# Упражнение: Моделиране на бази данни

Тествайте решенията си в **Judge системата**: <https://judge.softuni.org/Contests/Practice/Index/4606#0>

## База данни People

В настоящата поредица задачи ще се заемем със създаването на база данни "**People**". Тази база данни ще включва няколко **таблици**, които ще бъдат **свързани** с помощта на **foreign key** **constraints** (ограничения за външни ключове). Ще изпълним дейности като **добавяне** на **записи** и **дефиниране** на **ограничения**, които ще усъвършенстват **структурата** и цялостната **функционалност** на **базата** **данни**.

### Създайте база данни People

**Създайте база данни** симе"**People**".

### Насоки

**Създаваме** базата данни чрез следния **скрипт**:



След изпълнението на **заявката**, базата данни се **визулизира**:

A screenshot of a computer

Description automatically generated

### Създайте таблици People и Towns

В новосъздадената база данни **People** добавете таблица **People (Id, Name, Age)**. След това добавете нова таблица **Towns (Id, Name)**. Задайте **Id** колоните на двете таблици да бъдат **първичен ключ** като **ограничение**.

### Насоки

За да се **уверим**, че работим с базата данни **People**, от **падащото меню**, избираме "**People**".

A screenshot of a computer

Description automatically generated

Създаваме таблицата **People** чрез следния **скрипт**, като правим колоната **Id** да е **първичен ключ**:

A white background with black text

Description automatically generated

По същия начин процедираме с таблицата **Towns**:

A close-up of a table

Description automatically generated

**Резултатът** от **изпълнението** на **заявките** се отразява в **базата данни**:

A screenshot of a computer

Description automatically generated

### Свържете таблиците People и Towns

Променете структурата на таблицата People, за да имате **нова колона TownId**, която ще бъде от същия тип като колоната **Id** на **таблицата Towns**. Добавете **ново ограничение**, което прави **TownId външен ключ** и реферира към колоната **Id** на таблицата **Towns**.

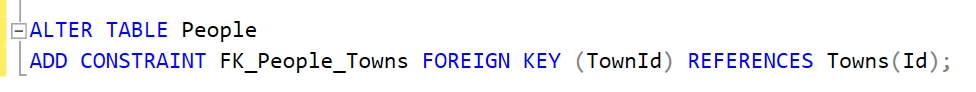
### Насоки

Добавете **нова колона** с името **TownId** от тип **INT** към таблицата **People**:

A close up of words

Description automatically generated

След това направете новодобавената колона **външен ключ**, сочещ към таблицата **Towns**:



Създадохме **връзка** между таблиците **People** и **Towns**:

A screenshot of a computer

Description automatically generated

### Вмъкнете записи в двете таблици

**Попълнете и двете таблици** с примерни записи, дадени в таблицата по-долу.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **People** | | | |  | **Towns** | |
| **Id** | **Name** | **Age** | **TownId** |  | **Id** | **Name** |
| 1 | Kevin | 22 | 1 |  | 1 | Sofia |
| 2 | Bob | 15 | 3 |  | 2 | Plovdiv |
| 3 | Steward | NULL | 2 |  | 3 | Varna |

**Използвайте само SQL заявки**. Въведете идентификатора **ръчно** (не използвайте identity).

### Насоки

Използвайте **INSERT** заявка, за да **вмъкнете** записите в **двете таблици**. Първо въведете **градове**, за да не се получи **конфликт** при въвеждане на **невалиден идентификатор** за **град** при **вмъкване** на **запис** в таблицата **People**:

A close up of text

Description automatically generated

Сега можем да **вмъкнем** записи в таблицата **People**:

A screenshot of a computer code

Description automatically generated

Виждаме, че **записите** са се **добавили** успешно:

A screenshot of a computer

Description automatically generated

### Създайте таблица Users

Използвайки **SQL заявка**, създайте таблица **Users** с колони:

* **Id** – уникален номер за всеки потребител. Няма да има **повече от 263-1 потребители.** (Автоматично нараства)
* **Username** – уникалният идентификатор на потребителя ще бъде **не повече от 30 знака** (**не Unicode**) (Задължително)
* **Password** – паролата няма да бъде **по-дълга от 26 символа** (**не Unicode**)(Задължително)
* **ProfilePicture** – изображение с размер **до 900 KB.**
* **LastLoginTime**
* **IsDeleted** – показва дали потребителят е изтрил своя профил. Възможните състояния са **true** или **false**.

Направете колоната **Id** първичен ключ. Попълнете таблицата с точно **5 записа**.

