Fundusze Europejskie
Wiedza Edukacja Rozwój

Unia Europejska
Europejski Fundusz Społeczny

Programowanie zespołowe 2.0

Modelowanie i tworzenie baz danych

Scenariusz zajęć

Czas realizacji: 180 minut
Cele ogólne: Modelowanie zbiorów danych w postaci relacyjnych baz danych; Tworzenie baz danych w systemie PostgreSQL.
Cel szczegółowy: Przygotowanie modelu bazy danych dla przygotowywanego projektu.
Konieczne wiadomości wstępne: —
Metoda prowadzenia zajęć: Dyskusja z elementami wykładu.

Podstawowe pojęcia

1. atrybut – cecha obserwacji przechowywanych w bazie;
2. krotka – zestaw atrybutów opisujących obserwację (rekord w tabeli);
3. relacja – zbiór krotek, tabela;
4. encja – połączenie między relacjami;
5. relacyjna baza danych – zbiór danych, w którym dane zgrupowane są w relacje;
6. klucz główny – atrybut lub zestaw atrybutów pozwalających zidentyfikować krotkę;
7. klucz obcy – atrybut lub zestaw atrybutów realizujących połączenie między tabelami;

Przykład relacyjnej bazy danych

W dobie informatyzacji szkół, bardzo popularnym zbiorem danych jest baza uczniów szkoły. W najprostszej wersji, zawiera ona informacje o poszczególnych uczniach i klasach, do których oni uczęszczają. Fragment takiej bazy w postaci tabelarycznej wygląda następująco:

Uczniowie

Id	Imię	Nazwisko	**Klasa**	...
1	Jan	Kowalski	**2**	...
2	Jacek	Nowak	**1**	...
3	Paulina	Wiśniewska	**3**	...
4	Maja	Kubik	**2**	...
⋮	⋮	⋮	⋮	⋮

Klasy

Id	Nazwa	Wychowawca	...
1	III a	mgr Jeży Klimek	...
2	III b	dr Karol Wilk	...
3	III c	mgr Józef Żuk	...
⋮	⋮	⋮	⋮

Uwaga. W trosce o jednoznaczność powinniśmy zadbać, by w każdej komórce tabeli znajdowała się niepodzielna informacja. Dlatego atrybut *Wychowawca* w tabeli *Klasy* może występować w tej postaci tylko przy założeniu, że nie będziemy pobierać oddzielnie ani imienia, ani nazwiska, ani stopnia naukowego wychowawcy.

Atrybut *Klasa* w tabeli *Uczniowie* jest tutaj kluczem obcym odnoszącym się do tabeli *Klasy*. Aby znaleźć szczegółowe informacje o klasie ucznia, musimy wyszukać je w tabeli *Klasy* wykorzystując odpowiednie Id klasy.

Modelując podział zbioru danych na relacje powinniśmy pamiętać o dwóch zasadach:

1. Dane nie powinny się powtarzać (dzięki podziałowi na dwie relacje, nie musimy powtarzać przy każdym uczniu wszystkich danych o jego klasie),

2. Usuwanie obiektów różnych typów nie powinno na siebie wpływać (usunięcie z bazy ucznia nie powinno skutkować usunięciem informacji o klasie).

Diagram encji

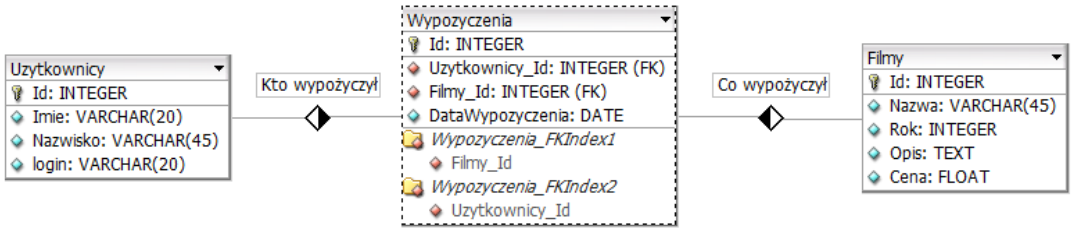
Zawartość zbioru danych nie ma znaczenia w trakcie modelowania. Istotne są tylko zależności między atrybutami (tzw. *zależności funkcyjne*). Przykładami takich zależności w bazie uczniów są np.: Id ucznia determinuje klasę, do której uczęszcza; nazwa klasy determinuje wychowawcę. Znajomość tych zależności pozwala zaprojektować bazę zgodnie z zasadami 1. i 2. bez znajomości danych w bazie.

