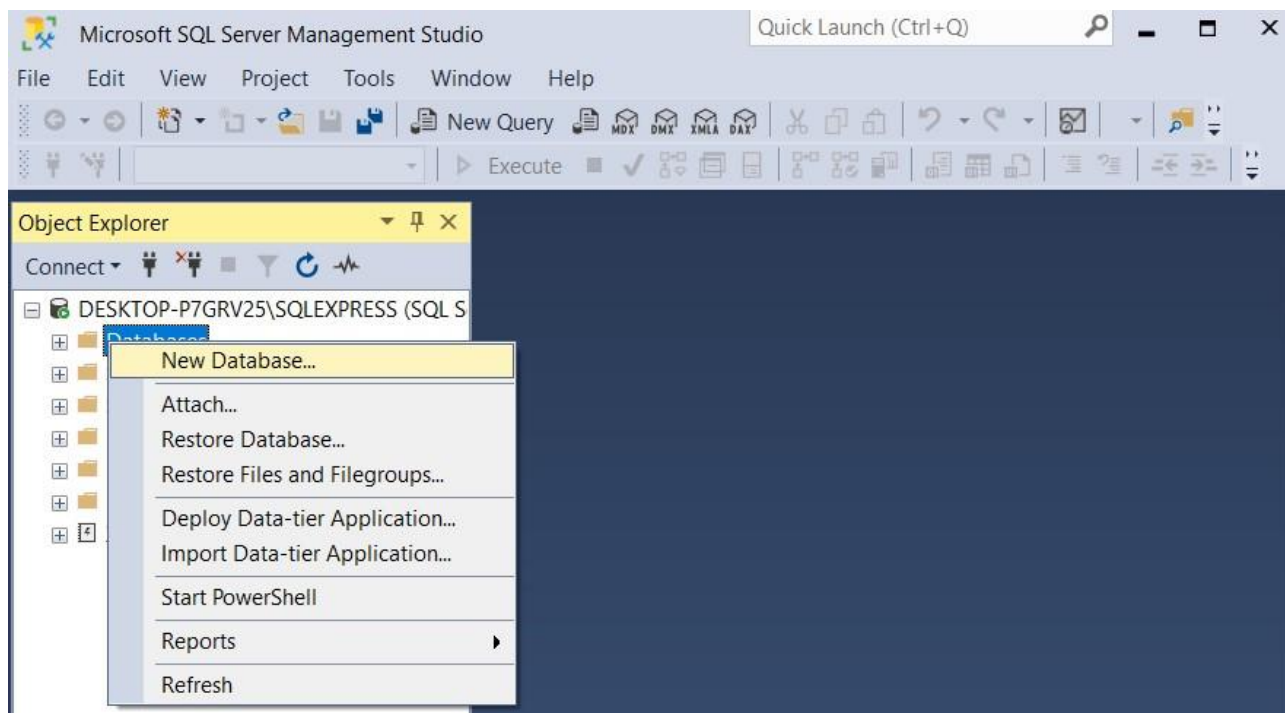
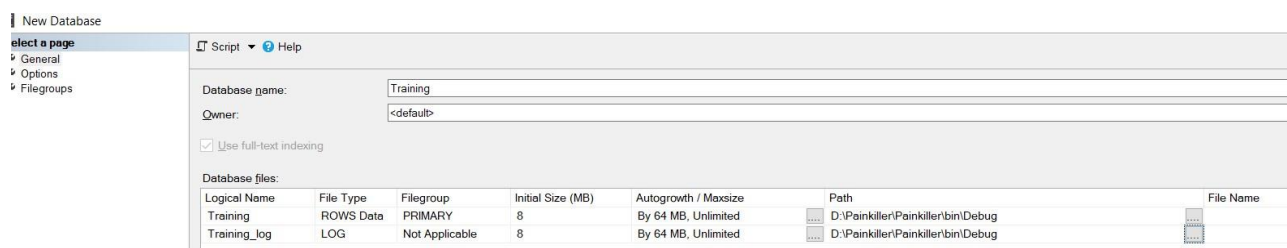


Створення локальної бази даних (БД):

- 1) установіть [SQL Server Management Studio](#)
- 2) при вході у цю програму створюємо порожню БД:
  - а) ПКМ в Object Explorer на Database → New Database



- б) Вводимо ім'я БД та можемо вибрати шлях до неї. Я розмістив її у папці Debug свого проекту з назвою Training.



