Методичні вказівки до лабораторної роботи №2
з дисципліни
"Організація баз даних та знань"
на тему
"Використання мов SQL DDL, SQL DML для створення структури (схеми) БД і заповнення її даними в СУБД MS SQL SERVER"

Методичні вказівки

МОВА SQL DML

Оператор INSERT INTO.
Оператор додає нові записи в таблицю.

INSERT INTO ім'я_таблиці [(ім'я_стовпця[,ім'я_стовпця ...])]
VALUES (значення[, значення ...])

Додає в таблицю рядок, присвоюючи зазначеним полям перелічені значення.

INSERT INTO ім'я_таблиці [(ім'я_стовпця[,ім'я_стовпця ...])]
SELECT ...

Додає в таблицю рядки, сформовані оператором **SELECT**.

Приклад 1.
INSERT INTO Журнал (НазваЖурналу, НомерЖурналу, РікЖурналу)
VALUES ("Новий мир", 4, 1987)**;**

Оператор DELETE FROM

DELETE FROM ім'я_таблиці [**WHERE** умова]

Оператор видаляє із зазначеної таблиці записи відповідно до заданої умови. За відсутності речення **WHERE** видаляються всі записи.

Приклад 2.
DELETE FROM Журнал **WHERE** НазваЖурналу="Новий мир";

Оператор UPDATE
UPDATE ім'я_таблиці
SET ім'я_стовпця = значення|(**SELECT...**)
[,ім'я_стовпця = значення|(**SELECT...**)...].
[**WHERE** умова]

Встановлює значення для зазначених стовпців у рядках таблиці, що відповідають умові відбору. Значення може бути отримано як результат вкладеного запиту.

Приклад 3 (збільшення мінімального розміру оплати праці до 1200).
UPDATE Зарплата **SET** Оклад = 1200 **WHERE** Оклад < 1200**;**

МОВА SQL DDL

Оператор CREATE TABLE.

Оператор створює таблицю із заданим ім'ям і набором стовпців.

Спрощена форма (мінімальна):
CREATE TABLE ім'я_таблиці (
ім'я_стовпця **тип_даних**[**(розмір)**]
[,ім'я_стовпця **тип_даних** [**(розмір)**] ...])

Назви та опис основних типів даних, допустимих при створенні таблиць, наведено в додатку А.

Приклад 4. Створюється таблиця з іменем "МояТаблиця", яка містить три стовпці (поля) з іменами "Поле1", "Поле2" і "Поле3".
CREATE TABLE МояТаблиця1
(Поле1 **INT IDENTITY (1,1),**
Поле2 **INT,**
Поле3 **CHAR(3)** **);**

INDENTITY(1,1) задає ідентифікатор із початковим значенням 1 і кроком приросту 1.
Якщо необхідно скинути значення лічильника, то для цього існує команда

DBCC CHECKIDENT (ім'я_таблиці, **RESEED**, нове_старт_значення)

Обмеження.
Розрізняють **прості** (на стовпець) і **складові** (табличні) обмеження. Прості обмеження задаються після визначення атрибутів конкретних стовпців, складові вказуються окремо і можуть застосовуватися до кількох стовпців. Усі обмеження можуть бути задані також візуальними засобами - у конструкторі таблиць.
Узагальнена форма оператора CREATE TABLE:
CREATE TABLE ім'я_таблиці
({<визначення стовпця>} [, ...n]

[< обмеження таблиці> [, ...n]])

Спрощена форма оператора CREATE TABLE:

CREATE TABLE ім'я_таблиці (
ім'я_стовпця **тип**[**(розмір)**] [**обмеження**]
[,ім'я_стовпця **тип**[**(розмір)**] [**обмеження**] ...]
[,**PRIMARY KEY** (ім'я_стовпця [, ім'я_стовпця ...])]
[,**FOREIGN KEY** (ім'я_стовпця [, ім'я_стовпця ...])
REFERENCES ім'я_таблиці [(ім'я_стовпця [, ім'я_стовпця])]
[**ON UPDATE CASCADE** | **SET NULL**] [
ON DELETE CASCADE | **SET NULL**]])

Спрощена форма оператора CREATE TABLE, із завданням імен для обмежень (CONSTRAINT):

CREATE TABLE ім'я_таблиці (
ім'я_стовпця **тип**[**(розмір)**] [**CONSTRAINT** ім'я_обмеження **обмеження** [... обмеження]]
[,ім'я_стовпця **тип**[**(розмір)** [**CONSTRAINT** ім'я_обмеження **обмеження**] [...
обмеження]]]
[, **CONSTRAINT** ім'я_обмеження **PRIMARY KEY** (ім'я_стовпця [, ім'я_стовпця ...])]
[,**CONSTRAINT** ім'я_обмеження **FOREIGN KEY** (ім'я_стовпця [, ім'я_стовпця ...])
REFERENCES ім'я_таблиці [(ім'я_стовпця [, ім'я_стовпця])]
[**ON UPDATE CASCADE | SET NULL | SET DEFAULT | NO ACTION**]
[**ON DELETE CASCADE | SET NULL | SET DEFAULT | NO ACTION**]])

Більш повний синтаксис оператора **CREATE TABLE** розглянуто в лекційному матеріалі.

Нижче наводяться пояснення щодо обмежень стовпців:

- **NULL** або **NOT NULL** (відповідно дозволяє або забороняє поміщати порожні значення в цей стовпець);
- **UNIQUE** (накладає вимогу унікальності значень. Для будь-яких двох рядків таблиці значення в цьому стовпці не можуть бути однаковими);
- **PRIMARY KEY** (оголошує стовпець первинним ключем таблиці. Має той самий сенс, що й завдання ключового поля в конструкторі таблиць у СУБД Access);
- **REFERENCES** ім'я_таблиці [(ім'я_стовпця)]
  [**ON UPDATE CASCADE | SET NULL | SET DEFAULT | NO ACTION**]
  [**ON DELETE CASCADE | SET NULL | SET DEFAULT | NO ACTION**]

  Оголошує цей стовпець зовнішнім ключем, який пов'язаний із заданим полем заданої таблиці. Поле, з яким встановлюється зв'язок, має бути ключовим, або, принаймні, унікальним (бути потенційним ключем). Необов'язкові атрибути **ON UPDATE** і **ON DELETE** можуть набувати значень **CASCADE, SET NULL, SET DEFAULT**, **NO ACTION.** Значення **NO ACTION** встановлюється за замовчуванням.

Приклад 5.

CREATE TABLE МояТаблиця2 (
Поле1 **INT PRIMARY KEY**,
Поле2 **INT UNIQUE NOT NULL**,

Поле3 **CHAR(7) NOT NULL,**
Поле4 **INT NOT NULL REFERENCES** МояТаблиця1(Поле1) **ON UPDATE CASCADE ON DELETE NO ACTION);**

Розглянемо тепер обмеження на кілька полів (складові обмеження):
- PRIMARY KEY (ім'я_стовпця [, ім'я_стовпця ...]) - оголошує первинний ключ таблиці. У дужках перераховуються стовпці, що входять до первинного ключа;
- FOREIGN KEY (ім'я_стовпця [, ім'я_стовпця ...])
REFERENCES ім'я_таблиці [(ім'я_стовпця [, ім'я_стовпця ...])]
[**ON UPDATE CASCADE | SET NULL**]
[**ON DELETE CASCADE | SET NULL**]
оголошує поля, перелічені в дужках, зовнішнім ключем. **Речення REFERENCES** вказує ім'я таблиці та її поля, з якими встановлюється зв'язок.

