів та представлень.

Теоретичні відомості

- ***Індекс*** – це список, який зазвичай розміщують в алфавітному порядку перелічення певних даних.
- Індекс є структурою на диску, яка пов'язана з таблицею (або представленням) і прискорює отримання записів з таблиці (або представлення), оскільки дані з певного запису отримуються без перебору всіх записів в таблиці.
- Індекс містить ключі, побудовані з одного або декількох полів в таблиці (або представлення). Ці ключі зберігаються у вигляді структури збалансованого дерева.
- Індекс займає додаткове місце в пам'яті і може мати зворотний ефект – не підвищення, а зниження продуктивності.
- Індекси поділяються на кластеризовані та некластиризовані. В свою чергу некластеризовані індекси поділяються на задані на невпорядкованій таблиці та задані на кластеризованій таблиці.
- Для створення індексу використовують оператор CREATE INDEX:
  CREATE [UNIQUE] INDEX index_name
  ON table_name (column [ASC | DESC] [, . . .])
  Тут UNIQUE – унікальність значень ключа індексу буде автоматично підтримуватися системою.
- PRIMARY KEY за замовчуванням створює унікальний кластеризований індекс, але можна створити первинний ключ, який задасть некластеризований індекс. Якщо у таблиці вже є кластеризований індекс, то створиться некластеризований індекс.

```
CREATE TABLE Persons (
    Id INTEGER PRIMARY KEY, -- унікальний кластеризований індекс
    Name VARCHAR(255)
);

CREATE TABLE Persons (
    Id INTEGER PRIMARY KEY NONCLUSTERED, -- унікальний некластеризований індекс (з явним вказуванням на це)
    Name VARCHAR(255)
);
```

- Обмеження UNIQUE за замовчуванням створює унікальний некластеризований індекс, але можна створити UNIQUE, який задасть кластеризований індекс.

```
CREATE TABLE Persons (
    Id INTEGER,
    Name VARCHAR(255) UNIQUE -- унікальний некластеризований індекс
);

CREATE TABLE Persons (
    Id INTEGER,
    Name VARCHAR(255) UNIQUE CLUSTERED -- унікальний кластеризований індекс (з явним вказуванням на це)
);
```

- Створюючи індекси не через PRIMARY KEY або UNIQUE, можна створити неунікальний індекс (кластеризований або некластеризований). За замовчуванням створюється неунікальний некластеризований індекс.

```
CREATE INDEX index_name -- неунікальний некластеризований індекс
ON table_name (column_name)

CREATE UNIQUE INDEX index_name -- унікальний некластеризований індекс
ON table_name (column_name)

CREATE CLUSTERED INDEX index_name -- неунікальний кластеризований індекс
ON table_name (column_name)

CREATE UNIQUE CLUSTERED INDEX index_name -- унікальний кластеризований індекс
ON table_name (column_name)
```

- Для видалення індексу використовують оператор DROP INDEX:

```
DROP INDEX index_name
```

- ***Сторінка*** – це основна одиниця збереження даних в SQL Server. Розмір сторінки становить 8 КБ. Кожна сторінка починається з 96-байтового заголовка, який використовується для збереження системних даних про сторінку: номер сторінки, тип сторінки, об’єм вільного місця на сторінці і ідентифікатор об'єкта, якому належить сторінка.
- Сторінки поділяються на: сторінки даних (крім великих типів даних типу varchar(max), varbinary(max) тощо), сторінки індексів, сторінки тексту/зображення (для даних типу varchar(max), varbinary(max) тощо), Index Allocation Map тощо.
- ***Екстент*** – це основна одиниця організації простору. Екстент складається з восьми безперервних сторінок (64 КБ). Кожна з восьми сторінок в змішаному екстенті може знаходитись у володінні різних

об'єктів. Однорідні екстенти належать одному об'єкту; всі вісім сторінок в кластері можуть бути використані тільки цим об'єктом.

- ***Купа*** – це таблиця, у якої немає кластеризованого індексу. Записи не мають певного порядку зберігання. Сторінки не пов'язані в двонапрямлений зв'язаний список.
- ***Index Allocation Map (IAM)*** – це сторінка, що містить карту всіх екстентів, які містять дані зазначеної таблиці.
- Для того, щоб знайти дані в купі, SQL Server використовує IAM. Він робить запит до системної таблиці sysindexes для знаходження FirstIAM сторінки. Послідовність записів, отриманих кінцевим користувачем залежить від того, в якому порядку дані зберігаються в IAM, а не від порядку вставки записів в таблицю. При виконанні запиту до таблиці, у якої немає індексів, виконується її сканування. SQL Server не знає, що в таблиці існує тільки один запис, який задовольняє умові і буде переглядати всі екстенти (які містять дані зазначеної таблиці), всі сторінки, всі записи.
- Індекс складається з набору сторінок, вузлів індексу і зберігається у вигляді B-дерева (B-tree), де «B» означає збалансоване. Кореневий вузол є «точкою входу» в індекс, складається лише з однієї сторінки і містить покажчики на ключі наступних рівнів індексу. Листя індексу можуть містити як самі дані таблиці (кластеризований індекс), так і просто вказівник на рядки з даними в таблиці (некластеризований індекс).
- ***Кластеризований індекс*** – це індекс, який реалізується у вигляді збалансованого дерева, яке підтримує швидке отримання рядків за значеннями ключа кластеризованого індекса. Кластеризований індекс зберігає реальні рядки даних в листі індексу. Важливою характеристикою кластеризованого індексу є те, що всі значення відсортовані в певному порядку (або зростання, або спадання). Таким чином, таблиця (або представлення) може мати тільки один кластеризований індекс. В SQL Server кластеризований індекс є унікальним індексом за визначенням. Якщо існують записи з однаковими значеннями, SQL Server робить їх унікальними, додаючи номери з внутрішнього (невидимого зовні) лічильника.
- ***Кластеризована таблиця*** – це таблиця, дані в якій відсортовані за певним полем або групою полів, на яких заданий кластеризований індекс.
- ***Некластеризований індекс*** – це індекс, який на листковому рівні містить відсортовані значення індексованого стовпця і row locator – вказівник на інші дані. Якщо у таблиці є кластеризований індекс (кластеризована таблиця), то row locator'ом є ключ кластеризованого індексу, якщо кластеризованого індексу немає (таблиця-купа), то row locator – це row ID (вказівник на певний рядок даних в певній таблиці).

- Кластеризовані ключі повинни бути мінімально короткими. Кожен некластеризований індекс буде використовувати значення кластеризованого індексу (якщо останній визначений). Отже збільшення розміру кластеризованого індексу призводить до збільшення пам'яті для всіх не кластеризованих індексів.
- Кластеризований індекс необхідно створювати першим, оскільки якщо вже будуть некластеризовані індекси, то всім їм потрібно буде перебудуватися – на листковому рівні замість row ID вказівником на інші дані повинен стати ключ кластеризованого індексу.
- При роботі з індексами зазвичай дотримуються наступних рекомендацій:
  - для кластеризованого індексу найбільш підходить поле, яке є унікальним, коротким, яке не буде оновлюватися і яке не підтримує NULL значень – ось чому первинний ключ часто використовується як кластеризований індекс;
  - доцільно створювати індекси на поля, по яким відбувається пошук, сортування, групування, з'єднання таблиць;
  - не доцільно створювати індекси на полях, які рідко використовуються в запитах;
  - не доцільно створювати індекси на полях, які містять кілька унікальних значень (наприклад, поле, яке містить лише значення чоловічої або жіночої статі) – чим більше дублікатів в полі, тим гірше працює індекс, чим більше унікальних значень, тим вище працездатність індексу (за можливістю використовуйте унікальний індекс);
  - необхідно пам’ятати, що для невеликих таблиць пошук за індексом може зайняти більше часу, ніж просте сканування всіх рядків;
  - доцільно використовувати якомога менше індексів для таблиць які часто оновлюються.
- ***Представлення (views)*** – це об'єкти бази даних, який завжди створюється на основі однієї або більше базових таблиць (або інших представлень), використовуючи інформацію метаданих, або ж іншими словами – це спосіб виведення обмеженого набору полів з реальної таблиці у вигляді віртуальної таблиці (на відміну від базових таблиць, представлення за замовчуванням не існують фізично, тобто їх вміст не зберігається на диску). Інформація метаданих (включаючи ім'я представлення і спосіб отримання рядків з базових таблиць) – все, що зберігається фізично для представлення.
- Для створення представлень використовують оператор CREATE VIEW:
  CREATE VIEW view name [(column name [, . . .])]
  AS subselect [WITH (CASCADED | LOCAL] CHECK OPTION]

Тут параметр column_name відповідає за необов'язкову можливість присвоєння власного імені кожному із стовпців представлення; параметр subselect задає визначальний запит із зазначенням полів в результуючій таблиці запиту; фраза WITH CHECK OPTION гарантує, що в тих випадках, коли рядок даних не задовольняє умові, зазначеній в реченні WHERE визначального запиту представлення, вона не буде додана в його базову таблицю.

- Для видалення представлень використовують оператор DROP VIEW:
  DROP VIEW view name [RESTRICT | CASCADE]
  Тут CASCADE – для видалення всіх пов'язаних з представленням або залежних від нього об'єктів; RESTRICT – для блокування видалення за наявності об’єктів, що залежать від існування видаляємого представлення (за замовчуванням приймається значення RESTRICT).
- Представлення використовують:
  - в якості механізму безпеки, що дозволяє користувачам звертатися до даних через представлення, але не надаючи їм дозволів на безпосередній доступ до базових таблиць;
  - для приховування подробиць складних запитів;
  - для обмеження на вставку або оновлення значень деяким діапазоном.
- Умови для зміни даних базової таблиці через представлення:
  - будь-які зміни (UPDATE, INSERT та DELETE) повинні посилатися на поля тільки однієї базової таблиці;
  - поля, які змінюються в представленні, повинні безпосередньо посилатися на дані полів базової таблиці – поля можна сформувати будь-яким іншим чином, в тому числі: агрегованими функціями або на основі обчислення;
  - GROUP BY, HAVING і DISTINCT не впливають на поля, які змінюються;
  - TOP не використовується з WITH CHECK OPTION.