Do wizualizacji struktury bazy (bez danych) służą diagramy encji. Tabele zapisujemy jako listy atrybutów, a klucze obce ilustrujemy za pomocą strzałek. Diagram encji dla przedstawionego wycinka bazy uczniów wygląda jak na grafice 1. Został on wykonany za pomocą aplikacji do modelowania baz danych DBDesigner 4.

Grafika 1: Diagram encji dla bazy uczniów szkoły

Przykład 1. Internetowa wypożyczalnia filmów udostępnia swoim użytkownikom filmy (do których ma prawa) do obejrzenia w ciągu 24 godzin od chwili wykupienia dostępu. Do sprawnego działania serwisu potrzebna jest baza danych przechowująca dane o dostępnych filmach, użytkownikach serwisu oraz dokonanych transakcjach.

Struktura danych wyraźnie wskazuje na podział na 3 relacje: filmy, użytkownicy i wypożyczenia. Załóżmy bardzo uproszczony model, w którym dla każdego filmu znamy tylko jego tytuł, rok wydania, opis szczegółów (zawierający opis fabuły, reżysera, obsadę, itp.) i cenę wypożyczenia, natomiast dla użytkownika znamy jego imię, nazwisko i login systemowy. Diagram encji dla takiej bazy danych ilustruje grafika 2.



Grafika 2: Diagram encji dla bazy internetowej wypożyczalni wideo

Ćwiczenie 1. Zaprojektuj diagram encji dla bazy danych studentów. Baza ma przechowywać informacje o studentach, zajęciach, na które ci studenci uczęszczają oraz uzyskanych ocenach. Pamiętaj, że baza powinna spełniać założenia 1. i 2.

Ćwiczenie 2. Zaprojektuj (na papierze lub za pomocą programu DBDesigner 4) diagram encji dla bazy danych wykorzystywanej przez projektowaną przez ciebie aplikację.

Serwer bazodanowy PostgreSQL

Relacyjne bazy danych można przechowywać na różne sposoby. Wśród użytkowników domowych najpopularniejsza jest aplikacja MS Access i jej podobne, które oferują przyjazny użytkownikowi interfejs graficzny. Dużym ograniczeniem takich baz jest ograniczona możliwość korzystania z nich zdalnie. Dlatego profesjonaliści korzystają z serwerów bazodanowych wykorzystujących stosunkowo intuicyjny język zapytań SQL (ang. *Structured Query Language*). Przykładami takich serwerów są wolnodostępne MySQL i PostgreSQL oraz komercyjne Oracle Database czy MS SQL Server.

Ze względu na wolny dostęp i duże możliwości programistyczne będziemy korzystać z serwera PostgreSQL (oficjalna strona projektu *https://www.postgresql.org/*) w wersji 11.3. System zarządzania bazą można za darmo pobrać ze strony projektu i zainstalować na swoim komputerze. Zarządzać bazą możemy z poziomu konsoli lub za pomocą aplikacji pgAdmin 4.