Приклад 7.
Нижче наведено оператори, що створюють три пов'язані таблиці:
CREATE TABLE Клієнт
(КодКлієнта **INT IDENTITY (1,1) PRIMARY KEY**,
Прізвище **VARCHAR(20) NOT NULL**,
Ім_я **VARCHAR(20) NOT NULL**,
По_батькові **VARCHAR(20) NOT NULL;**

CREATE TABLE Журнал
(КодЖурнала **INT PRIMARY KEY**,
НазваЖурналу **VARCHAR(50) NOT NULL**);

CREATE TABLE Підписка
(КодКлієнта **INTEGER REFERENCES** Клієнт(КодКлієнта),
КодЖурналу **INTEGER REFERENCES** Журнал (КодЖурналу),
PRIMARY KEY (КодКлієнта, КодЖурналу) **);**

Таблиці відповідають схемі даних, наведеній нижче (рис. 2.1):

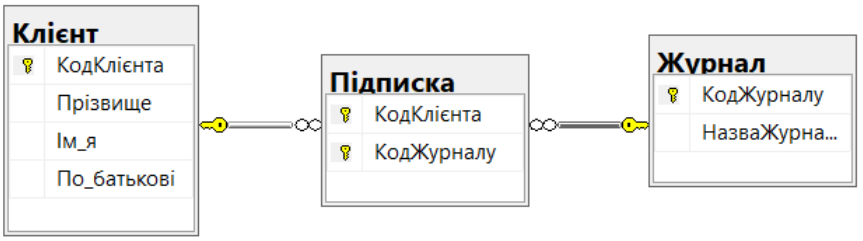


Рисунок 2.1

У СУБД MySQL, PostgreSQL, MariaDB є можливість перед створенням таблиці перевірити, чи існує така таблиця

CREATE TABLE IF NOT EXISTS mytable (id **INT**);

У СУБД MS SQL SERVER, ORACLE така можливість не підтримується. Однак можна її реалізувати попередньою перевіркою, наприклад (SQL SERVER):

```
if not exists (select * from sysobjects where name='Підписка' and type='U')
CREATE TABLE Підписка
(КодКлієнта INTEGER REFERENCES Клієнт(КодКлієнта),
КодЖурналу INTEGER REFERENCES Журнал(КодЖурналу),
PRIMARY KEY (КодКлієнта, КодЖурналу) );
```

де sysobjects - таблиця системних об'єктів, type='U' - ознака об'єкта типу таблиця.

Оператор DROP TABLE.

Видаляє таблицю із заданим ім'ям.

DROP TABLE ім'я_таблиці ;

Видаляє таблицю, перевіряючи попередньо, чи існує така таблиця.

DROP TABLE IF EXISTS ім'я_таблиці

Оператор ALTER TABLE

Застосовується для модифікації структури таблиці.
Нехай існує таблиця TestMy, задана скриптом

```
CREATE TABLE TestMy
(col1 INT NOT NULL, col2 INT);
```

Нижче наведено приклади використання оператора ALTER TABLE

Додавання колонки

```
ALTER TABLE TestMy ADD col3 INT;
ALTER TABLE TestMy ADD col3 INT, col4 INT;
ALTER TABLE TestMy ADD col3 INT, col4 as col3*0.1; // SQL Server
```

Видалення колонки

```
ALTER TABLE TestMy DROP COLUMN col3;
```

Редагування властивостей поля

```
ALTER TABLE TestMy ALTER COLUMN col3 VARCHAR(10) NOT NULL;
```

Додавання обмеження

```
ALTER TABLE TestMy ADD CONSTRAINT PK_TestMy PRIMARY KEY (col1);
// (якщо col1 NOT NULL)
ALTER TABLE TestMy ADD CONSTRAINT FK_TestMy_TM FOREIGN KEY (col2)
REFERENCES TestMy2(col2);
ALTER TABLE TestMy ADD CONSTRAINT CH_TestMy CHECK (col3 IN('перший',
'другий', 'третій')));
ALTER TABLE TestMy ADD CONSTRAINT Def_TestMy DEFAULT 'перший' FOR col3;
```

Додавання поля разом з обмеженням

```
ALTER TABLE TestMy ADD col2 VARCHAR(20) CONSTRAINT c_col2 Unique NOT
NULL;
ALTER TABLE TestMy ADD col2 VARCHAR(20) CONSTRAINT Def_TesMy_col2
DEFAULT 'перший';
```

Вимкнення/ввімкнення обмеження

SQL Server дає змогу відключати тільки обмеження Foreing key, Check.

Зверніть увагу, що в SQL Server не можна відключити обмеження UNIQUE, PRIMARY KEY або NOT NULL (в Oracle можна).

```
ALTER TABLE Sale NOCHECK CONSTRAINT Sale_FK00;
ALTER TABLE Sale CHECK CONSTRAINT Sale_FK00;
```

Видалення обмеження

```
ALTER TABLE TestMy DROP CONSTRAINT c_col2;
```

Для редагування CONSTRAINT необхідно спочатку виконати його видалення і потім створити.

Порядок виконання роботи

У рамках цієї лабораторної роботи необхідно виконати такі завдання:

Завдання 1.

Тема: Вивчення мов SQL DDL, SQL DML для створення структури (схеми) БД **TravelAgency (ТурАгентство)** і заповнення її даними в MS SQL Server.

Завдання 2.

Тема: Самостійна побудова скриптів DDL SQL для заданої схеми даних.

Далі наведено дії для виконання Завдання 1.

Завдання 1.

1. Запустіть MS SQL Server Management Studio і створіть нову базу даних з іменем **TravelAgency (ТурАгентство)**.

2. Створіть дві таблиці **Country** (Країна) і **TouristAttraction** (Пам'ятка), структура і зв'язок між якими наведено на рисунках 2.2, 2.3 відповідно.

Перевірте, що для вашого вікна запитів обрано БД **TravelAgency** або вкажіть USE **TravelAgency.**

Таблиці слід створювати засобами мови SQL DDL, без використання візуальних інструментів (конструктора).

Типи даних для SQL Server наведено наприкінці файлу методичних вказівок (додаток А).

Під час створення таблиць:

- НЕ використовуйте іменовані обмеження CONSTRAINT;
- усі обмеження вводьте в секції опису полів;
- для створення типу поля **id_country** таблиці **Country** використовуйте тип стандарту ISO INTEGER і відстежте в конструкторі (Design/Проект), що він перетвориться на базовий тип T-SQL (Transact-SQL) INT (рис. 2.4); також відстежте, що INDENTITY(1,1) задає ідентифікатор із початковим значенням 1 та кроком приросту 1 (рис. 2.4).
- під час завдання зовнішнього ключа **НЕ** призначайте режими ON UPDATE, ON DELETE і відстежте, що за замовчуванням вони будуть встановлені як NO ACTION (рис.2.5).

```
create table Country
(id_country integer identity(1,1) primary key,
name_country varchar(20) not null);

create table TouristAttraction
(id_tour_attr int primary key,
name_tour_attr varchar(50) not null,
id_country int not null references Country(id_country)
)
```