Практичні приклади

```
USE ShopDB
GO
DROP TABLE BTree;
GO
CREATE TABLE BTree
(
numbers int ,
word varchar(25)
)
GO

INSERT BTree
VALUES
(1,'one'),
(2,'two'),
(3,'three'),
(4,'four'),
(6,'six'),
```

```
(7,'seven'),
(8,'eight'),
(9,'nine');

SELECT * FROM BTree;

INSERT BTree
VALUES (5,'five');

SELECT * FROM BTree WHERE numbers IN (5,6);
-------------------------------------------------
DROP TABLE BTree;
GO

CREATE TABLE BTree
(
numbers int primary key, -- первинний ключ, додає кластеризований індекс
word varchar(25)
)
--SET STATISTICS IO ON
INSERT BTree
VALUES
(1,'one'),
(2,'two'),
(3,'three'),
(4,'four'),
(6,'six'),
(7,'seven'),
(8,'eight'),
(9,'nine');

SELECT * FROM BTree;  --SELECT * FROM BTree WHERE numbers = 2;SELECT * FROM BTree WHERE word =
'one';

SELECT * FROM sys.indexes
WHERE object_id = (SELECT object_id FROM sys.tables
                                    WHERE name = 'BTree')

INSERT BTree
VALUES (5,'five');

SELECT * FROM BTree;
-------------------------------------------------
DROP TABLE BTree;
GO

CREATE TABLE BTree
(
numbers int ,
word varchar(25)primary key -- кластеризований індекс
)

INSERT BTree
VALUES
(1,'one'),
(2,'two'),
(3,'three'),
(4,'four'),
(6,'six'),
(7,'seven'),
(8,'eight'),
(9,'nine');

SELECT * FROM BTree;

INSERT BTree
VALUES (5,'five');

SELECT * FROM BTree;
-------------------------------------------------
DROP TABLE BTree;
GO
CREATE TABLE BTree
(
numbers int unique, -- некластеризований індекс на невпорядкованій таблиці (дані зберігаються не
на листковому рівні дерева)
word varchar(25)
```

```
)

INSERT BTree
VALUES
(1,'one'),
(2,'two'),
(3,'three'),
(4,'four'),
(6,'six'),
(7,'seven'),
(8,'eight'),
(9,'nine');

SELECT * FROM BTree;

INSERT BTree
VALUES (5,'five');

SELECT * FROM BTree WHERE numbers = 9; --word = 'one';
-------------------------------------------------
DROP TABLE BTree;
GO

CREATE TABLE BTree
(
numbers int primary key, -- кластеризований індекс
word varchar(25)unique   -- некластеризований індекс на кластеризованій таблиці
)

INSERT BTree
VALUES
(1,'one'),
(2,'two'),
(3,'three'),
(4,'four'),
(6,'six'),
(7,'seven'),
(8,'eight'),
(9,'nine');

SELECT * FROM BTree;

INSERT BTree
VALUES (5,'five');

SELECT * FROM BTree where  word = 'five'; -- пошук по некластеризованому індексу -- word in
('five', 'one');
SELECT * FROM BTree where numbers = 3;   -- пошук по кластеризованому індексу -- numbers in
(3,5);
-------------------------------------------------
USE ShopDB
GO


DROP TABLE OrderDetails;
GO
DROP TABLE Orders;
GO
DROP TABLE Customers
GO

CREATE TABLE Customers
(
        CustumerNo int NOT NULL,
        CustumerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        Address2 varchar(25) NOT NULL,
        City      varchar(15) NOT NULL,
        State char(2) NOT NULL,
        Zip varchar(10) NOT NULL,
        Contact varchar(25) NOT NULL,
        Phone char(12),
        FedIDNo  varchar(10) NOT NULL,
        DateInSystem smalldatetime NOT NULL
)
GO
```

```
INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'Alex1', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE());
GO

SELECT * FROM Customers
WHERE CustumerName = 'Alex';
-------------------------------------------------------------------------------------------------
---

DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustumerNo int NOT NULL PRIMARY KEY,	-- кластеризований індекс
	CustumerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	Address2 varchar(25) NOT NULL,
	City	varchar(15) NOT NULL,
	State char(2) NOT NULL,
	Zip varchar(10) NOT NULL,
	Contact varchar(25) NOT NULL,
	Phone char(12),
	FedIDNo  varchar(10) NOT NULL,
	DateInSystem smalldatetime NOT NULL
)
GO

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE());

SELECT * FROM Customers; -- для пошуку використовуємо кластеризований індекс
-------------------------------------------------------------------------------------------------
---
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustumerNo int NOT NULL,
	CustumerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	Address2 varchar(25) NOT NULL,
	City	varchar(15) NOT NULL,
	State char(2) NOT NULL,
	Zip varchar(10) NOT NULL,
	Contact varchar(25) NOT NULL,
	Phone char(12) UNIQUE, -- некластеризований індекс
	FedIDNo  varchar(10) NOT NULL,
	DateInSystem smalldatetime NOT NULL
)
GO

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE());
GO

SELECT * FROM Customers;
SELECT Phone FROM Customers
```

```
       WHERE Phone = '(093)2221212'; -- для пошуку використовуємо некластеризований індекс
-------------------------------------------------------------------------------------------------
---
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
       CustumerNo int IDENTITY NOT NULL
              PRIMARY KEY,    -- кластеризований індекс
       CustumerName varchar(25) NOT NULL,
       Address1 varchar(25) NOT NULL,
       Address2 varchar(25) NOT NULL,
       City      varchar(15) NOT NULL,
       State char(2) NOT NULL,
       Zip varchar(10) NOT NULL,
       Contact varchar(25) NOT NULL,
       Phone char(12) UNIQUE, -- некластеризований індекс
       FedIDNo  varchar(10) NOT NULL,
       DateInSystem smalldatetime NOT NULL
)
GO

INSERT INTO Customers
( CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo, DateInSystem)
VALUES
('Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
('Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE());

SELECT * FROM Customers; -- для пошуку використовуємо лише кластеризований індекс

SELECT * FROM Customers                          -- для пошуку використовуємо кластеризований
       WHERE Phone = '(093)2221212'; -- та некластеризированый індекси !!!
------------------------ Кластеризований індекс ----------------------------------------------
---
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
       CustumerNo int NOT NULL,
       CustumerName varchar(25) NOT NULL,
       Address1 varchar(25) NOT NULL,
       Address2 varchar(25) NOT NULL,
       City      varchar(15) NOT NULL,
       State char(2) NOT NULL,
       Zip varchar(10) NOT NULL,
       Contact varchar(25) NOT NULL,
       Phone char(12),
       FedIDNo  varchar(10) NOT NULL,
       DateInSystem smalldatetime NOT NULL
)
GO

CREATE CLUSTERED INDEX PK_IND1 -- створення власного кластеризованого індексу
ON Customers(CustumerNo);
GO

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(1,'Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE()); -- CustumerNo = 1 як і в першому рядку
GO

SELECT * FROM Customers;-- для пошуку використовуємо кластеризований індекс
----------------------Кластеризований індекс ----------------------------------------------------
---
DROP TABLE Customers;
GO

CREATE TABLE Customers
```

```
(
	CustumerNo int NOT NULL,
	CustumerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	Address2 varchar(25) NOT NULL,
	City	 varchar(15) NOT NULL,
	State char(2) NOT NULL,
	Zip varchar(10) NOT NULL,
	Contact varchar(25) NOT NULL,
	Phone char(12) Primary key,
	FedIDNo  varchar(10) NOT NULL,
	DateInSystem smalldatetime NOT NULL
)
GO

CREATE UNIQUE CLUSTERED INDEX PK_IND1 -- створення власного кластеризованого індексу
ON Customers(CustumerNo);
GO

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(1,'Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE()); -- CustumerNo = 1 буде помилка
GO

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE()); --CustumerNo повинен бути унікальним для кожного рядка, якщо CREATE UNIQUE
CLUSTERED INDEX
GO

SELECT * FROM Customers; --для пошуку використовуємо кластеризований індекс
------------------ Створення некластеризованого індекса на невпорядкованій таблиці --------------
---
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustumerNo int NOT NULL,
	CustumerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	Address2 varchar(25) NOT NULL,
	City	 varchar(15) NOT NULL,
	State char(2) NOT NULL,
	Zip varchar(10) NOT NULL,
	Contact varchar(25) NOT NULL,
	Phone char(12),
	FedIDNo  varchar(10) NOT NULL,
	DateInSystem smalldatetime NOT NULL
)
GO

CREATE UNIQUE NONCLUSTERED INDEX NC_IND1 -- створення власного некластеризованого  індекса на
невпорядкованій таблиці
ON Customers(Phone);
GO
INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1111111', 'qweq',
GETDATE()),
(2,'Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2222222',
'qwq2', GETDATE()); --Phone повинен бути унікальним для кожного рядка, якщо CREATE UNIQUE
NONCLUSTERED INDEX
GO
SELECT * FROM Customers;
```

```
SELECT * FROM Customers
	WHERE Phone = '(093)2222222';

----------------Створення некластеризованого індекса на кластеризованій таблиці------------------
---
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustumerNo int NOT NULL,
	CustumerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	Address2 varchar(25) NOT NULL,
	City	 varchar(15) NOT NULL,
	State char(2) NOT NULL,
	Zip varchar(10) NOT NULL,
	Contact varchar(25) NOT NULL,
	Phone char(12),
	FedIDNo  varchar(10) NOT NULL,
	DateInSystem smalldatetime NOT NULL
)
GO

CREATE UNIQUE CLUSTERED INDEX PK_IND1 -- створення власного кластеризованого індекса
ON Customers(CustumerNo);
GO

CREATE UNIQUE NONCLUSTERED INDEX NC_IND1 -- створення власного некластеризованого індекса на
кластеризованій таблиці
ON Customers(Phone);
GO

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'Alex2', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE()); --CustumerNo повинен бути унікальним для кожного рядка, якщо CREATE UNIQUE
CLUSTERED INDEX
GO

SELECT * FROM Customers; -- для пошука використовуємо кластеризований індекс

SELECT * FROM Customers                         -- для пошука використовуємо кластеризований
	WHERE Phone = '(093)2221212'; --  і некластеризированый індекси !!!
-------------------------------------------------------------
	 /* Зміна індексу ALTER INDEX */
-------------------------------------------------------------
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
	CustumerNo int NOT NULL,
	CustumerName varchar(25) NOT NULL,
	Address1 varchar(25) NOT NULL,
	Address2 varchar(25) NOT NULL,
	City	 varchar(15) NOT NULL,
	State char(2) NOT NULL,
	Zip varchar(10) NOT NULL,
	Contact varchar(25) NOT NULL,
	Phone char(12),
	FedIDNo  varchar(10) NOT NULL,
	DateInSystem smalldatetime NOT NULL
)
GO

CREATE UNIQUE INDEX PK_IND1 --якщо включена опція IGNORE_DUP_KEY і немає обмеження UNIQUE буде
попередження (можлива вставка однакових індексів Alex та alex)
ON Customers(CustumerName)
WITH
(
IGNORE_DUP_KEY = ON -- включаємо ігнорування різниці між великими та малими літерами
);
```

```
GO
INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'alex', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE()); -- не буде помилки
GO

SELECT * FROM Customers

ALTER INDEX PK_IND1 -- зміна індексу PK_IND1,
ON Customers          -- заданого на таблиці Customers
REBUILD WITH
        (
                IGNORE_DUP_KEY = OFF
        );

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'alex', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE()); -- буде помилка
GO

SELECT * FROM Customers;

----------------------------Видалення індексів-----------------------------------------------------
---
DROP TABLE Customers;
GO

CREATE TABLE Customers
(
        CustumerNo int NOT NULL,
        CustumerName varchar(25) NOT NULL,
        Address1 varchar(25) NOT NULL,
        Address2 varchar(25) NOT NULL,
        City      varchar(15) NOT NULL,
        State char(2) NOT NULL,
        Zip varchar(10) NOT NULL,
        Contact varchar(25) NOT NULL,
        Phone char(12),
        FedIDNo  varchar(10) NOT NULL,
        DateInSystem smalldatetime NOT NULL
)
GO

CREATE UNIQUE INDEX PK_IND1 -- якщо включена опція IGNORE_DUP_KEY і немає обмеження UNIQUE буде
попередження (можлива вставка однакових індексів Alex та alex)
ON Customers(CustumerName)
WITH
(
IGNORE_DUP_KEY = ON -- включаємо ігнорування різниці між великими та малими літерами
);
GO

INSERT INTO Customers
( CustumerNo, CustumerName, Address1, Address2, City, State, Zip, Contact, Phone, FedIDNo,
DateInSystem)
VALUES
(1,'Alex', 'NewSTR', 'NewSTR2', 'City', 'NS', 'NewZip', 'dfgjs@mail.ru', '(093)1231212', 'qweq',
GETDATE()),
(2,'alex', 'NewSTR2', 'NewSTR22', 'City2', 'SN', 'NewZip2', 'dfg2@mail.ru', '(093)2221212',
'qwq2', GETDATE()); -- якщо включена опція IGNORE_DUP_KEY і є обмеження UNIQUE даний рядок не
запишеться
GO

SELECT * FROM Customers WHERE CustumerName = 'alex'; -- для пошуку використовуємо
некластеризований індекс

DROP INDEX Customers.PK_IND1; -- DROP INDEX PK_IND1 ON Customers;
```

```
SELECT * FROM Customers WHERE CustumerName = 'alex'; -- для пошуку не використовуємо індекс
-------------------------------------------------------------
USE ShopDB
GO
-- Представлення - це спосіб виведення обмеженого набору стовпців з реальної таблиці у вигляді
віртуальної таблиці.
DROP TABLE InfoPerson
GO

CREATE TABLE InfoPerson
(
	FirstName nvarchar(50) null,
	LastName nvarchar(50)null,
	BirthDate date null,
	AddressLine1 nvarchar(50)  null,
	AddressLine2 nvarchar(50) null,
	EmailAddress nvarchar(50) null,
)
GO

INSERT InfoPerson  -- переносимо дані з однієї таблиці в іншу
SELECT pc.FirstName, pc.LastName, CAST(he.BirthDate as DATE), pa.AddressLine1, pa.AddressLine2,
pea.EmailAddress
		FROM AdventureWorks2019.Person.Person as pc  -- БазаДаних.Схема.Таблиця AS аліас
		 JOIN AdventureWorks2019.HumanResources.Employee as he
			ON pc.BusinessEntityID = he.BusinessEntityID
		 JOIN AdventureWorks2019.Person.BusinessEntityAddress as hea
			ON he.BusinessEntityID = hea.BusinessEntityID
		 JOIN AdventureWorks2019.Person.Address as pa
			ON hea.AddressID = pa.AddressID
		 JOIN AdventureWorks2019.Person.EmailAddress as pea
			ON pea.BusinessEntityID = pc.BusinessEntityID
GO

SELECT * FROM InfoPerson;
GO
-------------------------------------------------------------
	/* Створення представлення - CREATE VIEW */
-------------------------------------------------------------

CREATE VIEW BthPerson   -- створення представлення
AS SELECT FirstName, LastName, BirthDate, EmailAddress
FROM InfoPerson;
GO

SELECT * FROM BthPerson; -- вибірка з представлення

SELECT * FROM InfoPerson; -- вибірка з таблиці
GO

-------------------------------------------------------------

CREATE VIEW BthPerson2
WITH SCHEMABINDING     -- забороняє видалення таблиці, для якої створене представлення
AS SELECT FirstName, LastName, BirthDate, EmailAddress
FROM dbo.InfoPerson;   -- двокомпонентна структура імені таблиці, з якою зв'язане представлення
GO

SELECT * FROM BthPerson2;
DROP TABLE InfoPerson; -- буде помилка
GO

-------------------------------------------------------------

CREATE VIEW BthPrsn2Addres   -- створюємо представлення для вибірки даних за умовою
AS SELECT FirstName, LastName, BirthDate, AddressLine1, AddressLine2
FROM InfoPerson
WHERE AddressLine2 is not null; -- представлення буде виводити інформацію про тих людей, які
мають 2 адреси
GO

SELECT * FROM BthPrsn2Addres; -- вибірка з представлення

SELECT FirstName, LastName, BirthDate, AddressLine1 FROM BthPrsn2Addres; -- як і при вибірці з
таблиці можна задавати певний набір стовпців для виводу
GO
```

```
-- Замість таблиці InfoPerson, в яку записана інформація з інших таблиць,
-- можна створити представлення, при цьому буде зекономлена фізична пам'ять бази даних.

CREATE VIEW InfoPerson2 AS
SELECT pc.FirstName, pc.LastName, he.BirthDate, pa.AddressLine1, pa.AddressLine2,
pea.EmailAddress
              FROM AdventureWorks2019.Person.Person as pc  -- БазаДаних.Схема.Таблиця AS аліас
               JOIN AdventureWorks2019.HumanResources.Employee as he
                     ON pc.BusinessEntityID = he.BusinessEntityID
               JOIN AdventureWorks2019.Person.BusinessEntityAddress as hea
                     ON he.BusinessEntityID = hea.BusinessEntityID
               JOIN AdventureWorks2019.Person.Address as pa
                     ON hea.AddressID = pa.AddressID
               JOIN AdventureWorks2019.Person.EmailAddress as pea
                     ON pea.BusinessEntityID = pc.BusinessEntityID

SELECT * FROM InfoPerson;

SELECT * FROM InfoPerson2;

-------------- Вставка даних за допомогою представлення
SELECT * FROM BthPerson;

INSERT INTO BthPerson
(BirthDate, EmailAddress, FirstName, LastName)
VALUES
('18/11/2013', 'mimimi@ukr.net', 'Marusya', 'Kititop');

-- Вибірка з представлення.
SELECT * FROM BthPerson WHERE LastName = 'Kititop';

-- Вибірка з таблиці.
-- Насправді дані записались в таблицю InfoPerson.
SELECT * FROM InfoPerson WHERE LastName = 'Kititop';
------------------------------------------------------------
-- Оновлення даних за допомогою представлення

UPDATE BthPerson
       SET EmailAddress = 'queen@ukr.net'
       WHERE LastName = 'Kititop';

SELECT * FROM BthPerson
       WHERE LastName = 'Kititop';

SELECT * FROM InfoPerson
       WHERE LastName = 'Kititop';

-- Також дані можна оновлювати і в таблиці
UPDATE InfoPerson
SET EmailAddress = 'mimimiQueen@ukr.net'
       WHERE LastName='Kititop';

SELECT * FROM BthPerson
       WHERE LastName='Kititop';

SELECT * FROM InfoPerson
       WHERE LastName='Kititop';

-----------------------Зміна представлень--------------
SELECT * FROM BthPerson;

ALTER VIEW BthPerson  -- зміна представлення
AS  SELECT FirstName, LastName, BirthDate, EmailAddress , AddressLine1
FROM InfoPerson;
GO

SELECT * FROM BthPerson;
---------------------- Видалення представлень ---------------
DROP VIEW BthPerson;

-- Представлення більше не    має
SELECT * FROM BthPerson
       WHERE LastName = 'Kititop';

-- Дані, які додавались через представлення залишаються в таблиці
SELECT * FROM InfoPerson
       WHERE LastName = 'Kititop';
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Зайти в створену раніше БД DreamHomeDB, яка складається з 5 таблиць: Staff, Property_for_Rent, Renter, Qwner, Viewing.

Branch	(Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff	(Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent	(Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter	(Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner	(Ono, FName, LName, Address, Tel_No)
Viewing	(Rno, Pno, Date, Comment)

Проаналізувати, які типи індексів задані на створених таблицях.

Завдання 3

Створити представлення для наступної вибірки: Співробітники компанії, які працюють в лондонському відділенні; дата огляду останнього об’єкта, що пропонується для здачі в оренду, за який відповідає працівник лондонського відділення; ПІ власника цього об’єкта.

Завдання 4

Зайдіть на сайт MSDN.

Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп’ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Представлення:

1. Тема; назва книги; прізвище читача; дата повернення – для книг, у яких прострочений термін повернення; кількість днів прострочення.
2. ПІБ читачів; номер паспорта; кількість прочитаних книг – для читачів, які завжди приносять книги вчасно.
3. Для книги «Великий Гетсбі» показати всі її стани за 5 останніх років.