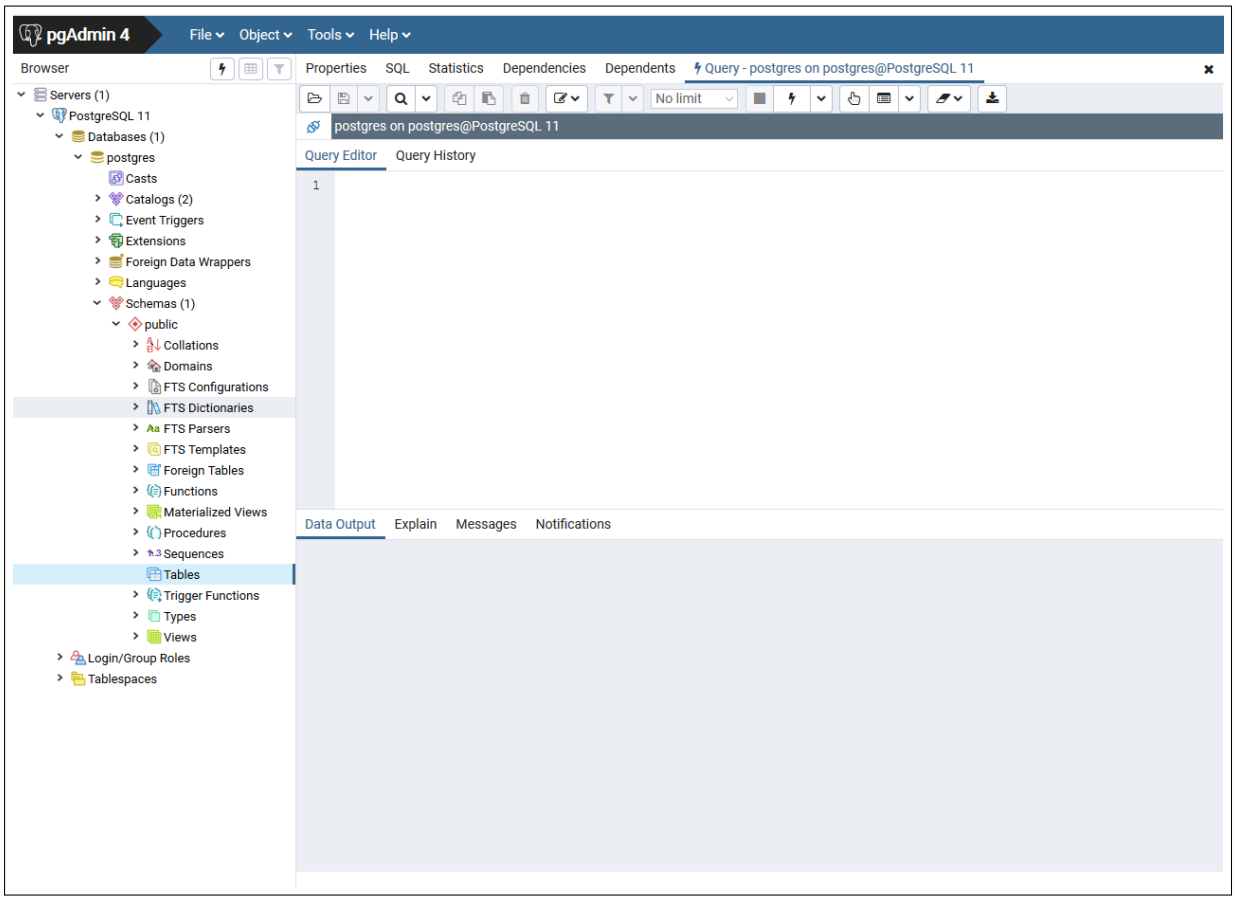
Serwer bazodanowy PostgreSQL jest również dostępny zdalnie na platformie Heroku (więcej na *https://www.heroku.com/*).

Tworzenie tabel w języku SQL

Po zainstalowaniu, uruchomieniu i zalogowaniu się na serwer możemy wykonywać zapytania SQL. Grafika 3 obrazuje interfejs użytkownika w aplikacji pgAdmin 4. Widać na niej m.in. domyślnie utworzoną bazę „postgres” oraz okno „Query Editor”, w którym będziemy pisać zapytania SQL do naszego serwera bazodanowego.

Zapytanie tworzące tabelę ma następującą strukturę:

```
CREATE TABLE NazwaTabeli(
NazwaAtrybutu1 Typ Ograniczenia,
NazwaAtrybutu2 Typ Ograniczenia,
...
);
```



Grafika 3: Interfejs użytkownika w aplikacji pgAdmin 4 pozwalający na wykonywanie zapytań SQL

Uwaga. W języku SQL wielkość liter w składni nie ma znaczenia. Zwyczajowo słowa kluczowe pisze się wielkimi literami, gdyż poprawia to czytelność kodu. Każde zapytanie kończy się średnikiem.