Рисунок 2.2

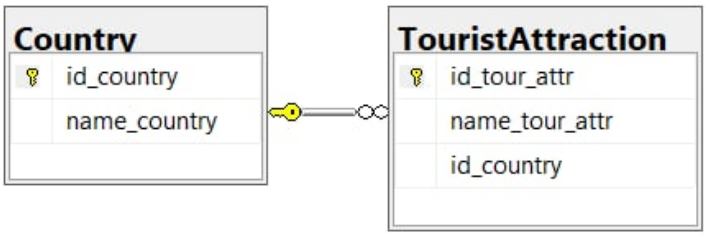


Рисунок 2.3

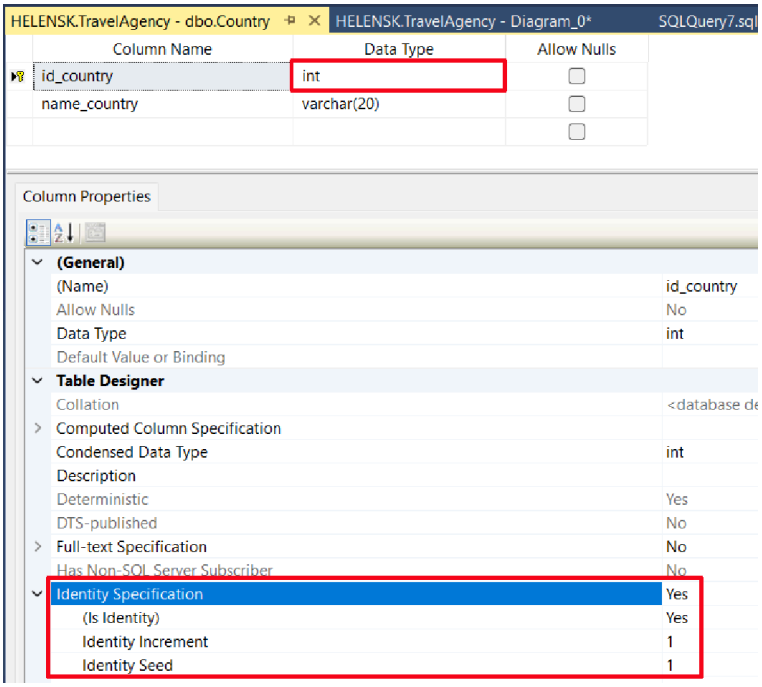
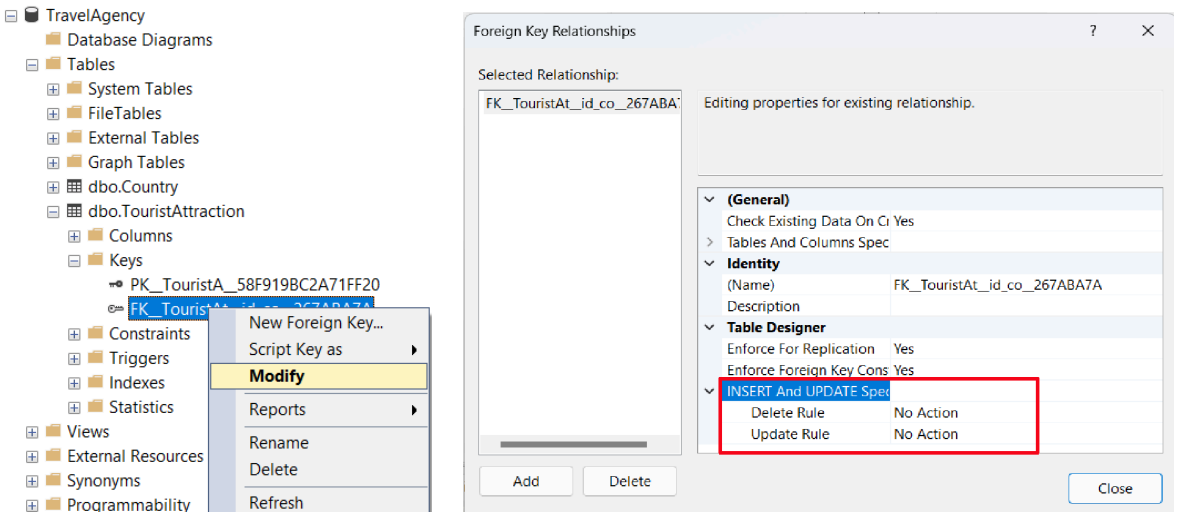


Рисунок 2.4



а б

Рисунок 2.5

Зауваження! Якщо виникає необхідність повторювати кілька разів скрипт створення таблиці (create table), щоб не запускати перед цим скрипт

на видалення таблиці (drop table), можна виконати перед запуском перевірку на існування таблиці (рис. 2.6).

```
if not exists (select * from sysobjects where name='TouristAttraction' and  type='U')
create table TouristAttraction
(id_tour_attr int primary key,
name_tour_attr varchar(50) not null,
id_country int not null references Country(id_country)
)
```

Рисунок 2.6

Запустіть скрипт `select * from sys.sysobjects` , щоб подивитися на системні об’єкти БД TravelAgency (рис.2.7).

	name	id	xtype	uid	info	status	base_schema_ver	replinfo	parent_obj	crdate	ftcatid	schema_ver	stats_schema_ver	type
97	sp_creatediagram	7...	P	1	0	0	0	0	0	2023-11-10 23:57:45.547	0	0	0	P
98	sp_renamediagram	7...	P	1	0	0	0	0	0	2023-11-10 23:57:45.550	0	0	0	P
99	sp_alterdiagram	7...	P	1	0	0	0	0	0	2023-11-10 23:57:45.550	0	0	0	P
100	sp_dropdiagram	8...	P	1	0	0	0	0	0	2023-11-10 23:57:45.550	0	0	0	P
101	fn_diagramobjects	8...	FN	1	0	0	0	0	0	2023-11-10 23:57:45.557	0	0	0	FN
102	TouristAttraction	8...	U	1	0	0	0	0	0	2023-11-11 00:15:48.320	0	0	0	U
103	PK__TouristA__58F919BC98B16ED3	8...	PK	1	0	0	0	0	837578022	2023-11-11 00:15:48.320	0	0	0	K
104	FK__TouristAt__id_co__33D4B598	8...	F	1	0	0	0	0	837578022	2023-11-11 00:15:48.320	0	0	0	F
105	QueryNotificationErrorsQueue	1...	SQ	1	0	0	0	0	0	2009-04-13 12:59:13.483	0	0	0	SQ
106	queue_messages_1977058079	1...	IT	4	0	0	0	0	1977058079	2009-04-13 12:59:13.483	0	0	0	IT
107	EventNotificationErrorsQueue	2...	SQ	1	0	0	0	0	0	2009-04-13 12:59:13.500	0	0	0	SQ

Рисунок 2.7

3. Спробуйте видалити таблицю **Country**, ознайомтеся з помилкою (рис. 2.8).

```
drop table Country;
```

```
0 %
Messages
Msg 3726, Level 16, State 1, Line 14
Could not drop object 'Country' because it is referenced by a FOREIGN KEY constraint.

Completion time: 2023-11-11T00:11:58.3088772+02:00
```

Рисунок 2.8

Таблицю **Country** видалити неможливо, оскільки вона батьківська щодо таблиці **TouristAttraction.** Щоб СУБД дозволила видалити таблицю **Country** необхідно або спочатку видалити дочірню таблицю, або видалити/відключити обмеження зовнішнього ключа.

4. Внесіть зміни до структури таблиці **TouristAttraction**:

4.1 Змініть режим зовнішнього ключа на

- ON UPDATE CASCADE
- ON DELETE CASCADE

Для зміни режиму обмеження FOREIGN KEY ... REFERENCES необхідно його видалити і створити заново. Щоб видалити об'єкт, треба звернутися до нього за іменем. Під час створення таблиці **TouristAttraction** іменований CONSTRAINT не використовувався, ім'я обмеження призначалося сервером. Ім'я обмеження FOREIGN KEY можна подивитися в системній таблиці sys.foreign_keys (рис.2.9) або в списку об'єктів таблиці

TouristAttraction (рис. 2.10). Причому при перестворенні таблиці ім'я буде вже інше.

```
Select *
from sys.foreign_keys;
```

Results | Messages

name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date
FK__TouristAt__id_co__33D4B598	869578136	NULL	1	837578022	F	FOREIGN_KEY_CONSTRAINT	2023-11-11 00:15:48.320	2023-11-11 00:15:48.320

Results | Messages

	is_schema_published	referenced_object_id	key_index_id	is_disabled	is_not_for_replication	is_not_trusted	delete_referential_action	delete_referential_action_desc	update_referential_action	update_referential_action_desc	is_system_na
1	0	581577110	1	0	0	0	0	NO_ACTION	0	NO_ACTION	1

Рисунок 2.9

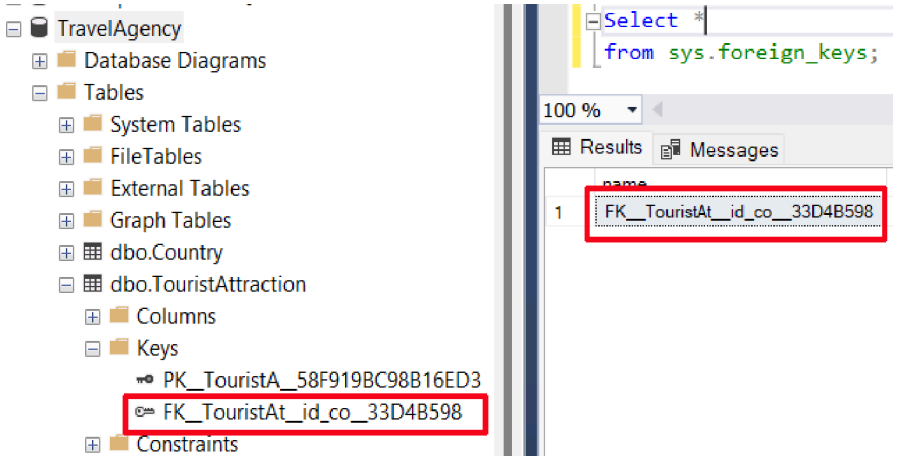


Рисунок 2.10

Перестворіть таблицю **TouristAttraction**, виконавши команди drop table і create table. Оновіть інформацію. Подивіться, що сервер дасть обмеженню нове ім'я.