4. Для кожного видавництва підрахувати кількість книг в бібліотеці і сумарну кількість їх видань.

Варіант 2

Представлення:

1. Назва предмету; кількість лекцій, семінарів, лабораторних занять по ньому.
2. Група; дата; час; назва предмета; викладач; аудиторія (вид заняття – лекція).
3. Назва кафедри; назва предмета; кількість викладачів, з ними пов'язаних.
4. Група; предмет; викладач; вид перевірки.

Варіант 3

Представлення:

1. Назва виробу; назва матеріалу; кількість; одиниці вимірювання.
2. Назва обладнання; виробник; назва специфікації.
3. Назва обладнання; кількість матеріалу і альтернатив; кількість специфікацій, пов'язаних з виробами, які виготовлені на цьому обладнанні.
4. Назва матеріалу; тип; сумарна тривалість виробництва специфікацій виробу.

Варіант 4

Представлення:

1. ПІБ власника; марка та рік випуску його автівки; вартість замовлення і ім'я механіка.
2. Власник; марки його автівок; сумарна вартість і час ремонту.
3. Автомеханік; розряд; сумарний час роботи за місяць; сумарна вартість замовлень.
4. Марка автівки; число раз ремонту автівок в цій автомайстерні, коли час ремонту перевищував один день.

Варіант 5

Представлення:

1. Факультет; курс; кількість заліків; кількість іспитів в літню сесію.
2. Група; назва предмета; дати всіх іспитів і заліків з цього предмету.
3. ПІБ викладача; назва предмета; кількість видів контролю з цього предмету для цього викладача.
4. Назва предмета; кафедра; загальна кількість студентів, що здають цей предмет в зимову сесію.

Варіант 6

Представлення:

1. Проєкт; співробітник; трудомісткість; дата видачі; трудомісткість проєкту.
2. Проєкт; ПІБ працівника; трудомісткість; відсоткове відношення трудомісткості працівника до загальної трудомісткості проєкту.
3. ПІБ працівника; проєкти за останній місяць; трудомісткість для кожного проєкту; час виконання.
4. Назва проєкту; сумарний час роботи над проєктом за рік – для проєктів, у яких не закінчився кінцевий термін протягом року.

Варіант 7

Представлення:

1. Лікар; пацієнт; час лікування – тільки для пацієнтів, які вилікувалися.
2. Соціальний статус; місяць; діагноз; кількість осіб, що вилікувалися.
3. Лікар; кількість пацієнтів за рік, лікування яких закінчилося успішно; загальна кількість пацієнтів за рік.
4. Рік народження; кількість пацієнтів цього року народження; діагноз, що найчастіше у них зустрічається.

Варіант 8

Представлення:

1. Оператор; тариф; кількість зареєстрованих абонентів для цього тарифу.
2. Назва оператора; тарифний план; місяць; число підключень в цей місяць.
3. ПІБ абонента; кількість тарифних планів; улюблений оператор зв'язку.
4. Оператор зв'язку; тариф; загальна заборгованість по тарифу за рік; кількість проданих за рік тарифів.

Варіант 9

Представлення:

1. Вид спорту; змагання; місце; дата; встановлений світовий рекорд.
2. Вид спорту; кількість змагань за рік; кількість потраплянь спортсменів з «України» в призери.
3. Країна; кількість спортсменів цієї країни; загальна кількість перемог спортсменів цієї країни за рік на змаганнях, в назви яких входить слово «плавання».
4. Вид спорту; рекорди, встановлені в цьому виді спорту за останні п'ять років; місце змагань, на яких встановлені рекорди.

Варіант 10

Представлення:

1. Підприємство; продукція; рік; обсяг поставок в цьому році.
2. Підприємство; місяць; обсяг продажів за місяць; сумарна вартість продажів.
3. Назва продукції; підприємство, що займається продажем цієї продукції; обсяг продажів за рік.
4. Вид діяльності підприємства; кількість підприємств; річний бюджет галузі.

Варіант 11

Представлення:

1. Вид транспорту; № маршруту; початковий пункт; кінцевий пункт.
2. Форма транспорту; кількість машин на маршрутах; сумарний прибуток; середній прибуток на одну машину.
3. Маршрут; початковий пункт; кінцевий пункт; всі можливі способи дістатися не більше ніж з двома пересадками в кінцевих пунктах.
4. Форма транспорту; середня кількість пасажирів в одній машині, якщо за день машина робить 20 рейсів; середній прибуток від одного рейсу.

Варіант 12

Представлення:

1. Держава; національність; чисельність; тип управління.
2. Держава; мова; кількість населення, що говорить цією мовою; відсоток від усієї загальної кількості населення.
3. Держава; кількість чоловічого населення; кількість жіночого населення; переважаюча мова; переважаюча національність.
4. Національність; зміна чисельності населення за рік; переважаюча країна проживання.

Варіант 13

Представлення:

1. Рік; будинок; квартира; заборгованість по оплаті.
2. Вид послуг; наявність лічильника; внесена сума оплати за рік; кількість осіб, які оплатили послугу.
3. Власник; послуга; витрати на послугу в рік; середня сума оплати за послугу в місяць.
4. Вулиця; номер будинку; кількість послуг, доступних мешканцям; сумарна вартість за місяць.

Варіант 14

Представлення:

1. Марка літака; місяць; кількість рейсів в цьому місяць; середній час польоту; середня відстань.
2. Пункт вильоту; пункт призначення; всі можливі способи дістатися з не більше ніж однією пересадкою; час шляху.
3. Номер літака; марка літака; розклад рейсів на найближчий місяць; кількість вільних місць.
4. Номер рейсу; пункт вильоту; пункт призначення; марка літака; дата; час – на поточний місяць.

Варіант 15

Представлення:

1. Модель; виробник; гарантія; ціна; дата.
2. Фірма виробник; модель ноутбука; місяць; всі ринкові пропозиції по цій моделі за місяць; ціна.
3. Фірма-виробник; кількість моделей доступних в 2020 році; найбільш популярна модель; її середня ціна.
4. Процесор; відеокарта; розмір екрану; обсяг пам'яті; всі пропозиції для даних характеристик.

Варіант 16

Представлення:

1. ПІБ студента; рік зарахування; середня стипендія за час навчання; факультет.
2. Місто; кількість студентів з цього міста, що навчаються в інституті; найбільш популярний у них факультет.
3. Рік народження студента; рік навчання; кількість студентів цього року народження.
4. Факультет; декан; число місць; середнє число студентів на курсі; чи є переповнення.

Варіант 17

Представлення:

1. Вид; комплекс; середня площа, яка припадає на одну особу; кількість.
2. Родина; кількість тварин цієї родини в зоопарку; найбільш сприятливе середовище проживання.
3. Приміщення; кількість тварин в цьому приміщення; найбільш численний вид у цьому приміщенні.
4. Навколишнє середовище; кількість видів числа дітей в цьому середовищі; кількість приміщень зоопарку, що підтримують це середовище.

Варіант 18

Представлення:

1. Звання; кількість шахістів в цьому званні; кількість перемог на турнірах в поточному році; середнє місце на турнірах в поточному році.
2. Країна; кількість шахістів з цієї країни, які брали участь в турнірах; кількість перемог шахістів з цієї країни; кількість перемог на батьківщині.
3. Шахіст; рейтинг; статистика по всім проведеним турнірам в поточному році; зміна рейтингу протягом року.
4. Місто; рік; шахіст; звання; місце – тільки призери.

Варіант 19

Представлення:

1. Порт; корабель; капітан; мета відвідування; час знаходження – тільки на поточний момент; тільки для «Австралії».
2. Порт приписки; кількість кораблів, яку приймав цей порт; кількість відвідувань цими кораблями порту «Миколаїв» в 2019 році; сумарний прибуток, отриманий з цих кораблів.
3. Порт; категорія; місяць; кількість кораблів, які відвідали порт в даному місяці.
4. Країна; кількість портів; кількість відвідувань країни за рік; загальний прибуток від відвідувань портів іноземними кораблями; загальна вартість відвідування кораблями цієї країни іноземних портів; сальдо між прибутками і витратами.

Варіант 20

Представлення:

1. Рейс; водій; дата відправлення; дата прибуття; пункт відправлення; пункт призначення; автівка.
2. Марка автівки; вантажопідйомність; кількість рейсів за місяць; середня протяжність рейсу.
3. Водій; кількість рейсів за місяць; найбільш часто використовувана ним марка автівки.
4. Пункт відправлення; пункт призначення; відстань; кількість автівок кожної марки, що курсують між цими пунктами.

Варіант 21

Представлення:

1. Організація; кількість вчених з цієї організації; кількість відвіданих ними конференцій; основний тип участі.
2. Тематика; кількість конференцій з цієї тематики; загальна кількість різних вчених, які взяли участь в конференціях з цієї тематики; їх основна спеціалізація.
3. Вчений; спеціалізація; вчений ступінь; кількість часток в конференціях по кожній країні, яка проводить конференції.
4. Вчений ступінь; ПІБ; конференція; тематика конференції; країна; рік; тема доповіді – тільки для вчених, спеціалізація яких не співпадає з тематикою конференції.

Варіант 22

Представлення:

1. Клас продукту; фірма; кількість програм; кількість продажів програм за рік.
2. Регіон; кількість користувачів в цьому регіоні; кількість інсталяцій протягом року; загальна сума інсталяцій.
3. Програмний продукт; версія; фірма-виробник; місяць; кількість інсталяцій в місяць; сумарний прибуток від інсталяцій за рік.
4. Фірма-виробник; продукт; число ліцензій; вартість ліцензій; покупець – тільки для останніх версій.

Варіант 23

Представлення:

1. П'єса; режисер; актори 1-го складу.
2. Амплуа; кількість акторів цього амплуа; відсоток акторів цього амплуа, задіяних в постановках; кількість разів, коли актор цього амплуа виконував роль іншого амплуа.
3. П'єса; кількість ролей в цій п'єсі; кількість різних постановок; дата останньої постановки.
4. Роль; кількість постановок за останні п'ять років, в яких використовувалася ця роль; актор, що найчастіше виконував цю роль.

Варіант 24

Представлення:

1. Виробник; кількість ліків цього виробника; відсоток ліків цього виробника від загального числа ліків в аптеках міста; середня ціна.
2. Спеціалізація; кількість аптек цієї спеціалізації; кількість аптек, які мають всі ліки «Bayer».
3. Ліки; показання; назва аптеки, в якій ці ліки продаються за мінімальною ціною; ціна.
4. Аптека; ліки; кількість; ціна – тільки для ліків, термін придатності яких більше місяця.

Варіант 25

Представлення:

1. Категорія; кількість страв цієї категорії; середня кількість замовлень в день; середня вартість страви.
2. Організація; кількість співробітників цієї організації, які харчуються в їдальні більше десяти разів на місяць; середні витрати співробітника цієї організації на харчування в день.
3. Клієнт; кількість разів, коли він харчувався в їдальні; середня вартість замовлення; кількість разів, коли жодна із замовлених страв не збігалася з уподобаннями клієнта.
4. Для даної організації (назва передається як параметр) вивести чек місячного споживання його співробітників (ПІБ співробітника, страва, кількість замовлень, ціна). Останній рядок – всі поля порожні, а в полі ціна – сумарна вартість замовлень співробітників організації.

Варіант 26

Представлення:

1. Мотор; гума; кількість подіумів в сезоні (пілот займає місце 1-3).
2. Етап; кількість пітстопів; гума; середнє зайняте місце пілотів з такою кількістю пітстопів і такою гумою.
3. Країна; кількість перемог на трасах цієї країни стаєнь-співвітчизників; середня відвідуваність на етапах в цій країні.
4. Стайня; статистика по зайнятим місцям в сезоні для цієї стайні (тільки для фінішувавших пілотів).

Варіант 27

Представлення:

1. Фірма; кількість деталей цієї фірми, встановлених за рік; власник, що вважає за кращу цю фірму найчастіше.
2. Дата; автівка; потужність; зовнішній вигляд; максимальна швидкість на цю дату (тільки дати змін).
3. Частина автівки; деталь; ціна; середні зміни швидкості, потужності і виду, вироблені цією деталлю.
4. Автівка; max зміни параметрів при установці найкращих деталей; вартість.

Варіант 28

Представлення:

1. Спеціалізація археологів; їх кількість в цій спеціалізації; кількість цінних (ціна більше 1000 $) знахідок за рік.
2. Епоха; предмет; вартість; кількість знахідок; стани при знахідках.
3. Місце; дата; кількість працюючих там археологів; кількість зроблених знахідок.
4. Соціальний статус людини; кількість предметів, знайдених на розкопках «Курган», що відносяться до людей з цим статусом; їх стан.

Варіант 29

Представлення:

1. Море; кількість місць відпочинку на цьому морі; кількість осіб, які відпочивали в цих місцях; середня вартість відпочинку в день.
2. Країна; кількість відпочиваючих з цієї країни; кількість разів на рік, коли вони відпочивали на батьківщині.
3. Назва готелю; кількість готелів з цією назвою, доступні для відпочинку на «Чорному» морі.
4. Місце відпочинку; море; готель; тривалість відпочинку – для тур-поїздок, вартість яких не перевищує 2000 $.

Варіант 30

Представлення:

1. Країна; кількість супутників; кількість різних каналів, які віщаються цими супутниками; кількість охоплених поясів.
2. Мова мовлення; кількість каналів, які транслюються цією мовою; телекомпанія, яка мовить найбільшу кількість каналів цією мовою; кількість охоплених поясів.
3. Для кожного пояса і мови мовлення вибрати кількість каналів, які транслюються на них, причому номер пояса повинен бути більше «3».
4. Країна; кількість телекомпаній; кількість каналів цих телекомпаній, які віщаються супутниками даної країни.

Рекомендовані ресурси

MSDN: Створення індексів
https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-index-transact-sql?view=sql-server-ver15

MSDN: Модифікація індексів
https://docs.microsoft.com/ru-ru/sql/relational-databases/indexes/reorganize-and-rebuild-indexes?view=sql-server-ver15

MSDN: Створення представлень та модифікація
https://docs.microsoft.com/ru-ru/sql/relational-databases/views/views?view=sql-server-ver15

Контрольні питання

1. Що таке індекс?
2. Для чого використовується індексація?
3. Чи може впливати індексація на зниження продуктивності?
4. Який оператор використовується для створення індексу?
5. Які типи індексів вам відомі?
6. Що таке сторінка?
7. Що таке екстент?
8. Що таке Index Allocation Map (IAM)?
9. Що таке кластеризований індекс?
10. Що таке кластеризована таблиця?
11. Що таке некластеризований індекс?
12. Який оператор за замовчуванням створює унікальний кластеризований індекс?
13. Який оператор за замовчуванням створює унікальний некластеризований індекс?
14. Як вручну створити унікальний некластеризований індекс (з явним вказуванням на це)?
15. Як вручну створити унікальний кластеризований індекс (з явним вказуванням на це)?
16. Як вручну створити неунікальний некластеризований індекс?
17. Що таке представлення?
18. Який оператор використовується для створення представлення?
19. Які існують умови для зміни даних базової таблиці через представлення?

Комп'ютерний практикум 8
Процедури та функції

Мета роботи:розгляд процедур та функцій в Microsoft SQL Server.

Теоретичні відомості

- ***Мова управління виконанням*** – це мова, яка дозволяє пов'язувати інструкції одна з одною, а також створювати, подібно до мов програмування, конструкції, що виконуються незалежно.
- Розрізняють три види алгоритмічних конструкцій:
  - лінійний алгоритм;
  - алгоритм з розгалуженням;
  - циклічний алгоритм.
- ***Алгоритм з розгалуженням*** – це алгоритм, за допомогою якого певний набір інструкцій виконується в залежності від того, чи є умова істиною. Алгоритм з розгалуженням в T-SQL представлений інструкцією IF ... ELSE.
- ***Циклічний алгоритм*** – це алгоритм, за допомогою якого певний набір інструкцій може виконатись багаторазово, поки умова є істиною. Циклічний алгоритм в T-SQL представлений інструкцією WHILE.
- ***Ітерація*** – це один прохід по тілу циклу.
- ***Оператор*** BREAK – це оператор, за допомогою якого переривається виконання поточної ітерації та циклу в цілому.
- ***Оператор*** CONTINUE – це оператор, за допомогою якого переривається виконання тільки поточної ітерації.
- ***Блок*** TRY – це блок, в який укладається набір інструкцій з великою ймовірністю появи серйозної помилки.
- ***Блок*** CATCH – це блок, в якому перехоплюються помилки, що виникли в блоці TRY.
- Помилки з рівнем серйозності, меншим або рівним 10, розглядаються як попередження або інформаційні повідомлення і не обробляються за допомогою конструкції TRY ... CATCH.
- Помилки з рівнем серйозності, великим або рівним 20, що змушують компонент Database Engine закрити з'єднання, не можуть бути оброблені конструкцією TRY ... CATCH. Однак конструкція TRY ... CATCH забезпечує обробку помилок з рівнем серйозності, великим або рівним 20, якщо з'єднання не закривається.
- Оператори EXISTS і NOT EXISTS – це оператори, результатами обробки яких є логічне значення true або false. Для EXISTS результат дорівнює true в тому і тільки в тому випадку, якщо в результуючій таблиці, яка

повертається підзапитом, присутній хоча б один рядок, в іншому випадку – false. Для NOT EXISTS використовуються правила обробки, зворотні по відношенню до EXISTS.

- ***Збережені процедури*** – це іменований набір операторів T-SQL, який зберігається на сервері.
- Переваги збережених процедур:
  - підвищення продуктивності – коли процедура виконується вперше, то вона компілюється та визначається оптимальний план отримання даних; як наслідок, процедура виконується швидше, оскільки серверу не потрібно розбирати запит і виконувати необхідні дії для вирішення поставленого завдання;
  - повторне використання коду та абстракція;
  - зменшення мережевого трафіку – користувачі можуть виконувати комплекс операцій, надсилаючи тільки один запит, що зменшує кількість запитів між клієнтом і сервером;
  - безпека – користувачі можуть отримувати право виконувати процедури, навіть якщо у них немає права на використання представлення або таблиці, на яку посилається процедура.
- ***Функція, що визначається користувачем*** – це підпрограма, яка приймає параметри, виконує дії та повертає результат цих дій у вигляді значення.
- Функції та процедуримають одні й ті самі переваги.

Таблиця 1 – Відмінності між збереженими процедурами та функціями, що визначаються користувачем

№	Функція	Процедура
1	*2*	*3*
1	Функція повинна повернути значення	Процедура може як повертати, так і не повертати значення
2	Функція дозволяє використання тільки операторів SELECT і не дозволяє використання інших DML-операторів	Процедура може містити як оператори SELECT, так і інші DML-оператори (INSERT, UPDATE, DELETE)
3	Функція підтримує лише вхідні параметри	Процедура підтримує як вхідні, так і вихідні параметри
4	Функція не дозволяє використовувати блоки TRY … CATCH	Процедура для обробки виключень дозволяє використовувати блоки TRY … CATCH
5	В середині функцій транзакції не допускаються	В середині процедур транзакції допускаються
6	Можна використовувати лише табличні змінні. Функція не дозволяє використання тимчасових таблиць	Функція дозволяє використання як табличних змінних, так і тимчасових таблиць
7	З функцій не можна викликати збережені процедури	Збережені процедури можуть викликати функції

Продовження таблиці 1

1	2	3
8	Функції можна викликати з оператора SELECT	Процедури не можуть бути викликані з SELECT/WHERE/HAVING тощо. Оператор EXECUTE/EXEC може використовуватись для виклику/виконання збереженої процедури
9	Функція може використовуватись в операціях JOIN	Процедура не може використовуватись в операціях JOIN

Практичні приклади