### 3) Створимо таблиці

У лівому вікні ЛКМ(лівою кнопкою миші) на біля Database, потім на таку ж кнопку біля назви нашої БД. Кладнемо ПКМ(правою кнопкою миші) на Tables → ЛКМ на New і потім на Table. У правому вікні SQL Server Management Studio з'явиться заготовка для створення стовпців таблиці.


Column Name – ім'я стовпця таблиці; Data Type – тип даних стовпця, який ми вказували раніше, коли створювали таблиці у нашому проекті; Allow Null визначає, чи може у цьому стовпці бути порожнє значення. Це, як правило, дозволяється завжди, окрім ключових полів. Ключові поля містять унікальні значення, які призначені для однозначної ідентифікації рядків і для пов'язування цієї таблиці з іншими, тому не можуть містити значення Null. У нижній частині Column Properties цього правого вікна деталізовано інформацію про стовпці, які ми створюємо зверху.

Стовпці ідентифікатори, або ключові поля, як правило, іменують так: до імені таблиці додають суфікс Id. Перший стовпець таблиці – ідентифікатор. Тут значення Null – не дозволено. Цей рядок зробимо ключовим наступним чином: у нижній частині вікна (Column Properties) подано множину можливих властивостей стовпця таблиці. Виберемо у розділі Column Properties властивість Identity Specification і встановимо їй значення Yes. Щоб це зробити потрібно розкрити цей ComboBox і надати властивості (Is Identity) значення Yes. Таким чином ми забезпечили цьому полю унікальне значення в кожному рядку, які будуть надалі створюватись. Для того, щоб це поле стало ключовим, активізуємо його мишкою і в піктографічному меню Table Designer натиснемо іконку з ключиком. Наша таблиця повинна набутися такого вигляду:

DESKTOP-P7GRV25\...ng - dbo.TabTrain			
Column Name	Data Type	Allow Nulls	
TabTrainId	int	<input type="checkbox"/>	
N_pp	int	<input checked="" type="checkbox"/>	
Група_Мязів	nvarchar(50)	<input checked="" type="checkbox"/>	
TabVyd_TrenuvanniaRef	int	<input checked="" type="checkbox"/>	
TabExerciseRef	int	<input checked="" type="checkbox"/>	
TabObtiazhenniaRef	int	<input checked="" type="checkbox"/>	
Положення	nvarchar(255)	<input checked="" type="checkbox"/>	
Max_vaga_kg	int	<input checked="" type="checkbox"/>	
Max_vaga_lb	int	<input checked="" type="checkbox"/>	
K_сть_повторень_з_max_vagoю	int	<input checked="" type="checkbox"/>	
Загальна_к_сть_підходів	int	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

Column Properties	
> Full-text Specification	No
Has Non-SQL Server Subscriber	No
▼ Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

У нас є поля TabVyd\_TrenuvanniaRef, TabExerciseRef та TabObtiazhenniaRef з типом int. Це будуть зовнішні ключі для зв'язку з нашими довідниками – таблицями TabVyd\_Trenuvannia, TabExercise та TabObtiazhennia відповідно. Тобто в нас будуть не назви видів тренувань, вправ та обтяжень, а посилання на рядок у відповідній таблиці. Щоб зберегти таблицю потрібно ЛКМ на  і вводимо назву нашої таблиці у діалоговому вікні. Доцільно зберігати інформацію, яка змінюється не занадто часто, у спеціальних таблицях, які можна назвати довідниками. Логічно зберігати таблиці довідників у БД разом з основною таблицею. У нашому випадку слід створити хоча б два довідники – виробників та груп

товарів. Отже, аналогічно до створення таблиці TabTrain створимо таблиці TabVyd\_Trenuvannia, TabExercise і TabObtiazhennia. У таблиці TabVyd\_Trenuvannia створимо два стовпці. Перший стовпець латинкою, згідно домовленості про назви ключових полів. Зберігаємо нашу таблицю.