### Насоки

За **създаване** на **базата данни** е необходимо да използвате **подходящ** **формат** за **всяка** **колона**:

A screenshot of a computer screen

Description automatically generated

Сега остана да **попълните** таблицата с **5** записа по ваш избор. Спазвайте **типа** на **данни** при въвеждането им.

### Сменете първичния ключ

Използвайки **SQL заявки**, модифицирайте таблицата **Users** от предишната задача. Първо **премахнете текущия първичен ключ**, след което създайте **нов първичен ключ**, който ще бъде **комбинацията** от полета **Id** и **Username**.

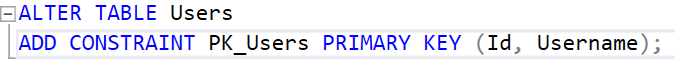
### Насоки

Първо премахваме **ограничението** за **първичния ключ**:

A close up of a sign

Description automatically generated

След това създаваме **композитен първичен ключ**:

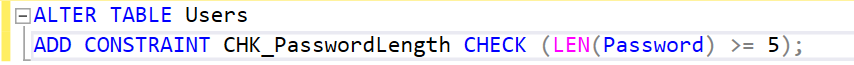


### Добавете ограничение за проверка

Използвайки **SQL заявки**, модифицирайте таблицата **Users**. Добавете **ограничение за проверка**, за да се уверите, че стойностите в полето за парола са **с дължина поне 5 символа**.

### Насоки

Трябва да добавим **ограничение**, което при **въвеждане** на **запис** в таблицата, да **провери** дали **отговаря** на дадено **условие**. За тази задача ще използваме функцията **LEN**:

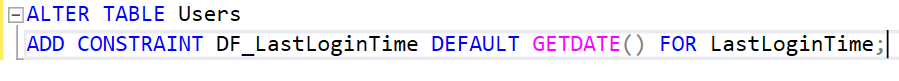


### Сложете подразбираща се стойност

Използвайки **SQL заявки**, модифицирайте таблицата **Users**. Направете **подразбиращата се стойност** (default value) за полето **LastLoginTime** да бъде **текущото време** (current data + time).

### Насоки

Трябва да добавим **ограничение**, което при **пропускане** на **въвеждане** на **данни** в колоната **LastLoginTime**, да се попълни **автоматично** зададена **стойност**. За тази задача ще използваме функцията **GETDATE**:



### Задайте уникално поле

Използвайки **SQL заявки**, модифицирайте таблицата **Users**. Премахнете полето **Username** от първичния ключ, така че само полето **Id** да бъде първичен ключ. Сега **добавете уникално ограничение** към полето **Username**, за да сте сигурни, че стойностите там са **дълги поне 3 символа**.

### Насоки

Първо **премахнете** сегашния **комбиниран първичен ключ**, след което задайте колоната **Id** да е **първичен ключ**. Накрая добавете **ограничение** за **проверка** (**CHECK CONTRAINT**), което да проверява дали **данните** за колоната **Username** са с размер **по-голям** или **равен** на **3** символа.

### Свържете таблиците People и Users

Свържете таблиците People и Users, така че всеки човек да може да има незадължителен потребител (username), с който е свързан. Дефинирайте подходящи **ограничения**, така че връзката **People – Users** да е с множественост **1 към 0 или 1**.

### Качете решението в Judge

Изберете **броя** на **всички** **редове** в таблиците **Towns**, **People** и **Users**, като използвате **3** **SELECT** **заявки**. За броя на редовете използвайте функцията **COUNT**.

## База данни Movies

С помощта на **SQL заявки** създайте база данни **Movies** със следните обекти:

* **Directors** (Id, DirectorName, Notes)
* **Genres** (Id, GenreName, Notes)
* **Categories** (Id, CategoryName, Notes)
* **Movies** (Id, Title, DirectorId, CopyrightYear, Length, GenreId, CategoryId, Rating, Notes)

Задайте **най-подходящите типове данни** за всяка колона. **Задайте първичен ключ** за всяка таблица. Попълнете всяка таблица с точно **5 записа**. Уверете се, че колоните, които присъстват в 2 таблици, са от **един и същи тип данни**. Помислете кои полета винаги са **задължителни** и кои са **незадължителни**. Свържете таблицата Movies с таблиците Directors, Genres и Categories чрез **foreign key constraint**.

### Качете решението в Judge

Изберете **броя** на **всички** **редове** в таблиците **Directors**, **Genres**, **Categories** и **Movies**, като използвате **4** **SELECT** **заявки**. За броя на редовете използвайте функцията **COUNT**.