```
/*Оператори управління ходом виконання*/
USE AdventureWorks2019;

-------------------------------------------------------------------------
--                    Умовна конструкція IF..ELSE
-------------------------------------------------------------------------

DECLARE @myVar varchar(15);

--SET @myVar = 'Hello World!!!'

-- В умовній конструкції IF, вказуємо вираз: @myVar is NULL
IF @myVar is NULL
        PRINT ('значення myVar не задане');    -- якщо умова задовольняє істиності.
ELSE
        PRINT @myVar;                          -- ІНАКШЕ якщо умова задовольняє істиності.


-------------------------------------------------------------------------
--                         Операція EXISTS
-------------------------------------------------------------------------
-- Операція EXISTS повертає значення TRUE або FALSE, в залежності від того
-- чи є дані, які відповідають запиту SELECT

IF EXISTS (SELECT * FROM Person.Address WHERE City = 'Seattle') -- В імені таблиці вказуємо ім'я
схеми (аналог простору імен)
        PRINT 'We have staff from the city Seattle'
ELSE
        PRINT 'We have no employees of the city Seattle'

-------------------------------------------------------------------------
USE ShopDB;

-- Системні таблиці sys.schemas, sys.tables

-- таблиця sys.schemas містить інформацію про схеми, які застосовуються в даній БД

SELECT * FROM sys.schemas

-- таблиця sys.tables містить інформацію про таблиці, які застосовуються в даній БД

SELECT * FROM sys.tables

-------------------------------------------------------------------------

--приклад створення таблиці TestTable з попередньою перевіркою чи не створена вона раніше.
IF NOT EXISTS (
                                SELECT s.name,t.name
                                FROM sys.schemas AS s                                   -- schemas -
системна таблиця, що зберігає інформацію про схеми
                                        JOIN sys.tables AS t                            -- tables -
системна таблиця, що зберігає інформацію про таблиці
                                         ON s.schema_id = t.schema_id -- вибірка таблиці, що
знаходиться в схемі
                                WHERE s.name = 'dbo'                            -- де схема dbo
                                       AND t.name = 'TestTable'                 -- і таблиця TestTable
                                )
        CREATE TABLE TestTable --створити таблицю, якщо вона не існує
```

```
		(
			Col1 int,
			Col2 varchar(20)
		)
ELSE
	PRINT 'Таблиця TestTable вже існує!' -- вивести повідомлення, якщо існує
-------------------------------------------------------------------------

----- групування операторів в блоки -----

DROP TABLE TestTable;
-- Якщо необхідно виконати декілька операторів, їх об'єднують в блоки BEGIN ... END
IF NOT EXISTS (
				SELECT s.name,t.name
				FROM sys.schemas s
				JOIN sys.tables t
					ON s.schema_id=t.schema_id
				WHERE s.name = 'dbo'
					AND t.name = 'TestTable'
				)

	BEGIN   -- Початок блоку
		PRINT 'Таблиця TestTable не знайдена';
		PRINT 'Створюємо таблицю TestTable';
		CREATE TABLE TestTable   -- Створити таблицю, якщо вона не існує
			(
				Col1 int,
				Col2 varchar(20)
			)
	END   -- Кінець блоку

ELSE
	BEGIN
		PRINT 'Таблиця TestTable існує!';
		PRINT 'Видаляємо таблицю TestTable';
		DROP TABLE TestTable;
		PRINT 'Таблиця TestTable видалена';
	END;
GO
-------------------------------------------------------------------------
--				Оператор CASE
-------------------------------------------------------------------------
-- Простий оператор CASE

DECLARE @myTinyVar TinyInt = 5;

PRINT CASE @myTinyVar		-- вхідний вираз для CASE
		WHEN 0 THEN 'zero'	-- якщо @myIntVar = 0, то виводимо 'zero'
		WHEN 1 THEN 'One'	-- якщо @myIntVar = 1, то виводимо 'One'
		WHEN 2 THEN 'Two'	-- якщо @myIntVar = 2, то виводимо 'Two'
		WHEN 3 THEN 'Three'	-- якщо @myIntVar = 3, то виводимо 'Three'
		ELSE 'More than three' -- якщо не спрацювала жодна з умов, то виводимо 'More than
three'
	END			--кінець оператора CASE
GO

-------------------------------------------------------------------------
-- Пошуковий оператор CASE
DECLARE @MyIntVar int;

SET @MyIntVar = 40;

PRINT CASE -- відсутній вхідний вираз
		WHEN @MyIntVar IS NULL THEN 'змінна @MyIntVar порожня'  -- вираз в конструкції
WHEN повинен приймати булеве значення
		WHEN @MyIntVar < 0 THEN 'змінна @MyIntVar менша за нуль'
		WHEN @MyIntVar > 0 THEN 'змінна @MyIntVar більша за нуль'
		WHEN @MyIntVar > 3 THEN 'змінна @MyIntVar більша за три' -- даний рядок ніколи не
буде виконуватись, оскільки на попередньому рядку була перевірка на умову більше за нуль, а будь-
яке значення, більше три, теж більше за нуль
		ELSE 'Непередбачена ситуація'
	END
GO

-------------------------------------------------------------------------
-- Оператор WHILE. Організація циклів
```

```
DECLARE @myVar int;
SET @myVar = 0;

	WHILE (@myVar < 21) -- умова виконання циклу - поки умова істина, виконується цикл.
	BEGIN
		PRINT 'Поточне Значення ' + CAST (@myVar as varchar);
		SET @myVar = @myVar + 1;
	END
GO
-------------------------------------------------------------------------

DECLARE @myVar int;
SET @myVar =0;

	WHILE @myVar < 21
	BEGIN
		PRINT 'Поточне Значення ' + CAST (@myVar as varchar);
		IF @myVar =5
			BEGIN
				SET @myVar = @myVar + 2;
				CONTINUE; -- Перериває подальше виконання поточної ітерації та
повертається на початок циклу WHILE
			END;
		SET @myVar = @myVar + 1;
	END
GO
-------------------------------------------------------------------------

DECLARE @myVar int;
SET @myVar = 0;

	WHILE @myVar < 21
	BEGIN
		PRINT 'Поточне Значення ' + CAST (@myVar as varchar);
		IF @myVar = 7
		BEGIN
			PRINT '@myVar = 7! Переривання циклу!'
			BREAK; -- Оператор переривання циклу (не рекомендується використовувати)
		END
		SET @myVar = @myVar + 1;
	END
GO

-------------------------------------------------------------------------
--------------------------- оператор WAITFOR -----------------------------

WAITFOR DELAY '00:00:10'; -- Затримка на проміжок часу. Можливі значення: в
годинах:хвилинах:секундах

PRINT 'Пройшло 10 секунд';

WAITFOR TIME '15:50:01'; -- Затримка до вказаного значення часу. Можливі значення: в
годинах:хвилинах:секундах

PRINT ' час продовжувати';

-------------------------------------------------------------------------
--------------------------- Блоки TRY и CATCH ----------------------------
-- призначені для обробки помилок

BEGIN TRY -- Спроба виконати код, якщо виникає помилка, то заносимо її до блоку CATCH, інакше
продовжуємо роботу без блоку CATCH.