Видаліть обмеження, звернувшись до нього за знайденим ім'ям, і створіть наново з потрібним режимом і задайте йому призначене для користувача ім'я FK_TourAttrCounty (створіть іменований CONSTRAINT) (рис. 2.11).

```
alter table TouristAttraction drop constraint FK__TouristAt__id_co__33D4B598;

alter table TouristAttraction add constraint FK_TourAttrCounty
foreign key(id_country) references Country(id_country) on update cascade on delete cascade;
```

Рисунок 2.11

Зауваження! Назва обмеження, що видаляється, буде у всіх різна.

Перевірте, що назва і режим обмеження зовнішнього ключа змінився (рис. 2.12).

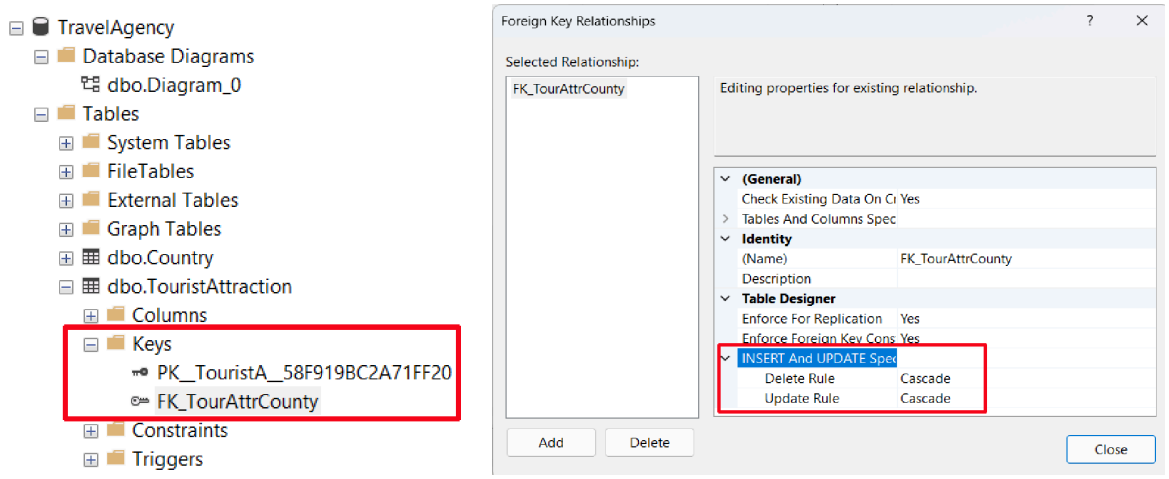


Рисунок 2.12

4.2 Додайте поле **price** типу NUMERIC(10,2) NOT NULL (рис. 2.13);

```
alter table TouristAttraction add price numeric(10,2) not null;
```

Рисунок 2.13

5. Перегляньте діаграму, подивитесь властивості зв'язку (режим тепер "Cascade") (рис.2.14).

Зауваження! Якщо поле price не з’явилося в переліку полів на діаграмі зробіть Refresh, або Disconnect/Connect для Server, та повторіть побудову діаграми.

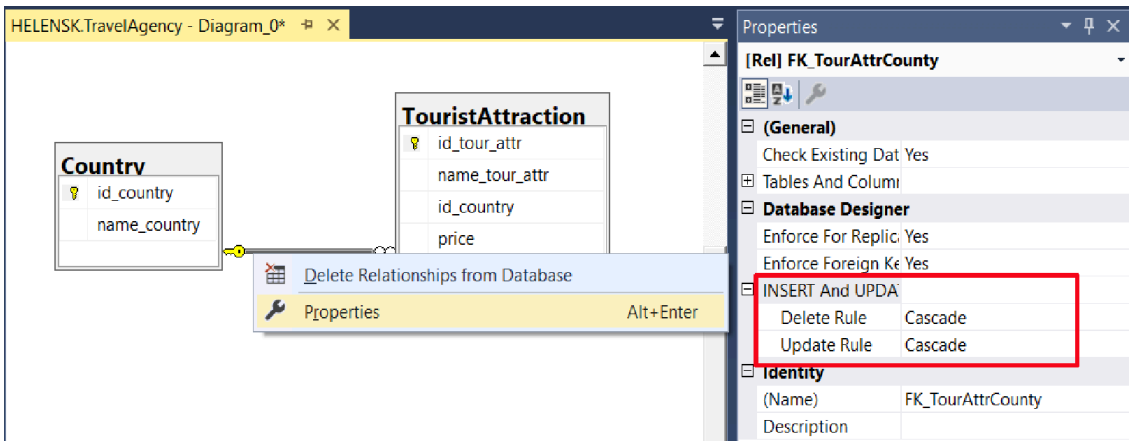


Рисунок 2.14

6. Занесіть у таблицю **Country** назви країн:
- Україна,
- Туреччина,
- Індія,
- Єгипет,
- Польща.

Зауваження!

- синтаксис T-SQL дає змогу заносити одним оператором INSERT ... VALUES одразу кілька рядків (рис. 2.15);

- поле **id_country** буде заповнено автоматично, оскільки поле є лічильником (2.16).

```
insert into Country(name_country)
values('Україна'), ('Туреччина'), ('Індія'), ('Єгипет'), ('Польща');
```

Рисунок 2.15

Для перегляду даних таблиці Country виділіть в Object Explorer таблицю, правою кнопкою миші викличте контекстне меню і виберіть "Edit Top 200 Rows" або просто

select * from Country;

	id_country	name_coun...
▶	1	Україна
	2	Туреччина
	3	Індія
	4	Єгипет
	5	Польща
*	NULL	NULL

Рисунок 2.16

7. Створіть таблицю **Client (**id_client, fio, address, date_birth, date_regist**)** з використанням іменованих обмежень CONSTRAINT (рис.2.17):

id_client - тип INT, лічильник, первинний ключ;

fio - тип VARCHAR(40), поле обов'язкове для заповнення;

address - тип VARCHAR(80),

date_birth - тип DATE, поле обов'язкове для заповнення;

date_regist (дата реєстрації) - тип DATE, поле обов'язкове для заповнення, за **замовчуванням має заповнюватися поточною датою**;

має бути задана перевірка, що значення **date_birth< date_regist**

```
create table Client
(id_client int identity(1,1) constraint PK_Client primary key,
name_client varchar(40) not null,
address_client varchar(80),
date_birth date not null,
date_reg date not null constraint D_ClientDateReg default GetDate(),
constraint Ch_date_birth_reg check (date_birth<date_reg)
)
```

Рисунок 2.17

8 Зробить зміни в структуру таблиці Client.