Przykład 2. Zapytania SQL tworzące tabele dla diagramu encji z przykładu 1. wyglądają następująco:

```
CREATE TABLE Uzytkownicy(
IdUzytkownika SERIAL PRIMARY KEY,
Imie VARCHAR(20),
Nazwisko VARCHAR(45),
Login VARCHAR(20) NOT NULL
);
```

Typ `VARCHAR(n)` oznacza typ znakowy o ustalonej długości, natomiast `SERIAL` mówi serwerowi, że wartość tego atrybutu ma być nadawana automatycznie z ciągu arytmetycznego. Flaga `NOT NULL` wymaga od użytkownika podania wartości tego atrybutu, a `PRIMARY KEY` oznacza klucz główny – atrybut identyfikujący krotkę (nie może być pusty i musi być unikatowy).

```
CREATE TABLE Filmy(
IdFilmu SERIAL PRIMARY KEY,
Tytul VARCHAR(100) NOT NULL,
Rok INTEGER CHECK(Rok > 1895),
Opis TEXT,
Cena NUMERIC CHECK(Cena >= 0)
);
```

```
CREATE TABLE Wypozyczenia(
Id SERIAL PRIMARY KEY,
IdUzytkownika INTEGER NOT NULL REFERENCES Uzytkownicy(IdUzytkownika),
IdFilmu INTEGER NOT NULL REFERENCES Filmy,
DataWypozyczenia DATE
);
```

Typy `INTEGER` i `DATE` są typami odpowiednio całkowitoliczbowym i datowym. Funkcja `CHECK` wymusza wprowadzanie danych spełniających określony warunek logiczny. Z kolei słowem kluczowym `REFERENCES` tworzymy klucz obcy – powiązanie z inną tabelą. W przypadku niepodania atrybutu, do którego się odwołujemy, serwer będzie się odnosił do klucza głównego w podanej tabeli. Klucz obcy może przyjmować tylko wartości, które już istnieją w tabeli, do której się odwołuje. Stąd też musi on mieć ten sam typ, co atrybut docelowy.

Uwaga. Jeżeli nie użyjemy flagi `NOT NULL`, to klucz obcy, w przeciwieństwie do głównego, może pozostać pusty.

Więcej o typach i ograniczeniach w tabelach dopuszczanych w bazie PostgreSQL można przeczytać w dokumentacji (*https://www.postgresql.org/docs/*).

Ćwiczenie 3. Napisz zapytania tworzące tabele dla diagramu encji z ćwiczenia 1. Pamiętaj o ograniczeniach w tabelach.

Ćwiczenie 4. Napisz zapytania tworzące tabele dla diagramu encji z ćwiczenia 2. Przemyśl odpowiednio ograniczenia w tabelach. Czy mogą być one przeszkodą we wprowadzaniu danych?

Zarządzanie danymi w bazie

Do zarządzania danymi w tabelach służą zapytania typu `INSERT`, `UPDATE` oraz `DELETE`.

Zapytanie wprowadzające dane do bazy ma następującą strukturę:

```
INSERT INTO NazwaTabeli(NazwyAtrybutów)
VALUES
(WartościAtrybutów1),
(WartościAtrybutów2), ..., (WartościAtrybutówN);
```

Nazwy atrybutów możemy podawać w dowolnej kolejności, jednak potem wartości dla kolejnych krotek podajemy zgodnie z kolejnością nazw. Nie musimy podawać wartości wszystkich atrybutów, pominiętym atrybutom zostaną przypisane wartości domyślne. Podawanie nazw atrybutów nie jest konieczne. W przypadku ich pominięcia, należy podawać wartości wszystkich atrybutów w kolejności ich deklarowania przy tworzeniu tabeli.