	CREATE TABLE TestTable
		(
			col1 int,
			col2 varchar(10)
		);

END TRY
BEGIN CATCH -- блок CATCH - блок-обробник помилок

	DECLARE @ErrorNo  int,
			@Message  nvarchar(4000);

	SELECT
```

```
		@ErrorNo = ERROR_NUMBER(),		--системна функція, повертає код поточної
помилки
		@Message = ERROR_MESSAGE();		--системна функція, повертає повідомлення
поточної помилки

	IF @ErrorNo = 2714
		PRINT 'Дана таблиця вже існує!'
	ELSE
		PRINT CAST(@ErrorNo as varchar)+' '+@Message;
END CATCH
GO


-------------------------------------------------------------------------

BEGIN TRY -- намагаємось виконати код, якщо виникає помилка, то заносимо її до блоку CATCH,
інакше продовжуємо роботу без блоку CATCH.

	DROP TABLE TestTable;

END TRY

BEGIN CATCH -- блок CATCH - блок-обробник помилок

	DECLARE @ErrorNo  int,
		@Message  nvarchar(4000);

	SELECT
		@ErrorNo = ERROR_NUMBER(),		--системна функція, повертає код поточної
помилки
		@Message = ERROR_MESSAGE();		--системна функція, повертає повідомлення
поточної помилки

	IF @ErrorNo = 3701
		PRINT 'Дана таблиця вже видалена!'
	ELSE
		PRINT CAST(@ErrorNo as varchar)+' '+@Message;
END CATCH
GO

-------------------------------------------------------------------------
--		Активація повідомлення про помилку вручну
IF NOT EXISTS (
				SELECT s.name,t.name
				FROM sys.schemas as s
				JOIN sys.tables t
					ON s.schema_id = t.schema_id
				WHERE s.name = 'dbo'
					AND t.name = 'TestTable'
				)

	BEGIN	-- початок блоку
		PRINT 'Таблиця TestTable не знайдена';
		PRINT 'Створюємо таблицю TestTable';
		CREATE TABLE TestTable --створити таблицю, якщо вона не існує
			(
				Col1 int,
				Col2 varchar(10)
			)
	END	-- кінець блоку

ELSE
	BEGIN
		RAISERROR('Дана таблиця існує', 11, 238) -- Помилка, що викликана користувачем
вручну
		-- RAISERROR('Дана таблиця існує', 11, 238)-- Зняти коментар
		-- (другий аргумент RAISERROR - степінь серйозносты помилки, позначення стану)
		-- RAISERROR (рядок повыдомлення, степінь серйозносты помилки, позначення стану)
		/*степінь серйозносты помилки - визначаэ якы дыъ необхыдно приймати при виникненны
помилки
		  позначення стану - довыльне значення, застосовуэться в ролы маркера мысця
виникнення помилки,
		  якщо помилка виникаэ в декылькох мысцях, то переглядаючи значення
		  оглядача стану можна визначити, де в коды виникла помилка*/
	END;
GO
USE AdventureWorks2019;
```

```
GO
-----------------------------------------------------------------

CREATE PROC spEmployee -- Створення зберігаємої процедури.
AS
  SELECT * FROM HumanResources.Employee;
GO

EXEC spEmployee; --Виклик зберігаємої процедури.
GO
-----------------------------------------------------------------

ALTER PROC spEmployee -- Модифікація зберігаємої процедури.
AS
  SELECT he.BirthDate,he.BusinessEntityID FROM HumanResources.Employee he;
GO

EXEC spEmployee; -- Виклик модифікованої зберігаємої процедури.

DROP PROC spEmployee; -- Видалення зберігаємої процедури.

EXEC spEmployee; -- ПОМИЛКА, оскільки дана процедура видалена.
GO

-----------------------------------------------------------------

/*Зберігаєма процедура з параметрами*/

DROP PROC spEmployeeByName

CREATE PROC spEmployeeByName
       @LastName nvarchar(25)    -- При ініціалізації параметра значення за замовчуванням не
задане.
AS

       SELECT pc.BusinessEntityID, pc.FirstName, pc.LastName, pc.ModifiedDate
       FROM Person.Person pc
       WHERE pc.LastName = @LastName;
GO


EXEC spEmployeeByName 'Abel' -- Передаємо процедурі обов'язкове значення.

EXEC spEmployeeByName -- ПОМИЛКА, якщо не передати обов'язкове значення за замовчуванням.

-----------------------------------------------------------------

DROP PROC spEmployeeByName;
GO

CREATE PROC spEmployeeByName
       @LastName nvarchar(25) = NULL -- Щоб вказати, що параметр є необов'язковим, при
ініціалізації необхідно ввести для нього значення за замовчуванням.

AS
IF @LastName IS NOT NULL  -- Умовна конструкція IF, де @LastName IS NOT NULL є перевірочним
виразом, який повертає TRUE або FALSE.

       SELECT pc.BusinessEntityID, pc.FirstName, pc.LastName, pc.ModifiedDate
       FROM Person.Person pc
       WHERE pc.LastName LIKE @LastName + '%'

ELSE

       SELECT pc.BusinessEntityID, pc.FirstName, pc.LastName, pc.ModifiedDate
       FROM Person.Person pc;

GO

EXEC spEmployeeByName     -- Виклик без параметра.

EXEC spEmployeeByName 'Ca' --Виклик з параметром.

EXEC spEmployeeByName 'Cao'
GO
-----------------------------------------------------------------------------
/*Вихідні параметри в зберігаємих процедурах*/
```

```
DROP PROC spEmployeeCount;
GO

CREATE PROC spEmployeeCount
	@Info int = null OUTPUT -- Для оголошення вихідного парметра використовується ключове
слово OUTPUT.
AS
BEGIN
	SET @Info =(SELECT Count(*) From Person.Person);
END
GO

DECLARE @MyInfo int;

EXEC  spEmployeeCount @MyInfo OUTPUT; -- При виклику зберігаємої процедури ключове слово OUTPUT
повинне використовуватись так само як і при оголошенні; зверніть увагу, як присвоюється значення
зовнішній змінній

PRINT CAST (@MyInfo as varchar);
GO
-----------------------------------------------

---   Оператор повернення значення RETURN   ---

DROP PROC TestProc

CREATE PROC TestProc
AS
BEGIN
	DECLARE @MyVar int = 10;
	RETURN @MyVar; -- Оператор RETURN в процедурах повертає ТІЛЬКИ ЦІЛОЧИСЕЛЬНЕ ЗНАЧЕННЯ!
END;
GO

DECLARE @MyRTN int;
EXEC @MyRTN = TestProc;
PRINT CAST(@MyRTN as varchar);
GO

-------------------------------------------
DROP PROC TestProc;
GO

CREATE PROC TestProc
AS
BEGIN
	DECLARE @MyVar int = 10;
	RETURN 'Done' -- Оператор RETURN в процедурах повертає ТІЛЬКИ ЦІЛОЧИСЕЛЬНЕ ЗНАЧЕННЯ!!!
END;
GO

DECLARE @MyRTN2 varchar(5);
EXEC @MyRTN2 = TestProc;
PRINT @MyRTN2;
GO
-------------------------------------------
DROP PROC spTestProc
GO

CREATE PROC spTestProc
AS
BEGIN
	PRINT 'Зараз виконається перший RETURN';
	RETURN; -- За замовчуванням оператор RETURN повертає 0.
	PRINT 'Зараз не виконається другий RETURN';
	RETURN 5; -- Після виконання першого оператора RETURN, процедура припиняє своє виконання.
END;
GO

DECLARE @MyVar3 int;
EXEC @MyVar3 = spTestProc;
PRINT @MyVar3;
GO
-----------------------------------------
--- Рекурсія ----
-- МАКСИМАЛЬНА ГЛИБИНА РЕКУРСІЇЇ 32 РІВНЯ
DROP PROC spFactorial
```

```
CREATE PROC spFactorial
@ValueIn int,
@ValueOut int OUTPUT
AS
BEGIN
	DECLARE @InWorking int;
	DECLARE @OutWorking int;
	IF (@ValueIn != 1)
	BEGIN
		SET @InWorking = @ValueIn -1;
		EXEC spFactorial @InWorking, @OutWorking OUTPUT; -- Виклик процедури з самої себе
(рекурсія).
		SET @ValueOut = @ValueIn * @OutWorking;
	END
	ELSE
		SET @ValueOut = 1;
END;
GO
-------------------------------------------

DECLARE @MyVarOut int,
		@MyVARIn int;

SET @MyVARIn = 7;
EXEC spFactorial @MyVarIn, @MyVarOut OUTPUT; -- 7!= 1*2*3*4*5*6*7
PRINT CAST(@MyVARIn as varchar) + ' факторіал: ' + CAST(@MyVarOut as varchar);
GO
----------------------
-- Помилка. Не вистачає діапазона значень типу, який використовується для змінних!
DECLARE @MyVarOut int,
		@MyVARIn int;

SET @MyVARIn = 13;
EXEC spFactorial @MyVarIn, @MyVarOut OUTPUT; -- 13!= 1*2*3*4*5*6*7*8*9*10*11*12*13
PRINT CAST(@MyVARIn as varchar) + ' факторіал: ' + CAST(@MyVarOut as varchar);

----Процедура реєстрації помилок в таблиці -----
DROP PROC uspLogError;
GO

DROP TABLE ErrorLog2

CREATE TABLE ErrorLog2
(
	ErrorId int IDENTITY,
	ErrorLine int,
	ErrorMessage varchar(200)
)
GO

-- Створюємо процедуру реєстрації помилок -----

CREATE PROC uspLogError
	@ErrorLogId int = 0 OUTPUT
AS
BEGIN
	INSERT dbo.ErrorLog2 -- Запис даних про помилку.
		(
			ErrorLine,
			ErrorMessage
		)
		VALUES
		(
			ERROR_LINE(),
			ERROR_MESSAGE()
		)
	SET @ErrorLogId = @@IDENTITY; -- @@IDENTITY повертає номер останнього ідентифікатора, який
записаний в таблицю.
END;
-------------------------------------------------

BEGIN TRY

	CREATE TABLE OurTest
	(
		col int
	)
```

```
END TRY
BEGIN CATCH

        DECLARE @OutIdError int;

        IF ERROR_NUMBER() = 2714
        BEGIN
                PRINT 'Спроба створити існуючу таблицю';
                EXEC uspLogError @OutIdError OUTPUT;
                PRINT 'Помилка записана в журнал Номер запису: ' + CAST(@OutIdError as varchar);
        END

        ELSE
          PRINT 'Невідома помилка ';

END CATCH

SELECT * FROM ErrorLog2
GO
/*Функції користувача, які повертають скаляре значення (не табличне)*/
USE ShopDB
GO
DROP FUNCTION Hello
CREATE FUNCTION Hello() -- Створити функцію.
RETURNS varchar(30)     -- Оголошуємо тип повертаємого значення.
AS
BEGIN --Початок тіла функції.
DECLARE @MyVar varchar(20) ='Hello World!';
RETURN @MyVar; --Повертаєме значення функції.
END; -- Кінець тіла функції.
GO

PRINT dbo.Hello();

DROP TABLE TestTable;
GO

CREATE TABLE TestTable
(
        id int identity not null,
        name varchar(25) not null,
        CDate smalldatetime not null
)
GO

-----------------------------------------------------

DECLARE @MyVar int =1;
DECLARE @MyVcVar varchar(10);

WHILE @MyVar < 20
BEGIN
        SET @MyVcVar = 'Test ' + CAST(@MyVar as varchar);
        -- Заносимо дані в таблицю.
        INSERT TestTable
        ( name, CDate )
        VALUES (@MyVcVar, DATEADD(MI, @MyVar, GETDATE()));

        SET @MyVar = @MyVar + 1;
END
GO

/* Дана вибірка буде порожньою, оскільки функція GETDATE()
 повертає значення типу datetime (повертає не лише
 дату, а і поточний час до 3-х сотих секунди).*/
SELECT * FROM TestTable
WHERE CDate = GETDATE();
GO
-----------------------------------------------------------
DROP FUNCTION DateOnly
CREATE FUNCTION DateOnly (@Date datetime) -- Створюємо функцію, вводячі її ім'я і аргумент.
RETURNS date --Оголошуємо повертаємий тип.
AS
BEGIN --Тіло функції.
 RETURN CAST(@Date as date); -- Приводимо до типу date повертаєме значення.
END
```

```
GO

SELECT * FROM TestTable -- Дана вибірка поверне 19 записів (всі за сьогоднішній день).
WHERE dbo.DateOnly(CDate)= dbo.DateOnly(GETDATE()); --Потрібно явно вказувати схему при виклику
функції користувача.
----------------------------------------
USE AdventureWorks2019
GO

/* Робимо вибірку для перегляду ціни кожного гірського велосипеду,
в даній вибірці використовуються запити всередині запитів - їх
називають вкладеними запитами.*/

SELECT  Name,
                ListPrice,
                (SELECT AVG(ListPrice) FROM Production.Product WHERE ProductSubcategoryID = 1) as
Average, -- Використовуємо вкладений запит; функція AVG - повертаєме среднє значення.
                ListPrice - (SELECT AVG(ListPrice) FROM Production.Product WHERE
ProductSubcategoryID = 1) as DifferencePrice
FROM Production.Product
WHERE ProductSubcategoryID = 1; -- Робимо вибірку для Mountain Bikers.
GO

DROP FUNCTION AvgPrice
DROP FUNCTION DfrncPrice
GO

-- Вкладений запит (SELECT AVG(ListPrice)FROM Production.Product) можна оформити у вигляді
функції,
-- оскільки він повертає вибірку з одного значення.
CREATE FUNCTION AvgPrice()
RETURNS money
WITH SCHEMABINDING    -- Забороняє видалення таблиці.
AS
BEGIN
        RETURN (SELECT AVG(ListPrice)FROM Production.Product WHERE ProductSubcategoryID = 1);
END
GO