8.1 Додайте поле **amount** типу NUMERIC(20,2) для зберігання розміру сумарного доходу, який приніс клієнт (рис. 2.18).

```
alter table Client add amount numeric(20,2);
```

Рисунок 2.18

8.2 Додайте поле, яке зберігатиме інформацію про розмір знижки (відсоток знижки): **discount** тип real, а також іменоване обмеження DEFAULT на це поле, щоб поле за замовчуванням приймало значення нуль (рис. 2.19).

```
alter table Client add discount real constraint Def_Client_discount Default 0;
```

Рисунок 2.19

8.3 Змініть тип поля DISCOUNT на INT з вимогою NOT NULL. Оскільки на це поле створено обмеження Def_Client_discount, то просто запуск криптпа на редагування поля (рис. 2.20) призведе до помилки (рис.2.21).

```
alter table Client alter column discount int not null;
```

Рисунок 2.20

Messages

```
Msg 5074, Level 16, State 1, Line 48
The object 'Def_Client_discount' is dependent on column 'discount'.
Msg 4922, Level 16, State 9, Line 48
ALTER TABLE ALTER COLUMN discount failed because one or more objects access this column.
```

Рисунок 2.21

Тому необхідно обмеження Def_Client_discount видалити, відредагувати тип поля, а потім ще раз створити обмеження Def_Client_discount (рис. 2.22).

```
alter table Client drop constraint Def_Client_discount;
alter table Client alter column discount int not null;
alter table Client add constraint Def_Client_discount default 0 for discount;
```

Рисунок 2.22

8.4 Задайте для поля **discount** іменоване обмеження Chek_Client_discount типу CHECK для перевірки, що значення в полі приймають одне зі значень: 0, 5, 10, 15, 20, 25, 50 (рис. 2.23):

```
alter table Client add constraint Ch_Client_discount Check (discount IN (0,5,10,15,20,25, 50));
```

Рисунок 2.23

Самостійно відредагуйте обмеження Check_Client_discount, щоб воно дозволяло приймати значення тільки 0, 5, 10, 15, 20 (видаліть обмеження і створіть заново).

8.5 Самостійно додайте в таблицю **Client** поля **Preference** (для зберігання побажань клієнта, тип поля VARCHAR(1000)) і **Phone** (для зберігання контактного телефону, тип поля CHAR(13), NULL).

8.6 Видаліть поле Preference (рис.2.24):

```
ALTER TABLE Client DROP COLUMN Preference;
```

Рисунок 2.24

8.7 Додайте в таблицю **Client** поле email, тип поля **VARCHAR(30) NOT NULL**, з 2 обмеженнями (рис. 2.25):

- на унікальність (повторів бути не може);
- на вміст символу '@'

```
alter table Client add Email varchar(30) not null
constraint Q_Client_Email unique constraint Ch_Client_Email check (Email like '%@%');
```

Рисунок 2.25

9. Використовуючи мову SQL DDL, приведіть схему БД **TravelAgency** до вигляду, як показано на рисунку 2.25 (додайте асоціативну таблицю **Client_TouristAttract**, що посилається на таблицю **Client** і на таблицю **TouristAttraction**).

Зробіть так, щоб поле date_visit таблиці **Client_TouristAttract** за замовчуванням приймало як значення сьогоднішню дату (DEFAULT GetDate()). Поле total_prise, тип numeric(10,2), NULL міститиме вартість відвідування пам'ятки клієнтом з урахуванням знижки.

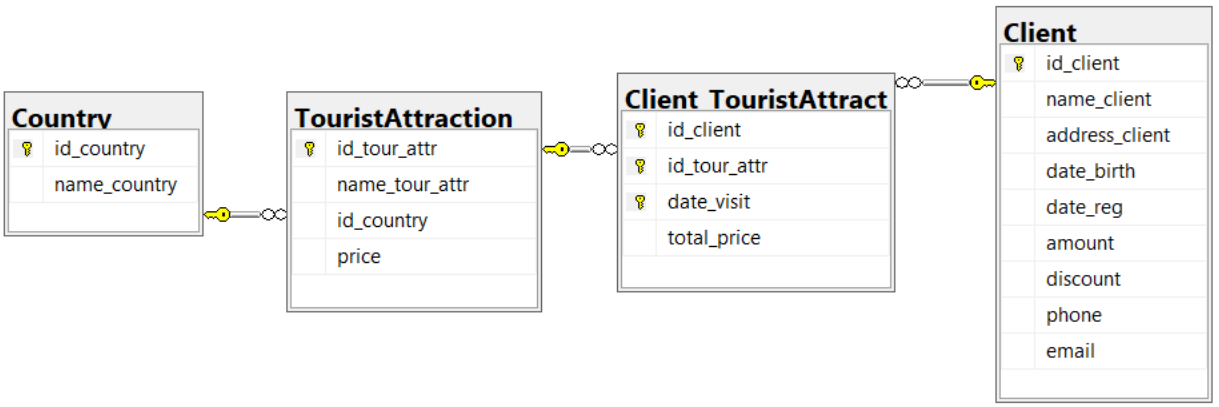


Рисунок 2.25

Зауваження! При заданні режиму зовнішнього ключа між **Client_TouristAttract** і **TouristAttraction** потрібно розуміти, що три сутності **Country**, **TouristAttraction, Client_TouristAttract** утворюють ланцюжок однаково спрямованих зв'язків. Режим зовнішнього ключа (зв'язку) для видалення та оновлення між **TouristAttraction** і **Country** нами було встановлено як **CASCADE**, тобто при видаленні/оновленні кортежів із **Country** (батьківської таблиці) будуть видалятися/оновлюватися кортежі з **TouristAttraction** (дочірньої щодо зв'язку з **Country**). Своєю чергою, поведінка БД під час видалення (оновлення) кортежів із **TouristAttraction** (батьківської, щодо зв'язку з **Client_TouristAttract**) визначатиметься режимом зовнішнього ключа (зв'язку) між **Client_TouristAttract** і **TouristAttraction**.

Якщо задати режиму цього зовнішнього ключа значення за замовчуванням **NO ACTION**, перша частина ланцюжка вимагатиме каскадного видалення, а друга заборонятиме видалення, виникне помилка (рис. 2.26).

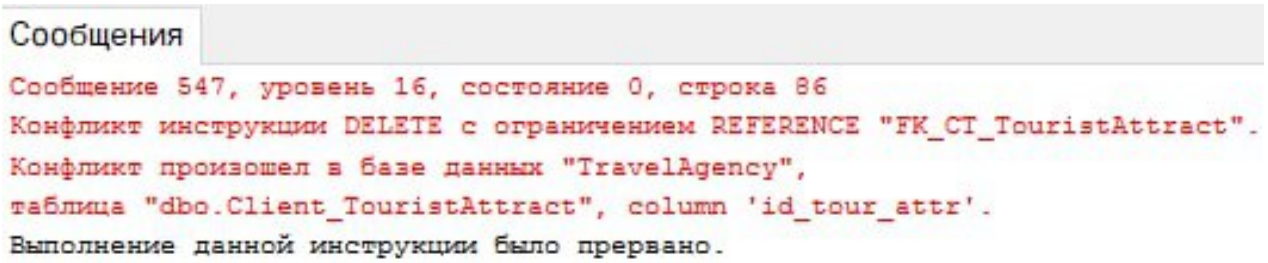


Рисунок 2.26

Режим SET NULL також неприпустимий для зовнішнього ключа між між **Client_TouristAttract** і **TouristAttraction**, тому що зовнішній ключ є частиною первинного і не може приймати значення NULL. Режим **SET DEFAULT** теж не досить доречний для цього випадку. Отже, режим зовнішнього ключа між **Client_TouristAttract** і **TouristAttraction** має бути налаштований або **CASCADE,** щоб увесь ланцюжок було налаштовано на каскадне видалення, або необхідно відредагувати режим першої частини ланцюжка на **NO ACTION,** і тоді можна і для другої частини використовувати режим **NO ACTION.**

Приймемо, як робочий, варіант каскадного режиму для цього ланцюжка.

Скрипт для створення таблиці **Client_TouristAttract** може бути таким, як показано на рисунку 2.27а або 2.27б:

```
create table Client_TouristAttract
(id_client int constraint FK_CT_Client references Client(id_client),
id_tour_attr int constraint FK_CT_TouristAttract references TouristAttraction(id_tour_attr)
                                                    on update cascade on delete cascade,
date_visit date constraint Def_CT_date_visit default GetDate(),
total_price numeric(10,2),
constraint PK_CT primary key (id_client, id_tour_attr, date_visit)
);
```

а

```
create table Client_TouristAttract
(id_client int,
id_tour_attr int,
date_visit date constraint Def_CT_date_visit default GetDate(),
total_price numeric(10,2),
constraint PK_CT primary key (id_client, id_tour_attr, date_visit),
constraint FK_CT_Client foreign key (id_client) references Client(id_client),
constraint FK_CT_TouristAttract foreign key (id_tour_attr) references TouristAttraction(id_tour_attr)
                                                    on update cascade on delete cascade
)
```

б

Рисунок 2.27

Повний DDL скрипт на створення БД **TravelAgency** міститься в додатку Б.

