Lab_02

March 7, 2022

1 LAB 02 - Projektowanie baz danych

1.1 Tworzenie nowej bazy danych

Na początku musimy stworzyć nową bazę danych. Służy do tego polecenie **CREATE DATABASE < nazwa bazy >**. Dla naszej wygody przed stworzeniem bazy usuniemy dotychczasową bazę występującą pod taką samą nazwą jeżeli istnieje.

[ ]:
```
USE master;
GO
```

[ ]:
```
if exists(select * from sysdatabases where name='Intro')
        drop database Intro
GO
```

[ ]:
```
CREATE DATABASE Intro;
GO
```

Pamiętajcie o tym, żeby po utworzeniu bazy się na nią przełączyć, bo inaczej wszystkie tabele stworzą się w bazie *master*.

[ ]:
```
USE Intro;
GO
```

1.2 Definiowanie tabel

Kolejnym krokiem jest zdefiniowanie tabel występujących w rozwiązaniu. Tabele powinny odpowiadać obiektom występującym w świecie rzeczywistym (np. kraj, samolot). Poszczególne cechy obiektów powinny zostać zapisane w postaci kolumn w tabeli.

Do tworzenia nowej tabeli wykorzystujemy polecenie **CREATE TABLE < nazwa tabel > (< lista kolumn >)**. Nazwa tabeli musi być unikalna w obrębie jednej bazy danych. Kolumny podajemy kolejno po przecinku podając nazwę kolumny, jej typ oraz informacje czy dopuszczamy brak wartości (NULL / NOT NULL).

Na poniższym przykładzie mamy dwie tabele **Countries** oraz **Planes**. Obie tabele mają sztuczne **klucz główne id** typu **INT**. Mimo, że często możliwe jest stworznie klucza naturalnego (np. nazwa państwa może być kluczem głównym, bo jest unikalna i niepusta) to często tworzy się klucze sztuczne w celu poprawienia wydajności. Klucz sztuczny jest najczęściej typu **INT**, który zajmuje zdecydowanie mniej miejsca niż np **VARCHAR(50)**

Relacje pomiędzy tabelami realizujemy poprzez dodanie do tabeli **kluczy obcych**. W naszym przykładzie takim kluczem obcym jest kolumna *origin_id* w tabeli Planes. Do stworzenia relacji służy słowo kluczowe **REFERENCES < nazwa tabeli obcje >(< klucz główny w tabeli obcej >)**, które dodajemy w definicji kolumny. Typy obu kolumn powinny być jednakowe.

[ ]:
```
CREATE TABLE Countries(
    id      INT     NOT NULL PRIMARY KEY,
    name    VARCHAR(30) NOT NULL,
)

CREATE TABLE Planes (
    id      INT     NOT NULL PRIMARY KEY,
    name    VARCHAR(30) NOT NULL,
    origin_id  INT NOT NULL REFERENCES Countries(id)
)
GO
```

W przypadku kluczy złożonych klucz główny należy zdefiniować oddzielnie.

[ ]:
```
CREATE TABLE test(
    name VARCHAR NOT NULL,
    surname VARCHAR NOT NULL,
    PRIMARY KEY(name, surname)
)
```

W przypadku relacji wiele-do-wielu będziemy potrzebowali dodatkowej tabeli przechowujące pary kluczy głównych z łączonych tabel.

[ ]:
```
CREATE TABLE A(
    id      INT     NOT NULL PRIMARY KEY,
    name    VARCHAR(30) NOT NULL,
)

CREATE TABLE B(
    id      INT     NOT NULL PRIMARY KEY,
    name    VARCHAR(30) NOT NULL,
)

CREATE TABLE A_B(
    A_id INT NOT NULL REFERENCES A(id),
    B_id INT NOT NULL REFERENCES B(id)
)
GO
```

1.3 Example1

1.3.1 Opis problemu

Celem zadania jest przygotowanie projektu tabel. Projekt powinien definiować tabele, kolumny (w tym typy danych, długość i wymagalność), klucze główne i klucze obce dla opisanego poniżej

problemu. Dla każdego klucza obcego należy zdefiniować tabelę i kolumnę/kolumny w tej tabeli, do której klucz obcy się odnosi. Opis problemu:

1. Należy stworzyć bazę danych dla międzynarodowej firmy produkcyjnej o cechach opisanych w poniższych punktach
2. Firma produkuje dużą liczbę produktów montowanych w różnych krajach
3. Każdy produkt jest montowany w jednym kraju, z części pochodzących potencjalnie z różnych krajów
4. Dla każdej części obowiązuje oddzielna cena i okres produkcji
5. Części, które nie są już produkowane, mają określoną datę zakończenia produkcji
6. W każdym dziale firmy jest wielu menedżerów sprzedaży.
7. Są oni przypisani do grup produktów, za które są odpowiedzialni.
8. Dla każdego produktu jest tylko jeden menedżer produktu. Każdy menedżer działa tylko w jednym dziale.
9. Istnieje unikalny numer statystyczny dla każdej części nadawany przez każdy kraj oddzielnie.
10. Nie ma ograniczeń co do wielkości opisu części. Niektóre opisy mogą być wielostronicowe.
11. Każdy produkt ma globalnie unikalny kod.

1.3.2 Rozwiązanie

Poniższe rozwiązanie przygotował: **Wilczyński Piotr**

Kilka uwag od prowadzącego

1. Można rozważyć wyodrębnienie działu pracownika do osobnej tabeli.

2. W zależności od interpretacji encji *Parts* możliwe jest także stworzenie relacji wiele-do-wielu pomiędzy *Parts* oraz *Products*

[ ]:
```
USE master
GO

if exists(select * from sysdatabases where name='Example1')
        drop database Example1
GO

CREATE DATABASE Example1
GO

USE Example1
GO
```

[ ]:
```
CREATE TABLE Countries(
    id          INT             NOT NULL PRIMARY KEY,
    name        VARCHAR(20)     NOT NULL
)

CREATE TABLE Managers(
    id          INT             NOT NULL PRIMARY KEY,
    departament VARCHAR(50)     NOT NULL
```

```
)

CREATE TABLE Product_gropus(
    id          INT             NOT NULL PRIMARY KEY,
    name        VARCHAR(5)      NOT NULL,
    manager_id  INT             NOT NULL REFERENCES Managers(id)
)

CREATE TABLE Products(
    product_code INT            NOT NULL PRIMARY KEY,
    installation_country_id INT NOT NULL REFERENCES Countries(id),
    products_group INT          NOT NULL REFERENCES Product_gropus(id)
)

CREATE TABLE Parts(
    statistical_code INT        NOT NULL,
    origin_country_id INT       NOT NULL REFERENCES Countries(id),
    product_code INT            NOT NULL REFERENCES Products(product_code),
    price       FLOAT           NOT NULL,
    start_of_production DATETIME NOT NULL,
    end_of_production   DATETIME,
    description  TEXT           NOT NULL,
    PRIMARY KEY(statistical_code, origin_country_id)
)
```

1.4 Dodatek

Większość informacji dostępnych przez GUI SQL Managment Studio jest dostępna także z poziomu poleceń SQL.

[ ]:
```
USE Northwind;
```

1.4.1 Lista kolumn z tabeli

[ ]:
```
EXEC sp_columns Orders;
```

1.4.2 Dodatkowe informacje o tabeli

[ ]:
```
EXEC sp_help Orders;
```

1.4.3 Lista tabel

[ ]:
```
EXEC sp_tables @table_type = "'TABLE'";
```