--- Використання вкладених функцій.
CREATE FUNCTION DfrncPrice(@Price money)
RETURNS money
AS
BEGIN
        RETURN @Price - dbo.AvgPrice(); -- Можливо створювати вкладені функції.
END;
GO

-- Виконаємо той самий запит для перегляду ціни кожного гірського велосипеду.
SELECT  Name,
                ListPrice,
                dbo.AvgPrice() as AvgPrice,
                dbo.DfrncPrice(ListPrice) as DifferencePrice
FROM Production.Product
WHERE ProductSubcategoryID = 1;
GO

/*Функції користуваач, які повертають таблицю*/

USE AdventureWorks2019;
GO

DROP FUNCTION fnContactList

CREATE FUNCTION fnContactList() -- Створюємо функцію.
RETURNS TABLE -- Повертаємий тип TABLE вказує на те, що повертатись буде таблиця.
AS
RETURN (SELECT LastName, FirstName, ModifiedDate
                FROM Person.Person);
GO

SELECT * FROM dbo.fnContactList(); -- Виводимо всю інформацію з таблиці, сформованої за допомогою
функції fnContactList();.
GO
--ЗАВДАННЯ: повторити попередній приклад за допомогою представлення.
DROP FUNCTION fnContactSearch
```

```
CREATE FUNCTION fnContactSearch(@LastName varchar(30)) --Передаємо один аргумент типу varchar
(Прізвище або частини прізвища)
RETURNS TABLE
AS
RETURN (SELECT FirstName , LastName, ModifiedDate
              FROM Person.Person
              WHERE LastName LIKE @LastName + '%'); -- Вибірка по прізвищу.
GO

SELECT * FROM dbo.fnContactSearch('Berry'); -- Виводимо всю інформацію про співробітників,
частини прізвища або прізвище яких 'Berry', з таблиці сформованої за допомогою fnContactSearch();

SELECT * FROM dbo.fnContactSearch('Ber'); -- Виводимо всю інформацію про співробітників, частини
прізвища або прізвище яких 'Ber', з таблиці сформованої за допомогою fnContactSearch();
USE ShopDB
GO
---------- Створюємо таблицю співробітників. ----------
CREATE TABLE MyEmployee
(
  EmployeeID int NOT NULL,
  ManagerID int NULL REFERENCES MyEmployee(EmployeeID), -- Показує підлеглість співробітників.
  JobTitle nvarchar(50) NOT NULL,
  LastName nvarchar(50) NOT NULL,
  FirstName nvarchar(50) NOT NULL,
    CONSTRAINT PK_Employee2_EmployeeID PRIMARY KEY CLUSTERED (EmployeeID ASC)
);
GO

-- Ієрархічні дані (оскільки таблиця містить інформацію про співробітників, які є підлеглими
співробітників, про яких інформація також зберігається в цій таблиці)
INSERT INTO MyEmployee(EmployeeID,ManagerID,JobTitle,LastName,FirstName)
VALUES
  (1, NULL, 'Chief Executive Officer','Ritchie', 'Mike'),
  (2, 1, 'Financial Director', 'Stewart', 'Aline'),
  (3, 1, 'Staff Support','Benson','Charlie'),
  (4, 1, 'Chief Executive', 'Evans', 'Jane'),
  (5, 4, 'Engineer', 'Kay', 'Riley'),
  (8, 5, 'Engineer', 'Smith', 'Tony'),
  (9, 5, 'Programmer', 'Walker', 'John'),
  (10 ,5, 'Data Architect', 'Fulton', 'Daniel'),
  (11 ,5, 'Programmer', 'Morrison', 'Noah'),
  (6, 4, 'Personal Secretary', 'Allford', 'Harry'),
  (7, 4, 'Chief of Security','Foster','Matthew');

SELECT * FROM MyEmployee;

-- Дізнаємося хто є підлеглими виконавчого директора.
SELECT  sub.EmployeeID,
              sub.FirstName,
              sub.LastName
FROM
       MyEmployee as boss
       JOIN
       MyEmployee as sub
       ON boss.EmployeeID = sub.ManagerID
WHERE boss.JobTitle LIKE 'Chief Executive Officer';
GO

-- Рекурсивна функція виводу всіх підлеглих.

CREATE FUNCTION fnGetSub (@EmployeeId int) -- Створюємо функцію з одним аргументом.
RETURNS @Sub TABLE  -- Оновлюємо повертаєму таблицю.
               (
                      EmployeeId int,
                      SubId int,
                      Name varchar (90)
               )
AS
BEGIN
       DECLARE @EmpId int;

INSERT @Sub -- Додаємо запис про керівника у вихідну таблицю.
       SELECT EmployeeID, ManagerID ,FirstName+' '+LastName
       FROM MyEmployee
       WHERE EmployeeID = @EmployeeId;

SET @EmpId = (SELECT MIN(EmployeeID) -- Визначаємо першого підлеглого.
```

```
                        FROM MyEmployee
                        WHERE ManagerID = @EmployeeId);

-- Якщо такий є, то заходимо в циклічну конструкцію WHILE
WHILE @EmpId IS NOT NULL -- Поки є підлеглі, виконуємо цикл.
BEGIN
       INSERT @Sub -- Додаємо запис про підлеглих відносно вищеобраного керівника.
              SELECT * FROM dbo.fnGetSub(@EmpId) -- Рекурсия.

       SET @EmpId =(SELECT MIN(EmployeeID) -- Визначаємо наступного підлеглого.
                           FROM MyEmployee
                           WHERE EmployeeID > @EmpId
                                  AND
                                  ManagerID = @EmployeeId);
END;
RETURN; -- Якщо ми оголосили повертаєму таблицю, то в RETURN не підставляємо ніякого значення,
інакше виникне помилка.
END;
GO

SELECT * FROM dbo.fnGetSub(4); -- Скористаємося для пошуку функцією - fnGetSub.
SELECT * FROM MyEmployee;
DROP FUNCTION fnGetSub
DROP TABLE MyEmployee
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Зайти в створену раніше БД DreamHomeDB, яка складається з 5 таблиць: Staff, Property_for_Rent, Renter, Qwner, Viewing.

Branch	(Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff	(Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent	(Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter	(Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner	(Ono, FName, LName, Address, Tel_No)
Viewing	(Rno, Pno, Date, Comment)

Завдання 3

Створити функції/процедури для таких завдань:

1) необхідно дізнатись номери телефонів співробітників та адреси об'єктів нерухомості, за які вони відповідають;

2) необхідно дізнатись про заробітну платню співробітників, які відповідають за об'єкти нерухомості, що знаходяться в Лондоні;

3) необхідно дізнатись, за які об'єкти нерухомості відповідають співробітники, молодші 30 років.

Завдання 4

Зайдіть на сайт MSDN.

Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп'ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Процедури та функції:

1. Визначити книгу, яка була найбільш популярною взимку 2020 року.
2. Визначити читачів, у яких на руках знаходяться книги видавництва «Освіта і Наука».

Варіант 2

Процедури та функції:

1. Вивести розклад занять викладача «Добролюбова М.В.» на листопад 2020 року.
2. Визначити для кожної групи кількість іспитів і заліків.

Варіант 3

Процедури та функції:

1. Вивести список виробів, яких виходить найбільше з одиниці об'єму сталі.
2. Вивести середній термін придатності обладнання.

Варіант 4

Процедури та функції:

1. Визначити тих власників автомобілів, яких завжди обслуговує один і той самий механік. Вивести прізвища механіка і його постійного клієнта.
2. Для кожної марки автомобіля вибрати механіка, найчастіше обслуговуючого цю марку.

Варіант 5

Процедури та функції:

1. Визначити, відсоток іспитів, що здаються групою «ПА-01мп», від загального числа предметів для кожної категорії.
2. Визначити, чи не здає будь-яка група два іспити або заліки в один день.

Варіант 6

Процедури та функції:

1. Визначити ті роботи, які до дати завершення були виконані не більше, ніж на 50 %.
2. Визначити 5 співробітників, що виконують найбільший обсяг робіт.

Варіант 7

Процедури та функції:

1. Визначити відсоток смертності від захворювання «карієс».
2. Вивести імена тих пацієнтів, які лікувалися у лікарів усіх спеціальностей.

Варіант 8

Процедури та функції:

1. Підрахувати, скільки є незареєстрованих номерів у кожного з операторів.
2. Вибрати список абонентів оператора «Київстар», що мають заборгованість більше 3000 грн.

Варіант 9

Процедури та функції:

1. Вивести список спортсменів, що беруть участь більш ніж у 8 змаганнях на рік в порядку зростання середнього місця на цих змаганнях.
2. Визначити найкращий показник спортсмена «Фуркад» у виді спорту «біатлон» і різницю зі світовим і олімпійським рекордами.

Варіант 10

Процедури та функції:

1. Визначити випадки, коли були продані товари із терміном придатності не більше тижня.
2. Вивести список продуктів, для яких закупівельна ціна, як правило, нижче собівартості виробника.

Варіант 11

Процедури та функції:

1. Вивести маршрути машини «Газель» в порядку спадання їх довжини.
2. Визначити очікуваний прибуток фірми за день.

Варіант 12

Процедури та функції:

1. Визначити тип управління тієї країни, де проживає найбільше представників національності «вірмени».
2. Вибрати список національностей, які проживають в країнах з анархічним ладом.

Варіант 13

Процедури та функції:

1. Визначити загальну суму виплат для квартир по вулиці Боткіна.
2. Вибрати список власників, які мають заборгованості понад рік.

Варіант 14

Процедури та функції:

1. Вибрати маршрут / маршрути, за якими найчастіше літають рейси, заповнені менш, ніж на 70 %.
2. Визначити наявність вільних місць на рейс №354 23 серпня 2019 року.

Варіант 15

Процедури та функції:

1. Визначити фірму, що здійснила обсяг продажів на найбільшу суму за 2020 рік.
2. Вибрати міста, в яких випускаються ноутбуки на базі «ASUS».

Варіант 16

Процедури та функції:

1. Для кожного факультету вибрати список груп із зазначенням чисельності студентів в кожній групі.
2. Вибрати прізвища і величину стипендії студентів з міста «Житомир».

Варіант 17

Процедури та функції:

1. Визначити загальну чисельність представників сімейства «вовчі» в зоопарку.
2. Вивести середню тривалість життя для мешканців «тераріуму».

Варіант 18

Процедури та функції:

1. Вибрати тих шахістів, які виграли не менше трьох турнірів протягом 2020 року.
2. Визначити турніри, в яких учасник з найвищим рейтингом посів останнє місце.

Варіант 19

Процедури та функції:

1. Визначити, який прибуток отримав порт міста Миколаїв від кораблів з Туреччини.
2. Визначити, з якою метою найчастіше заходять кораблі в порт «Тортуга».

Варіант 20

Процедури та функції:

1. Визначити загальну витрату палива у перевезеннях, що здійснені у 2019 році.
2. Визначити середній час поїздки для водія «Шевченко».

Варіант 21

Процедури та функції:

1. Визначити, в якій конференції брало участь більше всього докторів наук.
2. Визначити середній відсоток іноземних вчених на конференціях, що проходять в місті «Київ».

Варіант 22

Процедури та функції:

1. Визначити тих покупців, для яких сумарна вартість всіх модифікацій продукту перевищила вартість ліцензії на продукт останньої версії.
2. Вибрати список продуктів типу «серверні операційні системи», в порядку спадання їх популярності.

Варіант 23

Процедури та функції:

1. Вибрати список п'єс, в яких стать виконавців ролей змінювалась.
2. Вибрати список акторів, які зайняті більш ніж в трьох постановках одночасно.

Варіант 24

Процедури та функції:

1. Визначити, в яких аптеках найнижча середня ціна на болезаспокійливі.
2. Вибрати кількість таблеток, на яких доведеться змінити дату закінчення терміну придатності, якщо вони не будуть реалізовані протягом тижня.

Варіант 25

Процедури та функції:

1. Вибрати середній щоденний дохід їдальні.
2. Вибрати найбільш споживану страву в березні.

Варіант 26

Процедури та функції:

1. Вибрати етапи, на яких більшість очок набирали пілоти на гумі «Michelin».
2. Для кожного пілота визначити середню швидкість на етапі за рік.

Варіант 27

Процедури та функції:

1. Для кожного власника визначити витрати на тюнінг автомобілів.
2. Визначити найбільш оздоблені автомобілі для кожної марки і їх параметри.

Варіант 28

Процедури та функції:

1. Вивести повну інформацію про предмети, знайдені в районі «Каїра» епохи «Древньоєгипетська».
2. Вивести середню вартість знахідок для кожного археолога за час їх роботи.

Варіант 29

Процедури та функції:

1. Вибрати всі можливі варіанти не більше двох тур-поїздок на суму 3000 $, так щоб відпочинок пройшов на «Середземному» морі.
2. Вибрати кількість днів відпочинку і кількість місць відпочинку в минулому році для кожного клієнта фірми.

Варіант 30

Процедури та функції:

1. Вибрати супутники, що не ведуть мовлення жодного каналу своєї країни.
2. Вибрати телекомпанії, що ведуть мовлення на Україну каналів розважальної специфіки.

Рекомендовані ресурси

MSDN: Функції
https://msdn.microsoft.com/ru-ru/library/ms186755.aspx

MSDN: Процедури
https://msdn.microsoft.com/ru-ru/library/ms187926.aspx

Контрольні питання

1. Які типи процедур вам відомі?
2. Які типи функцій вам відомі?
3. Яка різниця між процедурою та функцією?
4. Чи мають процедури повертаємі значення?

Комп'ютерний практикум 9

Транзакції та тригери

Мета роботи: розгляд методів роботи з транзакціями та всіх видів тригерів.