10. Переглянути всі створені об'єкти БД **TravelAgency** можна в системних таблицях і поданнях (рис. 2.28):

```
select * from sys.tables;
select * from sys.key_constraints;
select * from sys.foreign_keys;
select * from sys.check_constraints;
select * from sys.default_constraints;
```

Зверніть увагу (рис.2.28), що всі створені обмеження мають користувацькі назви, окрім первинних ключів для таблиць **Country** і **TouristAttraction**, бо ми при їхньому створенні не вводили і не використовували іменовані CONSTRAINT (рис.2.2).

Співставлення системних таблиць із системними представленнями (Transact-SQL) :

https://docs.microsoft.com/ru-ru/previous-versions/sql/sql-server-2014/ms187997(v=sql.120)?redirectedfrom=MSDN

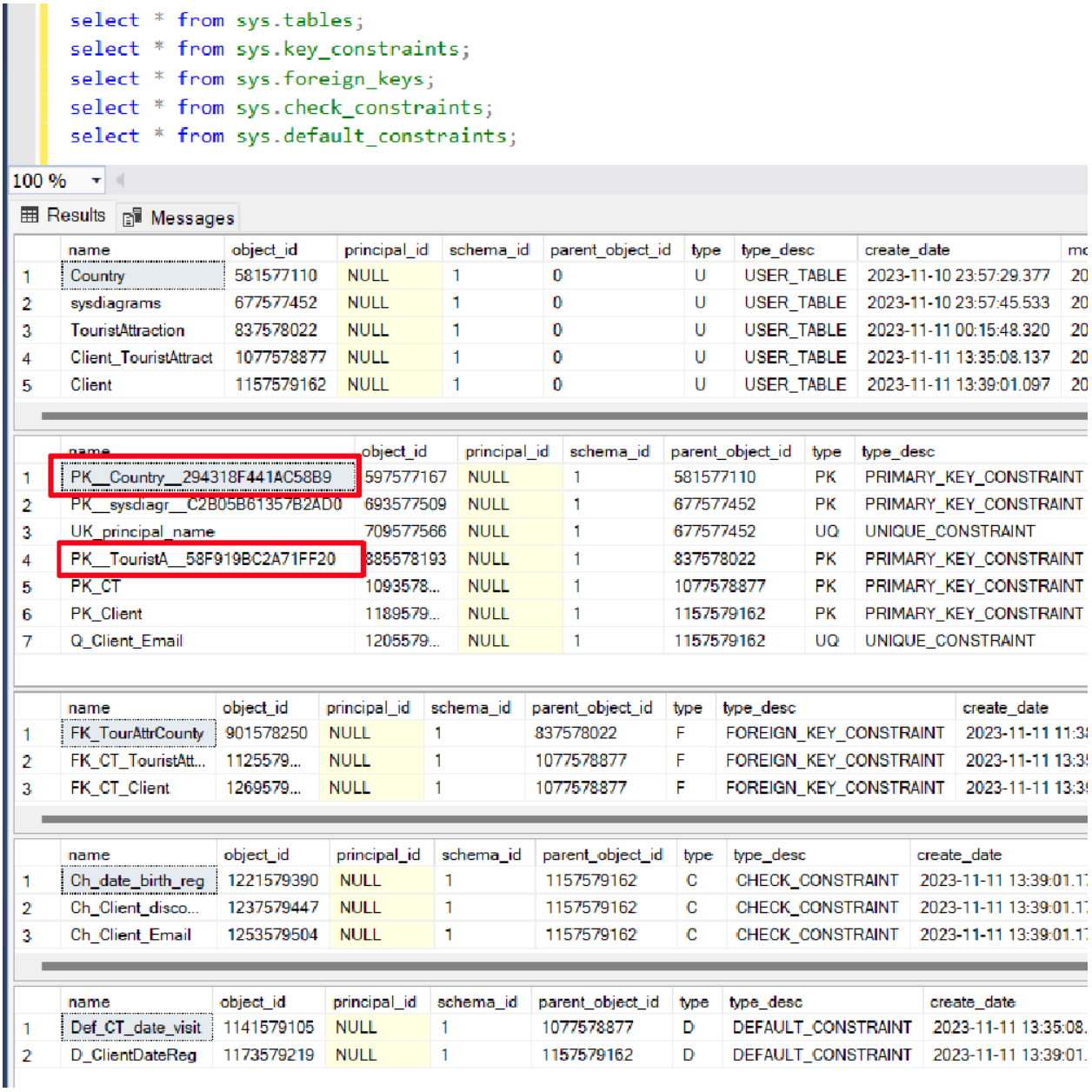


Рисунок 2.28

11. Заповніть даними таблиці **TouristAttraction, Client, Client_TouristAttract** (засобами SQL Server Management Studio, DML (insert, update)).

11.1 Заповніть даними таблицю **TouristAttraction** засобами SQL Server Management Studio (5-10 кортежів).

Для введення (редагування) даних таблиці **TouristAttraction** виділіть в Object Explorer таблицю, правою кнопкою миші викличте контекстне меню і виберіть Edit Top 200 Rows.

11.2 Заповніть таблицю **Client** даними за допомогою команд DML.

Наприклад, занесіть таку інформацію про першого клієнта (2.29):

name_client: Іванов В.І.;
address_client: м. Харків, вул.Наукова 54, кв. 45;
date_birth: 30.12.2000;
phone: +380504567654;

email: vasiliy.ivanov@nure.ua.

```
insert into Client (name_client, address_client, date_birth, phone, email)
values ('Іванченко В.В.','м. Харків, вул.Культури 6б кв 45',
       '2000.12.30', '+380504567654', 'vasiliy.ivanchenko@nure.ua');
```

Рисунок 2.29

Зауваження! Якщо раптом виникне необхідність скинути лічильник, можна виконати команду

DBCC CHECKIDENT (Client, RESEED, 0)

Занесіть самостійно інформацію ще про 9 клієнтів, залишаючи в кількох клієнтів поле phone незаповненим, а адреса кількох клієнтів нехай містить "Харків".

Внесіть email клієнта без @, з повторюваним e-mail(ом), ознайомтеся з повідомленнями про помилки.

11.3 Додайте засобами мови DML SQL у таблицю **Client_TouristAttract** щонайменше 15 записів про відвідування клієнтами визначних пам'яток. Нехай деякі відвідування здійснюються у 2022 році, а інші до 2023 року, частину пам'яток не буде відвідано жодного разу.

12. Внесіть зміни в дані (запит update).

12.1 Внесіть у поле discount розмір знижки (рис. 2.30) (*оператор CASE ще будемо вивчати*):

- 5 (відсотків), якщо клієнт звертався 1 раз;
- 10, якщо клієнт звертався 2 рази;
- 15, якщо клієнт звертався 3 рази;
- 20, якщо 4 і більше разів.

```
update client set discount=
case when (select count(*) from Client_TouristAttract where id_client=client.id_client)=1 then 5
     when (select count(*) from Client_TouristAttract where id_client=client.id_client)=2 then 10
     when (select count(*) from Client_TouristAttract where id_client=client.id_client)=3 then 15
     when (select count(*) from Client_TouristAttract where id_client=client.id_client)>=4 then 20
end;
```

Рисунок 2.30

12.2 **(Завдання на доп. бал)*

Внесіть новий кортеж у таблицю **Client_TouristAttract** з інформацією про відвідування історичної пам'ятки клієнтом, який уже відвідував пам'ятки раніше, і перерахуйте для цього кортежу поле total_price з урахуванням поточної знижки клієнта (поле discount таблиці Client).

13. Побудуйте такі DQL SQL запити (select) до БД **TravelAgency** *(повторення матеріалу)***:**

13.1 Виведіть тих клієнтів, у яких відсутня інформація про телефон.

13.2 Виведіть клієнтів, які проживають у Харкові.

13.3 Підрахуйте середню вартість пам'яток у кожній країні. Вивести назву країни, середню вартість пам'яток. Відсортувати за назвою країни за зростанням, за середньою вартістю за спаданням.