Column Name	Data Type	Allow Nulls
TabVyd_TrenuvanniId	int	<input type="checkbox"/>
Вид_тренування	nvarchar(255)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Properties	
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1



Аналогічно створимо таблиці TabExercise і TabObtiazhennia.

Column Name	Data Type	Allow Nulls
TabExercisId	int	<input type="checkbox"/>
Вправа	nvarchar(255)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
TabObtiazhennId	int	<input type="checkbox"/>
Обтяження	nvarchar(255)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

#### 4) Встановимо зв'язки між таблицями

Для цього у переліку таблиць ПКМ на таблицю TabTrain і виберемо пункт Design. Натиснемо ЛКМ на , що ми можемо знайти справа на панелі інструментів. Відкриється вікно для створення зв'язків, у якому ЛКМ на Add → праворуч рядка Tables And Columns Specification клацнемо ЛКМ на . Ліворуч під текстом Primary key table у ComboBox виберемо таблицю TabVyd\_Trenuvannia. Під чим з'явиться ще один ComboBox, який містить перелік полів вибраної таблиці. Виберемо поле TabVyd\_TrenuvanniaRef

Tables and Columns

Relationship name:

Primary key table:

Foreign key table:

OK Cancel

Так чином створено зв'язок між таблицями TabVyd\_Trenuvannia та TabTrain через первинний ключ TabVyd\_TrenuvanniaId та тепер вже зовнішній ключ TabVyd\_TrenuvanniaRef, тобто встановлено відношення один до багатьох. Так само пов'яжемо таблиці TabExercise і TabTrain через поля TabExerciseRef та TabExerciseId, а також TabObtiazhennia і TabTrain через TabObtiazhenniaId та TabObtiazhenniaRef. Отримаємо:

Foreign Key Relationships

Selected Relationship:

Editing properties for existing relationship.