Теоретичні відомості

- ***Транзакція*** – це група послідовно виконуємих операторів SQL, які послідовно виконуються і повинні бути або виконані всі, або не повинен бути виконаний жодний з них.
- Типи транзакцій:
  - неявні – кожен оператор, такий як INSERT, UPDATE, DELETE, виконується в транзакції;
  - явні – група інструкцій мови SQL, початок і кінець якої позначаються такими інструкціями, як BEGIN TRANSACTION, COMMIT, ROLLBACK.
- ***Відкат транзакції*** (rollback) – це дія, яка забезпечує анулювання всіх змін даних, які були зроблені в тілі поточної незавершеної транзакції.
- ***Точки збереження*** – це іменовані мітки, які розбивають транзакцію на етапи, дозволяючи повертатись до виконання будь-якого з етапів, вказавши до якої мітки потрібно виконати повернення.
- ***Атомарність*** (atomicity) – це властивість транзакції, яка гарантує, що жодна транзакція не буде зафіксована в системі частково.
- ***Узгодженість*** (consistency) – це властивість транзакції, яка гарантує, що в результаті виконання транзакції база даних не буде містити неузгоджених даних, тобто трансформації даних, які виконуються транзакцією, переводять базу даних з одного узгодженого стану в інший.
- ***Ізольованість*** (isolation) – це властивість транзакції, яка гарантує, що під час виконання транзакції паралельні транзакції не повинні впливати на її результат. Однак, оскільки ізольованість – вимога дорога, в реальних БД існують режими, які не повністю ізолюють транзакцію (рівні ізольованості Repeatable- Read і нижче).
- ***Стійкість*** (durability) – це властивість транзакції, яка гарантує, що в разі системної помилки (знеструмлення системи або збої в обладнанні) зміни, зроблені успішно завершеною транзакцією, повинні залишитися збереженими після повернення системи до робочого стану.
- ***Втрачене оновлення*** (lost update) – побічний ефект паралелізму, який означає, що при одночасній зміні одного блоку даних різними транзакціями одна із змін втрачається.
- ***«Брудне» читання*** (dirty reads) – побічний ефект паралелізму, який означає читання однією транзакцією даних, доданих або змінених іншою транзакцією, яка згодом не підтвердиться (відкотиться);

- ***Неповторюване читання*** (non-repeatable reads) – побічний ефект паралелізму, який означає, що при повторному читанні в рамках однієї транзакції раніше прочитані дані виявляються зміненими.
- ***Фантомне читання*** (phantom reads) – побічний ефект паралелізму, який означає, що одна транзакція в ході свого виконання кілька разів вибирає множину записів за одними і тим самими критеріями. Інша транзакція в інтервалах між цими вибірками додає/видаляє записи або змінює поля деяких записів, які використовуються в умовах вибірки першої транзакції, і успішно закінчується. В результаті вийде, що одні й ті самі вибірки в першій транзакції дають різні множини записів.
- ***Тригер*** – це зберігаєма процедура особливого типу, яку користувач не викликає безпосередньо, а виконання якої обумовлено настанням певної події на сервері бази даних. Існують тригери на події DDL та DML. Тригери DML часто використовуються для застосування бізнес-правил та забезпечення цілісності даних. Події тригера DML: INSERT, UPDATE, DELETE.
- Типи тригерів:
  - FOR або AFTER – тригер DML спрацьовує тільки після успішного виконання всіх операцій в інструкції SQL, яка запускається тригером.
  - INSTEAD OF – тригер DML спрацьовує замість інструкції SQL, яка використовується тригером, перевизначаючи таким чином дії інструкції тригера, яка виконується.
- Службові таблиці всередині тригера:
  - *inserted* – для події INSERT містить рядки, які вставляються в цільову таблицю, для UPDATE – рядки з новими даними, для DELETE – порожня;
  - *deleted* – для INSERT – порожня, для UPDATE – рядки зі старими даними, для DELETE – рядки, які видаляються.

Таблиця 1 – Рівні ізольованості

Проблема / Рівень ізольованості	Lost update (втрачене оновлення)	Dirty reads («брудне» читання)	Non-repeatable reads (неповторюване читання)	Phantom reads (читання фантомів)
READ UNCOMMITTED	Запобігає	Не запобігає	Не запобігає	Не запобігає
READ COMMITTED	Запобігає	Запобігає	Не запобігає	Не запобігає
REPEATABLE READ	Запобігає	Запобігає	Запобігає	Не запобігає
SERIALIZABLE	Запобігає	Запобігає	Запобігає	Запобігає

Практичні приклади

```
USE ShopDB;
GO
--IF OBJECT_ID('MyUserName') IS NOT NULL
--     DROP TABLE MyUserName;
```

```
--GO
-- Видаляємо таблиці, якщо вони існують
DROP TABLE MyUserName;
GO
DROP TABLE MyUserTell;
GO
DROP TABLE MyUserInfo;
GO
-- ... помилка, яка і як виправити?

--- Створити таблицю ---
CREATE TABLE MyUserName
(
	IdTest int IDENTITY PRIMARY KEY,
	FName varchar(25),
	LName varchar(25)
)
GO

CREATE TABLE MyUserTell
(
	IdTest int IDENTITY PRIMARY KEY,
	IdUser int FOREIGN KEY REFERENCES MyUserName(IdTest) UNIQUE,
	TellN char(14)
	CHECK (TellN LIKE ('([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]')) -- Що це
значить?
)
GO

CREATE TABLE MyUserInfo
(
	IdTest int IDENTITY PRIMARY KEY,
	IdUser int FOREIGN KEY REFERENCES MyUserName(IdTest),
	Bdate date
)
GO
-------------------------- Транзакції ------------------------------
-- Ми вже використовували транзакції (неявно)
 BEGIN TRANSACTION; -- початок транзакції
 INSERT MyUserName
 VALUES('Mike','Ritchie')
 COMMIT TRANSACTION; -- успішне завершення транзакції

 SELECT * FROM MyUserName;
------------------------------------------------------------
 -- відкат транзакції

 BEGIN TRANSACTION  -- початок транзакції
 INSERT MyUserName
 VALUES
	('Aline','Stewart')

 ROLLBACK TRANSACTION; -- відбувається відкат транзакції, наша вставка не виконається, тобто:
                                    -- INSERT MyUserName VALUES ('TestName1','RollTestL1') ,-
не відбудеться

SELECT * FROM MyUserName;

------------------------------------------------------------

-- В одній транзакції можна виконувати декілька дій

BEGIN TRANSACTION; -- початок транзакції

DECLARE @Id int;

INSERT MyUserName
VALUES
	('Mike','Ritchie')

SET @Id = @@IDENTITY;

INSERT MyUserTell
VALUES
(@Id,'(093)777-77-77');

INSERT MyUserInfo
```

```
VALUES (@Id,'17/05/1982');

COMMIT TRANSACTION; -- успішне завершення транзакції

SELECT * FROM MyUserName;
SELECT * FROM MyUserInfo;
SELECT * FROM MyUserTell;
GO

------------------------------------------------------------

-- В одній транзакції можна "відкотити" декілька дій

BEGIN TRANSACTION; -- початок транзакції

DECLARE @Id int;

INSERT MyUserName
VALUES
       ('Charlie','Benson')

SET @Id = @@IDENTITY;

INSERT MyUserTell
VALUES
(@Id,'(097)555-55-55');

INSERT MyUserInfo
VALUES (@Id,'14/09/1981');

ROLLBACK TRANSACTION; -- відкат послідовності дій

SELECT * FROM MyUserName;
SELECT * FROM MyUserInfo;
SELECT * FROM MyUserTell;
------------------------------------------------------------
-- Точка збереження транзакції (проміжне збереження транзакції)

BEGIN TRAN

INSERT MyUserName
VALUES
       ('Riley','Kay')
SAVE TRAN SavePointTran; -- точка збереження транзакції (проміжне збереження транзакції)
PRINT 'SAVE POINT';
INSERT MyUserName
VALUES
       ('Tony','Smith')
ROLLBACK TRAN SavePointTran; -- відкат до точки відновлення
--COMMIT TRAN -- зняти коментар після виконання транзакції нижче по коду

SELECT * FROM MyUserName;
GO
------------------------------------------------------------
-- Точка збереження транзакції (проміжне збереження транзакції)

BEGIN TRAN

INSERT MyUserName
VALUES
       ('John','Walker')
SAVE TRAN SavePointTran; -- точка збереження транзакції (проміжне збереження транзакції)
PRINT 'SAVE POINT';
INSERT MyUserName
VALUES
       ('Daniel','Fulton')

ROLLBACK TRAN -- якщо не вказана точка збереження, до якої потрібно зробити відкат,
                        -- то відкат робиться до початку транзакції

SELECT * FROM MyUserName;
GO

------------------------------------------------------------

DROP PROC MyTransactProc;
GO
```

```
-- Процедура запису даних про користувача (ім'я, прізвище, номер телефону, дата народження); є
мальнька помилка, виправте її, будь ласка...
CREATE PROC MyTransactProc
        @FName varchar(25),
        @LName varchar(25),
        @TellN char(14),
        @BDate date
AS
BEGIN
DECLARE @Id int;
BEGIN TRAN -- початок транзакції

                INSERT MyUserName
                VALUES (@FName,@LName);
                SET @Id = @@IDENTITY;

                INSERT MyUserTell
                VALUES (@Id,@TellN);

                INSERT MyUserInfo
                VALUES (@Id,@BDate);

IF EXISTS (SELECT * FROM MyUserName WHERE @FName = FName AND @LName = LName AND IdTest != @Id) --
перевірка на наявність даних про цю людину
                        BEGIN
                                ROLLBACK TRAN; -- відкат транзакції, якщо людина існує
                                RETURN 1;
                        END;

                COMMIT TRAN ; -- успішне завершення транзакції, якщо даний користувач ще не
записаний в БД.
END;

 -- Перевірка результатів роботи процедури --
EXEC MyTransactProc                     -- запис даних
        @Fname = 'Matthew',
        @LName = 'Foster',
        @TellN = '(093)777-77-77',
        @BDate = '17/05/1982';
GO

EXEC MyTransactProc                     -- повторна спроба ввести ті ж дані
        @Fname = 'Matthew',
        @LName = 'Foster',
        @TellN = '(093)777-77-77' ,
        @BDate = '17/05/1982';
GO

EXEC MyTransactProc
        @Fname = 'Harry',
        @LName = 'Allford',
        @TellN = '(093)777-77-77' ,
        @BDate = '17/05/1982';
GO

SELECT * FROM MyUserName;
SELECT * FROM MyUserInfo;
SELECT * FROM MyUserTell;
------------------------------------------------------------
USE AdventureWorks2019;
GO
-- Блок налаштувань для успішної роботи тригерів
SET ANSI_NULLS ON  --  якщо SET ANSI_NULLS встановлено в ON, будь-яка інструкція SELECT, що
використовує речення WHERE column_name = NULL, не поверне жодного запису, навіть якщо в стовпці
column_name є значення NULL
GO
-- Якщо параметри SET ARITHABORT та SET ANSI WARNINGS встановлені в значення ON,
-- ці умови помилок призведуть до завершення запиту. Якщо параметр SET ARITHABORT встановлений в
значення ON,
-- а параметр SET ANSI WARNINGS встановлений в значення OFF, то ці умови помилок призведуть до
завершення пакета
SET ARITHABORT ON -- якщо при виконанні транзакції відбулися помилки, то для цієї транзакції
виконується відкат
GO
SET CONCAT_NULL_YIELDS_NULL ON -- коли параметр SET CONCAT_NULL_YIELDS_NULL встановлений в ON,
об'єднання значення NULL з рядком дає в результаті NULL
GO
```

```
SET QUOTED_IDENTIFIER ON -- якщо параметру SET QUOTED_IDENTIFIER присвоєне значення ON, то
ідентифікатори можна укласти в подвійні лапки і літерали повинні бути розділені поодинокими
лапками
GO
SET ANSI_PADDING ON -- стовпці, які визначені типами даних char, varchar, binary та varbinary,
мають певний розмір
GO
SET ANSI_WARNINGS ON -- якщо значення NULL з'являються в агрегатних функціях, таких як SUM, AVG,
MAX, MIN, STDEV, STDEVP, VAR, VARP або COUNT, то при встановленні значення ON формується
попереджувальне повідомлення
GO
-- Якщо параметр SET NUMERIC_ROUNDABORT має значення ON, після втрати точності у виразі
формується помилка
-- Якщо задане значення OFF, втрати точності не призводять до формування повідомлень про помилки,
а результат заокруглюється з точністю стовпця або змінної, в яких буде збережений
SET NUMERIC_ROUNDABORT OFF
GO
-- http://msdn.microsoft.com/ru-ru/library/ms188048.aspx
-------------------------------------------------------------------------
DROP TRIGGER Sales.SaleaOrderDetailNotDiscontinued; -- видалення тригерів
GO
-------------------------------------------------------------------------
-- Тригер - це обробник, який можна виконати під час виконання операцій - INSERT, UPDATE, DELETE
-- Разом із створенням тригера постійно створюються дві службові таблиці: Inserted і deleted

CREATE TRIGGER Sales.SaleaOrderDetailNotDiscontinued
       ON Sales.SalesOrderDetail
       FOR INSERT, UPDATE
AS
       IF EXISTS
       (
              SELECT *
              FROM Inserted as i -- дані, які вставляються користувачем в таблицю
Sales.SalesOrderDetail(inserted - внутрішня таблиця тригера, де зберігаються рядки, які будуть
перевірятися, і при гарному результаті перевірки потім вставляться в таблицю
Sales.SalesOrderDetail)
              JOIN Production.Product p
              ON i.ProductID = p.ProductID
                WHERE p.DiscontinuedDate IS NOT NULL -- перевіряємо, чи є дата зняття з продажу
       )
BEGIN
       RAISERROR('Товар відсутній на складі!', 1, 16); -- виводимо помилку, якщо товар вже знятий
з продажу
       ROLLBACK TRAN
END
GO

SELECT * FROM Production.Product WHERE ProductID = 777;
UPDATE Production.Product -- створюємо перевірочний рядок, де вказана дата зняття з продажу,
оскільки в таблиці Production.Product таких даних немає
SET DiscontinuedDate = '20-04-2021'
WHERE ProductID = 777;

SELECT * FROM Sales.SalesOrderDetail;
INSERT INTO Sales.SalesOrderDetail -- спробуємо додати це замовлення з товаром, ID якого 777
VALUES
(43659,'4911-403C-98',1,777,1,6.45,0.00,NEWID(),GETDATE());

INSERT INTO Sales.SalesOrderDetail -- спробуємо додати це замовлення з товаром, ID якого 707
VALUES
(43659,'4911-403C-98',1,707,1,6.45,0.00,NEWID(),GETDATE());

-------------------------------------------------------------------------------------------------
-----
--- Застосування тригерів для перевірки дельти (зміни) оновлення ---
--Дельта оновлення - велечина зміни оновленого значення відносно попереднього
SELECT * FROM Production.ProductInventory
GO

DROP TRIGGER Production.ProductIsRationed
GO

CREATE TRIGGER Production.ProductIsRationed
       ON Production.ProductInventory
       FOR UPDATE
AS
IF EXISTS
```

```
(
	SELECT *
	FROM inserted i
	JOIN deleted d
		ON d.ProductID = i.ProductID
		 AND d.LocationID=i.LocationID
	WHERE (d.Quantity - i.Quantity) > d.Quantity/2 --обчислюємо дельта оновлення (різниця між
поточною кількістю товару та тією, що залишиться після продажу)
		 AND d.Quantity - i.Quantity > 0
)

BEGIN
	RAISERROR('Не можливо купувати більше 50 відсотків від кількості товарів, що залишились
',1,2)
	ROLLBACK TRAN
END
GO


SELECT * FROM Production.ProductInventory WHERE ProductID = 2 AND LocationID = 6


UPDATE Production.ProductInventory
SET Quantity = 150 -- намагаємось продати 318-150 = 168 -- більше 50% відсотків
WHERE ProductID = 2
	AND LocationID = 6

UPDATE Production.ProductInventory
SET Quantity = 168 -- намагаємось продати 318-168 = 150 -- менше 50% відсотків
WHERE ProductID = 2
	AND LocationID = 6

UPDATE Production.ProductInventory
SET Quantity = 318
WHERE ProductID = 2
	AND LocationID = 6
GO
------------------------------------------------------------------------
---відміна дії тригера на таблиці без його видалення---
ALTER TABLE Production.ProductInventory
DISABLE TRIGGER ALL ; -- відключаємо тригери
GO

UPDATE Production.ProductInventory
SET Quantity = 150 --намагаємось продати 318-150 = 168 -- більше 50% відсотків
WHERE ProductID = 2
	AND LocationID = 6
GO

UPDATE Production.ProductInventory
SET Quantity = 318
WHERE ProductID = 2
	AND LocationID = 6
GO


ALTER TABLE Production.ProductInventory
ENABLE TRIGGER ALL ; -- включаємо тригер
GO


UPDATE Production.ProductInventory
SET Quantity = 150 --намагаємось продати 318-150 = 168 -- більше 50% відсотків
WHERE ProductID = 2
	AND LocationID = 6
GO

---Production.ProductIsRationed - реалізовано правильно, але не оптимізовано! --
------------------------------------------------------------------------------

---- Правильно так: -----
ALTER TRIGGER Production.ProductIsRationed -- ALTER TRIGGER дозволяє змінити існуючий тригер, не
видаляючи його
		ON Production.ProductInventory
	FOR UPDATE
AS
```

```
	IF UPDATE(Quantity) -- Функція UPDATE() - повертає TRUE або FALSE в залежності від того,
чи здійснюється оновлення конкретного стовпця
	BEGIN
		IF EXISTS
		(
			SELECT *
			FROM inserted i
			JOIN deleted d
				ON d.ProductID = i.ProductID
				 AND d.LocationID=i.LocationID
			WHERE (d.Quantity - i.Quantity) > d.Quantity/2 --обчислюємо дельта
обновлення (різниця між поточною кількістю товара і скільки залишиться після продажу)
				 AND d.Quantity - i.Quantity > 0
		)
		BEGIN
			RAISERROR('Не можливо купувати більше 50 відсотків від кількості товарів,
що залишились ',1,16)
			ROLLBACK TRAN
		END
	END
GO
```