Przykład 3. Dodamy po kilka rekordów do utworzonych wcześniej tabel:

```
INSERT INTO Uzytkownicy(Imie, Nazwisko, Login)
VALUES
('Karol', 'Wiśniewski', 'kawi'),
('Marek', 'Kowalski', 'mako'),
('Jan', 'Szulc', 'jasz');

INSERT INTO Filmy
VALUES
(1, 'Django', 2012, 'Kolejny film QT', 5),
(2, 'O północy w Paryżu', 2011, 'Scenariusz: Woody Allen', 2);

INSERT INTO Wypozyczenia VALUES (2, 1, 1, CURRENT_DATE);

INSERT INTO Wypozyczenia(IdUzytkownika, IdFilmu, DataWypozyczenia)
VALUES
(1, 2, '2019-06-17');
```

Zauważmy, że w tabeli `Wypozyczenia` atrybut `Id` nadaliśmy „ręcznie" mimo że domyślnie przypisywany jest automatycznie (podobnie w tabeli `Filmy`).

Ćwiczenie 5. Spróbuj dwukrotnie wykonać następujące zapytanie:

```
INSERT INTO Wypozyczenia(IdUzytkownika, IdFilmu, DataWypozyczenia)
VALUES
(2, 2, '2019-06-15');
```

Dlaczego za pierwszym razem serwer zgłosił błąd, a za drugim dodał odpowiedni wpis? Jakie przypisał mu `Id`?

Po kliknięciu PPM na nazwę tabeli w drzewie w lewym panelu i wybraniu „Przeglądaj/Edytuj dane > Wszystkie wiersze" w panelu podglądu pojawi się tabela wraz z zawartością (zob. grafikę 4). Zapytania wyświetlające dane z tabel omówimy na następnych zajęciach.

Modyfikacja i usuwanie danych

Modyfikacji danych dokonujemy poprzez aktualizację tabeli wg następującego schematu

	iduzytkownika integer	imie character varying (20)	nazwisko character varying (45)	login character varying (65)
1	1	Karol	Wiśniewski	kawi
2	2	Marek	Kowalski	mako
3	3	Jan	Szulc	jasz

	idfilmu integer	tytul character varying (100)	rok integer	opis text	cena numeric
1	1	Django	2012	Kolej...	5
2	2	O północy w Paryżu	2011	Scen...	2

	id integer	iduzytkownika integer	idfilmu integer	datawypozyczenia date
1	2	1	1	2019-07-02
2	1	1	2	2019-06-17
3	3	2	2	2019-06-15

Grafika 4: Dane wprowadzone do tabel

```
UPDATE NazwaTabeli
SET
Atrybut1 = Wartość1,
Atrybut2 = Wartość2,
...
AtrybutN = WartośćN
WHERE WarunekLogiczny;
```

Na przykład, zapytanie:

```
UPDATE Wypozyczenia
SET
IdUzytkownika = IdUzytkownika+1,
DataWypozyczenia = NULL
WHERE IdUzytkownika = 1;
```

zwiększa o 1 atrybut `IdUzytkownika` i usuwa datę wypożyczenia we wszystkich wypożyczeniach użytkownika nr 1.

Do wyczyszczenia wszystkich danych z tabeli używamy zapytania postaci

```
TRUNCATE NazwaTabeli;
```

Podobne działanie ma

```
DELETE FROM NazwaTabeli;
```

Ostatnie zapytanie możemy jednak zmodyfikować w taki sposób, by usuwało tylko krotki spełniające pewien warunek logiczny:

```
DELETE FROM NazwaTabeli
WHERE WarunekLogiczny;
```

Na przykład, zapytanie

```
DELETE FROM Wypozyczenia
WHERE DataWypozyczenia <= CURRENT_DATE - 90;
```

usuwa wszystkie wypożyczenia starsze niż 90 dni.

Więcej o konstrukcji warunków logicznych powiemy przy omawianiu zapytać wyświetlających dane.

Ćwiczenie 6. a) Uzupełnij przykładowymi danymi tabele utworzone w ćwiczeniu 3.

b) Podnieś o 1 oceny wszystkich studentów o parzystych identyfikatorach.

c) Usuń wszystkie oceny negatywne.

Odnośniki i załączniki

[1] DBDesigner 4: https://dbdesigner.softonic.pl/

[2] Strona główna projektu PostgreSQL: https://www.postgresql.org/

[3] Dokumentacja PostgreSQL: https://www.postgresql.org/docs/

[4] Zdalny serwer Heroku: https://www.heroku.com/

[5] Pliki `diagram_klasa.xml` i `diagram_vod.xml` z diagramami encji wykonane w DBDesigner 4.