☒ **(General)**  
 Check Existing Data On Cr Yes  
 > Tables And Columns Spec

☒ **Identity**  
 (Name) FK\_TabTrain\_TabVyd\_Trenuvannia  
 Description

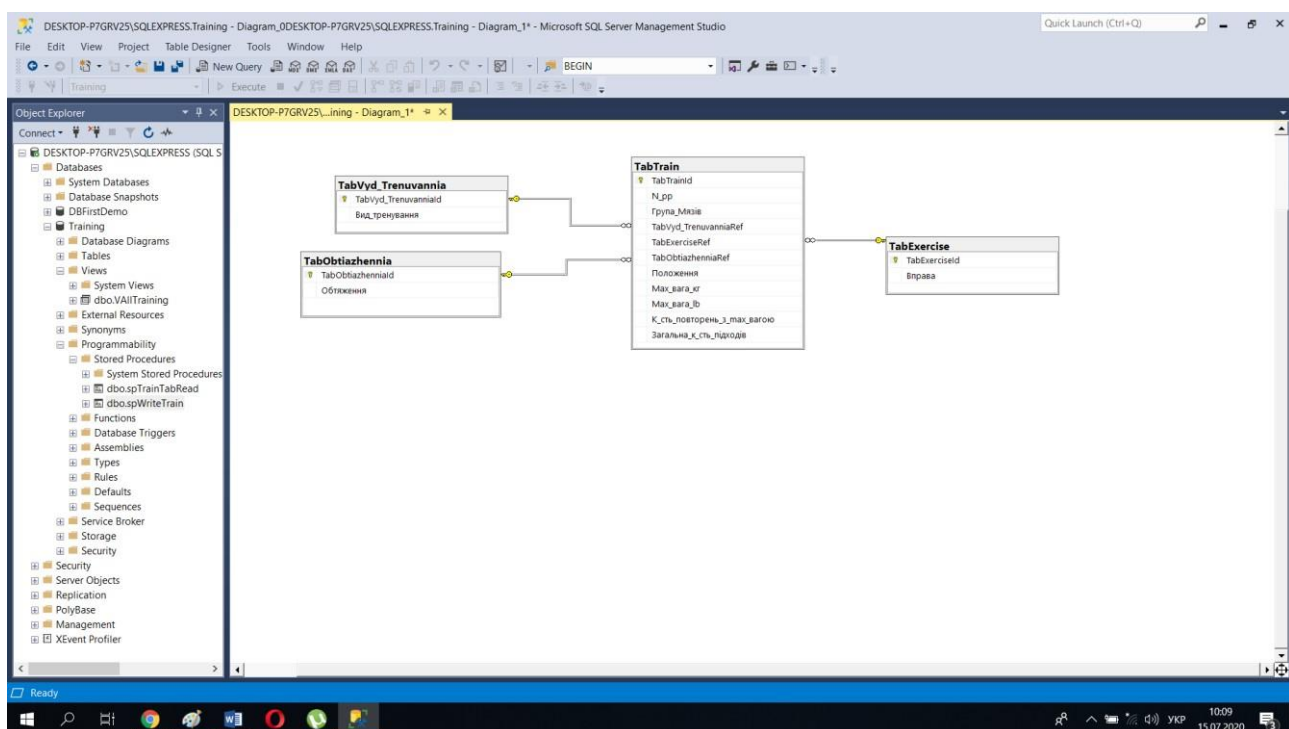
☒ **Table Designer**  
 Enforce For Replication Yes  
 Enforce Foreign Key Cons Yes  
 > INSERT And UPDATE Spec

Add Delete Close

Тепер SQL Server не дозволить нам ввести у таблицю TabTrain у поля TabExerciseRef, TabVyd\_TrenuvanniaRef або TabObtiazhenniaRef значення, відсутні у ключових полях відповідних довідників.

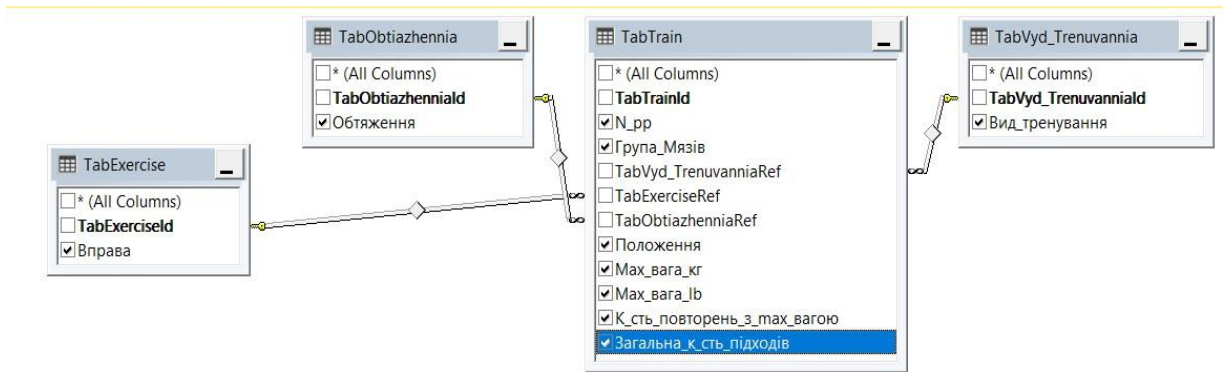
## 5) Створення діаграм

SQL Server Management Studio дозволяє візуалізувати архітектуру бази за допомогою засобу Database Diagrams. Натиснемо ПКМ на Database Diagrams → ЛКМ на New Database Diagrams → послідовно натискаємо Add, поки не додадуться всі таблиці. Якщо виникає помилка “Index was outside the bounds of the array”, то варто перезавантажити SQL Server Management Studio, така ж помилка може виникнути й при створенні View.