Завдання для виконання

Загальні завдання

Завдання 1

Вивчити основні конструкції та поняття, розглянуті на занятті.

Завдання 2

Зайти в створену раніше БД DreamHomeDB, яка складається з 5 таблиць: Staff, Property_for_Rent, Renter, Qwner, Viewing. Створити всі види тригерів на цих таблицях, задавши перевірки на правильність даних, які вводяться.

Branch	(Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff	(Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent	(Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter	(Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner	(Ono, FName, LName, Address, Tel_No)
Viewing	(Rno, Pno, Date, Comment)

Завдання 3

Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Завдання 4

Зайдіть на сайт MSDN.

Використовуючи механізми MSDN, знайдіть самостійно опис теми по кожному прикладу, який був розглянутий на занятті з даної теми, так, як це представлено нижче, в розділі «**Рекомендовані ресурси**», опису даного комп'ютерного практикуму. Збережіть посилання та дайте їм короткий опис.

Індивідуальні завдання

Варіант 1

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Бібліотека», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 2

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Університет», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 3

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Виробництво», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 4

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Автомайстерня», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 5

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Сесія», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 6

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Керування проєктом», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 7

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Поліклініка», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 8

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Телефонізація», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 9

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Спорт», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 10

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Сільскогосподарські постачання», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 11

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Міський транспорт», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 12

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Географія», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 13

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Управління будинком», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 14

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Аеропорт», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 15

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Ринок ПК», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 16

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Деканат», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 17

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Зоопарк», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 18

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Шахи», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 19

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Судноплавство», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 20

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Автотранспортне підприємство», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 21

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Наукові конференції», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 22

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Програмні продукти», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 23

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Театр», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 24

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Довідникова аптек», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 25

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Ресторан», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 26

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Формула-1», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 27

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Тюнінг автомобілей», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 28

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Археологія», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 29

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Тур-фірма», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Варіант 30

Транзакції та тригери:

1. Створити всі види тригерів на таблицях БД, що розробляється за напрямком «Телемовлення», задавши перевірки на правильність даних, які вводяться.
2. Виконати ряд записів, змін та видалень в контексті 1-ої транзакції. Прослідкувати за правильністю роботи тригера і відката транзакцій при невірному форматі введення або зміни даних.

Рекомендовані ресурси

MSDN: Транзакції
https://msdn.microsoft.com/ru-ru/library/ms174377.aspx

MSDN: Тригери
https://msdn.microsoft.com/ru-ru/library/ms189799.aspx

Контрольні питання

1. Що таке транзакції?
2. Що таке тригер?
3. Що таке точка збереження?
4. Що таке відкат транзакції?
5. Які види тригерів вам відомі?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Дейт К. Введение в системы баз данных : пер. с англ. / К. Дейт. – 8-е изд. – М., СПб.: Вильямс, 2005. – 1328 с.

2. Мейер Д. Теория реляционных баз данных / Д. Мейер. – М., Мир, 1987. – 608 с.

3. Кузнецов С.Д. Основы баз данных / С.Д. Кузнецов. – 2-е изд. – Москва: Бином, 2007. – 251 с.

4. Кодд Э.Ф. Реляционная модель данных для больших совместно используемых банков данных / Э.Ф. Кодд // СУБД. – 1995. – № 1.

5. Чемберлин Д. Анатомия объектно-реляционных баз данных / Д. Чемберлин // СУБД. – 1998. – №. 1-2.

6. Бойко В.В. Проектирование баз данных информационных систем / В.В. Бойко, В.М. Савинков. – М.: Финансы и статистика, 1989. – 351 с.

7. Васкевич Д. Стратегии клиент/сервер / Д. Васкевич. – Киев: Диалектика, 1996. – 384 с.

8. Кузнецов С.Д. Стандарты языка реляционных баз данных SQL: краткий обзор / С.Д. Кузнецов // СУБД. – 1996. – №2. – С.6-36.

9. Date C. J. Databases, Types, and the Relational Model. The Third Manifesto / C. J. Date, Hugh Darwen. – 3th edition. – Addison Wesley, 2006. – 572 p.

10. Чен П. Модель "сущность-связь" - шаг к единому представлению о данных / П. Чен //СУБД. – 1995. – №3. – С.137-158.

11. Диго С.М. Проектирование и использование баз данных / С.М. Диго. – М.: Финансы и статистика, 1995. – 208 с.

12. MSDN: Microsoft SQL Server. [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/en-in/sql-server/sql-server-2019 – Дата доступу: 17.03.2021.

13. MSDN: SQL Management Studio. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/sql/ssms/download-sql-

server-management-studio-ssms?view=sql-server-ver15 – Дата доступу: 14.05.2021.

14. MSDN: Електронна документація по SQL Server. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/uk-ua/sql/?view=sql-server-ver15 – Дата доступу: 17.05.2021.

15. Справочник по Transact-SQL (Transact-SQL). [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/library/ms189826(v=sql.90).aspx – Дата доступу: 17.05.2021.

Додаток А

ЗРАЗОК ОФОРМЛЕННЯ РЕЗУЛЬТАТІВ ВИКОНАННЯ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

кафедра інформаційно-вимірювальних технологій

Комп'ютерний практикум 2

на тему «Запити. Маніпулювання даними»

з дисципліни «Програмування баз даних»

Виконав: студент групи ПА-91

Петренко А.О.

Перевірив: Добролюбова М.В.

Київ -2021

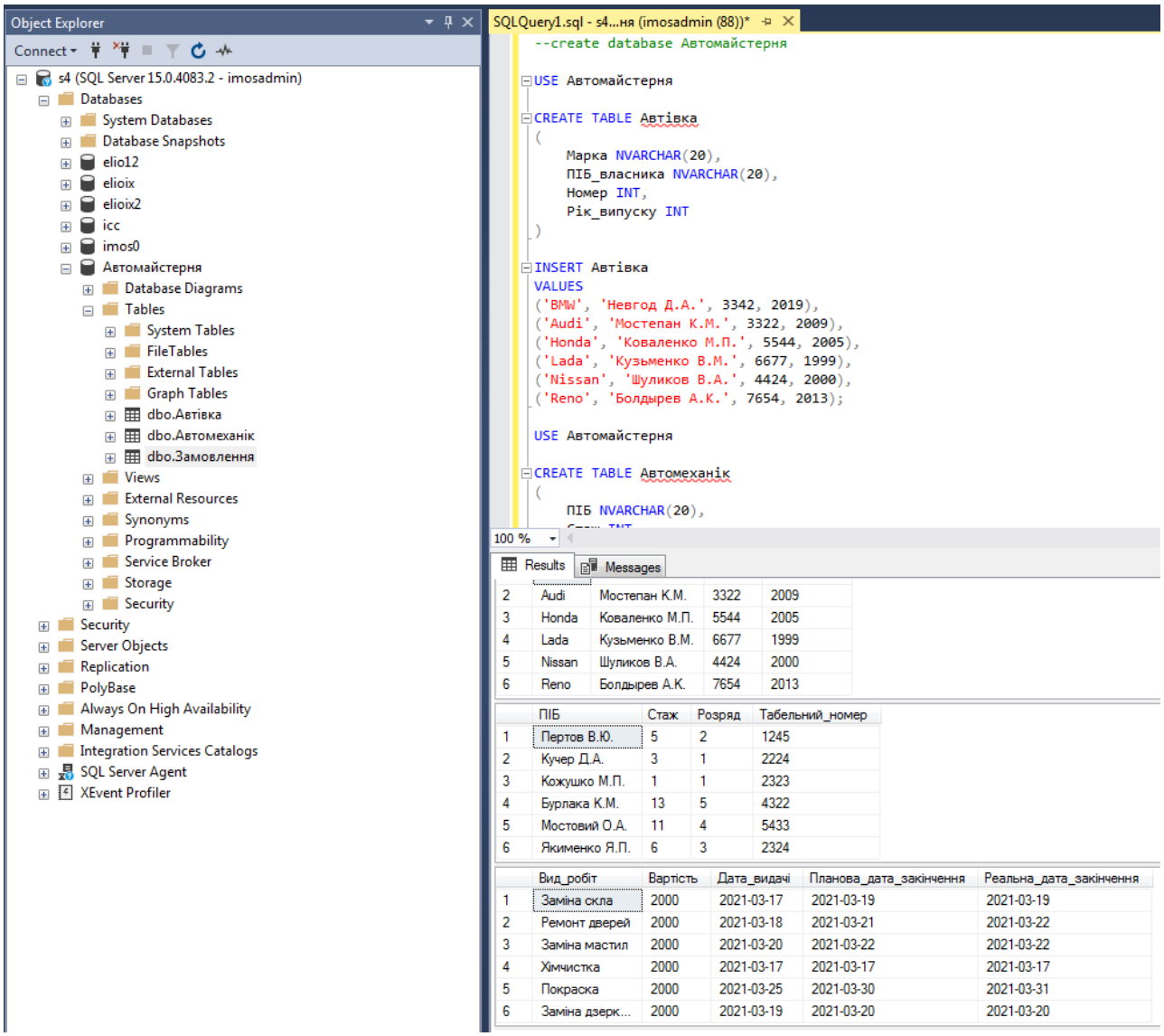
Варіант №4

Індивідуальне завдання:

1. Записати рядок в таблиці Автівка, Замовлення, Автомеханік за допомогою оператора INSERT.
2. Вибрати всі дані таблиці Автівка. Вибрати дані стовпця ПІБ.
3. Вибрати дані стовпців ПІБ, Стаж таблиці Автомеханік, де дані впорядковані по стовпцю Стаж за спаданням.
4. Визначити автомеханіків, стаж роботи яких перевищує 5 років.

Результат виконання програми:

Завдання 1:



Завдання 2:

```
SELECT * FROM Автівка
SELECT ПІБ_власника FROM Автівка
```

100 %

Results | Messages

	ПІБ	Стаж
1	Бурлака К.М.	13
2	Мостовий О.А.	11
3	Якименко Я.П.	6
4	Пертов В.Ю.	5
5	Кучер Д.А.	3
6	Кожушко М.П.	1

Завдання 3:

```
SELECT ПІБ, Стаж
FROM Автомеханік
ORDER BY Стаж DESC;
```

100 %

Results | Messages

	ПІБ	Стаж
1	Бурлака К.М.	13
2	Мостовий О.А.	11
3	Якименко Я.П.	6
4	Пертов В.Ю.	5
5	Кучер Д.А.	3
6	Кожушко М.П.	1

Завдання 4:

```
SELECT * FROM Автомеханік
WHERE Стаж > 5
```

100 %

Results | Messages

	ПІБ	Стаж	Розряд	Табельний_номер
1	Бурлака К.М.	13	5	4322
2	Мостовий О.А.	11	4	5433
3	Якименко Я.П.	6	3	2324