13.4 Підрахувати кількість відвідувань кожної країни у 2022 році. Вивести назву країни та кількість відвідувань.

13.5 Вивести назву пам'яток, які ще жодного разу не відвідували, використовуючи оператори:

- IN і підзапит або ALL і підзапит;
- EXISTS;
- LEFT JOIN або RIGNT JOIN;

13.6 Вивести назву пам'ятки та дату її останнього відвідування.

13.7 Вивести клієнтів, які відвідали найбільшу кількість пам'яток.

13.8 Вивести назву 5-ти найпопулярніших пам'яток (підрахувати кількість відвідувань кожної пам'ятки, відсортувати за кількістю відвідувань за зменшенням і, використовуючи оператор TOP (аргумент), вивести назви перших 5-ти пам'яток).

13.9! Вивести назву країни, кількість визначних пам'яток у країні та кількість клієнтів, які відвідали країну (одним запитом).

13.10! Вивести інформацію про пам'ятки в такому вигляді: назва країни, назва пам'ятки, ціна пам'ятки. Відсортувати результуючий набір за країнами за сумарною вартістю пам'яток у них (за зменшенням), потім за ціною пам'ятки (теж за зменшенням). Тобто країни мають виводитися, починаючи з тієї, в якій сумарна вартість пам'яток найбільша, і закінчуючи країною, де сумарна вартість найменша.

Підказка: у FROM створіть підзапит, що сформує віртуальну таблицю, яка міститиме номер країни та сумарну вартість пам'яток, задайте псевдонім; з'єднайте створену підзапитом таблицю з таблицею Dostoprim; відсортуйте результат.

Завдання 2. Самостійна побудова скриптів DDL SQL для заданої схеми даних

Побудуйте скрипти SQL DDL для створення в SQL Server заданої схеми даних БД AutoRacingSQL (рис. 2.31).

Рекомендовано використання CONSTRAINT.

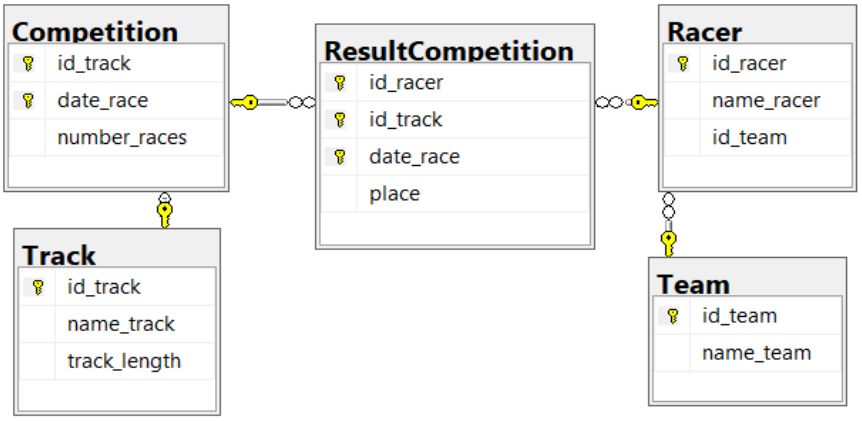


Рисунок 2.31

На захист необхідне вміння:
а) користуватися операторами модифікації даних (insert, update, delete);
б) користуватися операторами створення та модифікації структури БД (create, alter, drop);
в) створювати запити за допомогою оператора select;

Звіт має містити:

1. Усі DDL скрипти зі створення та зміни структури БД **TravelAgency**, DML скрипти для наповнення БД даними
   (з коментарями та номерами завдань із методичних вказівок).
2. DQL SQL запити до БД **TravelAgency** (з коментарями та номерами завдань із методички).
3. DDL скрипти create table для створення структури БД **AutoRacingSQL**

Усі скрипти можуть міститися у файлі одного з форматів .sql, .doc, .pdf (скрипти мають бути відформатовані для зручного читання).

Додаток А
Опис типів даних у T-SQL

Типи даних у MS SQL Server поділяються на такі категорії:

- Точні числа;
- Приблизні числа;
- Символьні рядки;
- Символьні рядки в Юнікоді;
- Дата і час;
- Бінарні дані;
- Інші типи даних.

https://learn.microsoft.com/ru-ru/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16

Точні числа

Найменування типу	**Сховище**	**Опис**
bit	Якщо в таблиці до 8 bit-стовпців 1 байт, якщо від 9 до 16, то 2 байти і так далі.	Може набувати значень 1, 0 або NULL. Часто використовується як тип даних Boolean. Рядкові значення TRUE і FALSE можна перетворити на значення цього типу: TRUE перетворюється на 1, а FALSE на 0.
tinyint	1 байт	Цілі числа від 0 до 255
smallint	2 байти	від -2^15 (-32 768) до 2^15-1 (32 767).
int	4 байти	від -2^31 (-2 147 483 648) до 2^31-1 (2 147 483 647). Це основний цілочисельний тип даних у Microsoft SQL Server.
bigint	8 байт	від -2^63 (-9 223 372 036 854 775 808) до 2^63-1 (9 223 372 036 854 775 807).
numeric (p, s) и decimal (p, s)	Точність: від 1 до 9 = 5 байт; від 10 до 19 = 9 байт; від 20 до 28 = 13 байт; від 29 до 38 = 17 байт.	Тип числових даних із фіксованою точністю та масштабом. numeric і decimal функціонально еквівалентні. p (точність) - максимальна кількість десяткових розрядів числа, що зберігатимуться (як ліворуч, так і праворуч від десяткової коми). Точність може бути значенням у діапазоні від 1 до 38, за замовчуванням 18. s (масштаб) - максимальна кількість десяткових розрядів числа праворуч від десяткової коми. Максимальне число цифр зліва від десяткової коми визначається як p - s (точність - масштаб). Масштаб може бути значенням від 0 до p, за замовчуванням 0. Максимальний розмір сховища залежить від точності. Тип даних numeric і decimal може набувати значення від -10^38+1 до 10^38-1.
smallmoney	4 байти	Тип даних для зберігання грошових значень з точністю до однієї десятитисячної грошової одиниці. Число від -214 748,3648 до 214 748,3647

money	8 байт	Тип даних для зберігання грошових значень з точністю до однієї десятитисячної грошової одиниці. Число від -922 337 203 685 477,5808 до 922 337 203 685 477,5807

Приблизні числа

Найменування типу	Сховище	Опис
float (n)	Залежить від значення n: Від 1 до 24 (7 знаків) = 4 байти; Від 25 до 53 (15 знаків) = 8 байт.	Використовується для числових даних із плаваючою комою. n - це кількість бітів, які використовуються для зберігання мантиси числа у форматі float під час експоненціального подання. n визначає точність даних і розмір для зберігання. Може набувати значення від 1 до 53, за замовчуванням 53. Діапазон значень від -1,79E +308 до 1,79E+308.
real	4 байти	Використовується для числових даних із плаваючою комою. real відповідає в ISO типу float(24). Діапазон значень від -3.40E+38 до 3.40E+38.

Не рекомендується використовувати стовпці з типами float і real у реченні WHERE, оскільки ці типи не зберігають точних значень. Також не рекомендується використовувати float і real у фінансових додатках, в операціях, пов'язаних з округленням. Для цього краще використовувати decimal, money або smallmoney.

Символьні рядки

Найменування типу	Сховище	Опис
char (n)	n байт	Рядок з фіксованою довжиною **НЕ** в Юнікоді, де n довжина рядка (від 1 до 8000). За замовчуванням n = 1, якщо значення n не вказано під час використання функцій CAST і CONVERT, довжина за замовчуванням дорівнює 30.
varchar (n|max)	Розмір займаної пам'яті в байтах = кількість введених символів + 2 байти. Якщо вказати MAX, то максимально можливий розмір = 2^31-1 байт (2 ГБ).	Рядкові дані змінної довжини **НЕ** в Юнікоді, де n довжина рядка (від 1 до 8000). За замовчуванням n = 1, якщо значення n не вказано під час використання функцій CAST і CONVERT, довжина за замовчуванням дорівнює 30.
text	Розмір займаної пам'яті в байтах = кількість введених символів. Максимальний розмір 2^31-1 (2 147 483 647 байт, 2 ГБ).	Рядок змінної довжини **НЕ** в Юнікоді. Є застарілим типом даних, рекомендується використовувати varchar(max).