## 6) Створення view

Створимо представлення (View), яке б читало із бази даних інформацію у вигляді, зручному для користувача. Для цього клацнемо ПКМ на об'єкті View нашої бази даних і виберемо елемент New view із контекстного меню, що відкриється. На екрані буде форма із пропозицією включити таблиці у наше представлення. Додаємо усі таблиці. У правій панелі побачимо зображення цих таблиць та встановлених нами зв'язків між ними. Повідмічаємо галочками ті поля, які ми хочемо бачити у нашому представленні:



Натиснемо , щоб зберегти дане представлення. Введемо назву «VTrain».

## 7) Створення Stored Procedures

Зауважимо лише, що поля у цій програмі – це назви стовпців таблиць, а все решта – параметри. Імена усіх параметрів, як зовнішні, описані у заголовку процедури, так і внутрішні, описані у тілі процедури оператором declare, починаються із символу @. Для побудови правильних операторів Select можна скористатись засобами створення представлень, копіюючи створені ними оператори Select і включаючи потім їх у Storedпроцедуру. Щоб створити нову процедуру потрібно двічі ЛКМ на Stored Procedures розділу Programmability і вибрати New Stored Procedure.

Праворуч з'явиться вікно із заготовкою для нової процедури. Наберіть такий текст процедури:

```
USE [Training]
GO
/***** Object:  StoredProcedure [dbo].[spTrainTabRead]    Script Date: 15.07.2020
7:58:01 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description:  <Description,,>
-- =====
CREATE PROCEDURE [dbo].[spTrainTabRead] AS
BEGIN
SELECT TOP(100) PERCENT dbo.TabTrain.N_pp, dbo.TabTrain.Група_М'язів,
dbo.TabVyd_Trenuvannia.Вид_тренування, dbo.TabExercise.Вправа,
dbo.TabObtiazhennia.Обтяження,
dbo.TabTrain.Положення, dbo.TabTrain.Max_vaga_kg, dbo.TabTrain.Max_vaga_lb,
dbo.TabTrain.K_сть_повторень_з_max_vagoю, dbo.TabTrain.Загальна_к_сть_підходів FROM
dbo.TabObtiazhennia INNER JOIN
dbo.TabTrain ON dbo.TabObtiazhennia.TabObtiazhenniaId = dbo.TabTrain.TabObtiazhenniaRef
INNER JOIN
dbo.TabVyd_Trenuvannia on dbo.TabVyd_Trenuvannia.TabVyd_TrenuvanniaId =
dbo.TabTrain.TabVyd_TrenuvanniaRef INNER JOIN
dbo.TabExercise ON dbo.TabTrain.TabExerciseRef = dbo.TabExercise.TabExerciseId END
```



Як правило, більш поширений підхід з'єднання даних з різних таблиць представляє застосування оператора JOIN. Загальний формальний синтаксис застосування оператора INNER JOIN:

```
SELECT стовпці
FROM таблиця 1
    [INNER] JOIN таблиця 2
ON умова 1
    [[INNER] JOIN Таблиця 3
ON умова 2]
```

Після оператора JOIN йде назва другої таблиці, з якої треба додати дані до вибірки. Перед JOIN може використовуватися необов'язкове ключове слово INNER. Його наявність або відсутність ні на що не впливає. Потім після ключового слова ON вказується умова з'єднання. Ця умова встановлює, як дві таблиці будуть порівнюватися. У більшості випадків для з'єднання застосовується первинний ключ головної таблиці і зовнішній ключ залежної таблиці.

Щоб процедура записалась у базу даних і зберігалась, потрібно ЛКМ Execute у піктографічному меню. Цій процедурі надано ім'я "spTrainTabRead". Процедура не створиться, поки ви не наберете її текст без помилок. Якщо наступного разу ви захочете змінити текст створеної процедури, то перший оператор повинен починатись словами Alter PROCEDURE. Утиліта сама виправить цей рядок, якщо ви виберете елемент меню Modify, цикнувши ПКМ на потрібну процедуру.

Напишемо ще одну процедуру для запису в БД інформації, яка міститься в таблицях клієнтської програми

```
USE [Training]
GO
/***** Object:  StoredProcedure [dbo].[spWriteTrain]    Script Date: 15.07.2020
8:02:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description:  <Description,,>
-- =====
CREATE PROCEDURE [dbo].[spWriteTrain] (@N_pp as int, @groupMuscle as nvarchar(50),
@typeTraining as nvarchar(255), @exercise as nvarchar(255),
@encumbrance as nvarchar(255), @position as nvarchar(255),
@weightKg as int, @weightLb as int, @reps as int, @sets as int)
AS
Declare @typeTrainingRef as int, @exerciseRef as int, @encumbranceRef as int --
оголошення робочих змінних
BEGIN
```

```

-- Insert statements for procedure here
SELECT @typeTrainingRef = TabVyd_TrenuvanniaId from TabVyd_Trenuvannia where
Вид_тренування=@typeTraining --вибираємо усі значення поля TabVyd_TrenuvanniaId з
таблиці TabVyd_Trenuvannia,
--де Вид_тренування=@typeTraining, присвоюючи при цьому змінній
@typeTrainingRef значення поля TabVyd_TrenuvanniaId if @@ROWCOUNT = 0 --
@@ROWCOUNT - системна змінна, куди SQL поміщає к-сть рядків, які були вибрані
під час останньої операції.
--Тобто ми вставляємо в TabVyd_Trenuvannia новий рядок, якщо більше немає
ідентичних значень поля Вид_тренування, так само це робимо у наступних умовних
операторах if
begin
insert TabVyd_Trenuvannia (Вид_тренування) values (@typeTraining) --додаємо
рядок у таблицю TabVyd_Trenuvannia у поле Вид_тренування значення змінної
@typeTraining
set @typeTrainingRef = @@IDENTITY --у системну змінну SQL поміщає індекс
рядка, який був утворений у останній операції
end
select @exerciseRef=TabExerciseId from TabExercise where Вправа=@exercise -
перевіряємо, чи є у таблиці TabExercise вправа з назвою @exercise if
@@ROWCOUNT = 0 --якщо немає, тобто @@ROWCOUNT = 0
begin --то додаємо новий рядок в цю таблицю і в поле Вправа надаємо значення
@exercise
insert TabExercise(Вправа) values (@exercise)
set @exerciseRef = @@IDENTITY --засилаємо у поле @exerciseRef значення
@@IDENTITY, яка містить Id нового рядка
end
--у наступних 6-ти рядках відбуваються ідентичні дії
SELECT @encumbranceRef = TabObtiazhenniaId from TabObtiazhennia where
Обтяження=@encumbrance
if @@ROWCOUNT = 0
begin
insert TabObtiazhennia(Обтяження) values (@encumbrance)
set @encumbranceRef = @@IDENTITY
end
begin tran
insert TabTrain (N_pp, Група_Мязів, TabVyd_TrenuvanniaRef, TabExerciseRef,
TabObtiazhenniaRef, Положення, Max_вага_кг, Max_вага_lb, К_сть_повторень_з_max_вагою,
Загальна_к_сть_підходів)
values (@N_pp, @groupMuscle, @typeTrainingRef, @exerciseRef, @encumbranceRef,
@position, @weightKg, @weightLb, @reps, @sets) if @@ERROR > 0 rollback tran else
commit tran --якщо виникає помилка, тобто @@ERROR > 0, то припиняємо записування і
приводимо базу у стан, в якому вона була до початку записування END

```

Виконуємо команду Execute та зберігаємо цю Stored Procedure з назвою “spWriteTrain”.

Залишилось лише створити процедуру для очищення рядків з таблиці TabTrain з назвою spClearTrain:

```

CREATE PROCEDURE [dbo].[spClearTrain] AS
BEGIN
TRUNCATE table Tabtrain
END

```