Символьні рядки в Юнікоді

Найменування типу	Сховище	Опис
nchar (n)	n * 2 байт	Рядок з фіксованою довжиною в Юнікоді, де n довжина рядка (від 1 до 4000). За замовчуванням n = 1, якщо значення n не вказано під час використання у функції CAST, довжина за замовчуванням дорівнює 30.
nvarchar (n|max)	Розмір займаної пам'яті в байтах = кількість введених символів, помножена на 2 + 2 байти. Якщо вказати MAX, то максимально можливий розмір = 2^31-1 байт (2 ГБ).	Рядок змінної довжини в Юнікоді, де n довжина рядка (від 1 до 4000). За замовчуванням n = 1, якщо значення n не вказано під час використання у функції CAST, довжина за замовчуванням дорівнює 30.
ntext	Розмір займаної пам'яті в байтах = кількість введених символів, помножена на 2. Максимальний розмір 2^30 - 1 (1 073 741 823 байт, 1 ГБ).	Рядок змінної довжини в Юнікоді. Є застарілим типом даних, рекомендується використовувати nvarchar(max).

Дата і час

Найменування типу	Сховище	Діапазон	Точність	Опис
дата	3 байти	Від 01.01.0001 до 31.12.9999	1 день	Використовується для зберігання дати.
datetime	8 байт	Від 01.01.1753 00:00:00 до 31.12.9999 23:59:59,997	0,00333 секунди	Використовується для зберігання дати, включно з часом з точністю до однієї трьохсотої секунди.
datetime2	Від 6 до 8 байт (залежно від точності: менше ніж 3 цифри = 6 байт, 3-4 цифри = 7 байт, більше ніж 4 цифри = 8 байт)	Від 01.01.0001 00:00:00.0000000 до 31.12.9999 23:59:59.9999999	100 наносекунд	Розширений варіант типу даних datetime, має ширший діапазон дат і більшу точність у частках секунди (до 7 цифр).
smalldatetime	4 байти	Від 01.01.1900 00:00:00 до 06.06.2079 23:59:00	1 хвилина	Скорочений варіант типу даних datetime, має менший діапазон дат і не має часток секунд.
time [Точність]	Від 3 до 5 байт	Від 00:00:00.0000000 до 23:59:59.9999999	100 наносекунд	Використовується для зберігання часу дня. Точність може бути цілим числом від 0 до 7, за замовчуванням 7 (100 наносекунд, 5 байт). Якщо вказати 0,

				то точність буде до секунди (3 байти).
datetimeoffset [Точність]	Від 8 до 10 байт	Від 01.01.0001 00:00:00.0000000 до 9999-12-31 23:59:59.9999999	100 наносекунд	Використовується для зберігання дати і часу, включно зі зміщенням часової зони щодо універсального глобального часу. Точність визначає кількість знаків у дробовій частині секунди, це значення може бути від 0 до 7, за замовчуванням 7 (100 наносекунд, 10 байт).

Бінарні дані

Найменування типу	Сховище	Опис
binary (n)	n байт	Бінарні дані фіксованої довжини. n - значення від 1 до 8000. Якщо не вказувати n, то значення за замовчуванням 1, якщо не вказати у функції CAST, то 30. Даний тип краще використовувати у випадках, коли розмір даних, які зберігатимуться в стовпці, можна заздалегідь визначити.
varbinary (n|max)	Розмір займаної пам'яті в байтах = фактичний розмір даних + 2 байти. Якщо вказати MAX, то максимально можливий розмір = 2^31-1 байт (2 ГБ).	Бінарні дані зі змінною довжиною. n - значення від 1 до 8000. Якщо не вказувати n, то значення за замовчуванням 1, якщо не вказати у функції CAST, то 30. Даним типом краще користуватися, якщо розмір даних у стовпці заздалегідь визначити важко. Якщо розмір даних перевищує 8000 байт, необхідно використовувати тип varbinary(max).
image	Максимальний розмір до 2^31-1 (2 147 483 647 байт, 2 ГБ).	Бінарні дані зі змінною довжиною. Є застарілим типом даних, рекомендується використовувати varbinary(max).

Інші типи даних: cursor, table, sql_variant, rowversion (timestamp), xml, uniqueidentifier, hierarchyid, просторові типи,

Пріоритети типів даних у T-SQL

У Microsoft SQL Server у випадках, коли оператор об'єднує два вирази з різними типами даних, відбувається неявне перетворення типів, якщо таке перетворення не підтримується, SQL сервер видаватиме помилку. Щоб визначати який тип даних з виразів перетворювати, SQL Server застосовує правила пріоритету типів даних. Тип даних, який має менший пріоритет, буде перетворено на тип даних із більшим пріоритетом. Якщо обидва вирази мають однаковий тип даних, результат операції матиме такий самий тип даних.

У MS SQL Server існує такий пріоритет типів даних:

1) визначені користувачем типи даних (вищий пріоритет);
2) sql_variant
3) xml
4) datetimeoffset
5) datetime2
6) datetime
7) smalldatetime

8) date
9) time
10) float
11) real
12) decimal
13) monez
14) smallmoney
15) bigint
16) int
17) smallint
18) tinyint
19) bit
20) ntext
21) text
22) image
23) timestamp
24) uniqueidentifier
25) nvarchar (включаючи nvarchar(max));
26) nchar
27) varchar (включаючи varchar(max));
28) char;
29) varbinary (включаючи varbinary(max));
30) binary (нижчий пріоритет).

Синоніми типів даних у Microsoft SQL Server

У MS SQL Server для сумісності зі стандартом ISO існують синоніми системних типів даних. Ці синоніми можна використовувати в інструкціях мови Transact-SQL так само, як і відповідні системні типи даних, єдиний момент - після створення об'єкта (*таблиці, процедури*) синоніму призначають базовий тип даних, пов'язаний із цим синонімом, іншими словами, будь-яких ознак, що в інструкції використовували синонім, немає.
Синоніми та відповідні їм системні типи даних подано в таблиці нижче:

Системний тип даних	Синонім типу
varbinary	Binary varying
varchar	char varying
char	character
char(1)	character
char(n)	character(n)
varchar(n)	character varying(n)
decimal	Dec
float	Double precision
real	float[(n)]; n = 1-7
float	float[(n)]; n = 8-15
int	Integer
nchar(n)	national character(n)
nchar(n)	national char(n)
nvarchar(n)	national character varying(n)

nvarchar(n)	national char varying(n)
ntext	national text
rowversion	timestamp

Додаток Б

DDL T-SQL скрипт для створення БД TravelAgency

```
-------------------------------------------
create table Country
(id_country integer identity(1,1) primary key,
name_country varchar(20) not null);
-------------------------------------------
create table TouristAttraction
(id_tour_attr int primary key,
name_tour_attr varchar(50) not null,
id_country int not null constraint FK_TourAttrCounty foreign key(id_country)
    references Country(id_country) on update cascade on delete cascade,
price numeric(10,2) not null
)
-------------------------------------------
create table Client
(id_client int identity(1,1) constraint PK_Client primary key,
name_client varchar(40) not null,
address_client varchar(80),
date_birth date not null,
date_reg date not null constraint D_ClientDateReg default GetDate(),
amount numeric(20,2),
discount int not null constraint Def_Client_discount default 0,
phone char(13),
email varchar(30) not null constraint Q_Client_Email unique constraint
    Ch_Client_Email check (email like '%@%'),
constraint Ch_date_birth_reg check (date_birth<date_reg)
)
-------------------------------------------------
create table Client_TouristAttract
(id_client int constraint FK_CT_Client references Client(id_client),
id_tour_attr int constraint FK_CT_TouristAttraction references
    TouristAttraction(id_tour_attr) on update cascade on delete cascade,
date_visit date constraint Def_CT_date_visit default GetDate(),
total_price numeric(10,2),
constraint PK_CT primary key (id_client, id_tour_attr, date_visit)
);
```