Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

Zaawansowane Techniki Bazodanowe

Krzysztof Koptyra, Dawid Tomaszewski

Projekt logiczny

Prowadzący:
dr inż. Sebastian Ernst

Kraków 2011

Spis treści

1. Projekt bazy w języku SQL.

Serwis kucharski korzysta z bazy danych PostgreSQL. W celu prostszego jej utworzenia na środowisku produkcyjnym przygotowane zostały skrypty SQL tworzące bazę danych, dodające tabele, klucze główne, obce i połączenia między nimi oraz wprowadzające dane do bazy.

Ponieważ kompletne skrypty zajmują ponad 3 MB (ok. 30 000 linii skryptu SQL-owego) wklejenie całości do dokumentacji byłoby nieco kłopotliwe. Pokazane zostaną jedynie fragmenty kodu SQL, aby pokazać ogólną strukturę skryptów.

Fragment skryptu tworzącego bazę danych:

```
CREATE TABLE users (
  user_name VARCHAR(64) PRIMARY KEY NOT NULL,
  user_pass VARCHAR(64) NOT NULL
);
CREATE TABLE user_roles (
  user_name VARCHAR(64) NOT NULL REFERENCES
users(user_name),
  role_name VARCHAR(16) NOT NULL,
PRIMARY KEY (user_name, role_name)
);
CREATE TABLE Moderator (
  ModID SERIAL PRIMARY KEY NOT NULL,
  Imie VARCHAR(64),
  Nazwisko VARCHAR(64),
  user_name VARCHAR(64) REFERENCES users(user_name)
);
CREATE TABLE ServiceUser (
  UserID SERIAL PRIMARY KEY NOT NULL,
  user_name VARCHAR(64) REFERENCES users(user_name),
  Imie VARCHAR(64),
  Nazwisko VARCHAR(64),
  Plec CHAR(1),
  DataUrodzenia timestamp,
  Opis VARCHAR(512),
  Ulubione integer array[10],
  OstatnioObejrzane integer array[10],
  Status CHAR(1),
  DataRejestracji timestamp,
  DataBan timestamp,
```

```
  OcenyCache DECIMAL(6,2)
);
```

Fragment skryptu dodającego do bazy użytkowników:

```
INSERT INTO users (user_name ,user_pass)
  VALUES('Krystian.Nowak@gmail.com','user');
INSERT INTO ServiceUser(UserID,user_name,Imie,Nazwisko,Plec,
  DataUrodzenia,Opis,Ulubione,OstatnioObejrzane,Status,
  DataRejestracji,DataBan,OcenyCache)VALUES('1',
  'Krystian.Nowak@gmail.com','Krystian','Nowak','m','1969-01-24'
  ,NULL,NULL,NULL,'r',NULL,NULL,NULL);
INSERT INTO user_roles (user_name,role_name)
  VALUES('Krystian.Nowak@gmail.com','user');
INSERT INTO users (user_name ,user_pass)
  VALUES('Kajetan.Kowalski@gmail.com','user');
INSERT INTO ServiceUser(UserID,user_name,Imie,Nazwisko,Plec,
  DataUrodzenia,Opis,Ulubione,OstatnioObejrzane,Status,
  DataRejestracji,DataBan,OcenyCache)VALUES('2',
  'Kajetan.Kowalski@gmail.com','Kajetan','Kowalski','m',
  '1982-05-24',NULL,NULL,NULL,'r',NULL,NULL,NULL);
INSERT INTO user_roles (user_name,role_name)
  VALUES('Kajetan.Kowalski@gmail.com','user');
```

Fragment skryptu dodającego przepisy do bazy danych:

```
INSERT INTO Category(KatName) VALUES('Ciasta');
INSERT INTO Category(KatName) VALUES('Wieprzowina');
INSERT INTO Category(KatName) VALUES('Wołowina');
INSERT INTO Category(KatName) VALUES('Zupy');
INSERT INTO Recipe(PrzepisID,UserID,Ocena,PoziomTrudnosci,Opis,
  OpisPrzygotowania,Nazwa,ShortDesc,Status,IloscOcen,
  CzasPrzygotowania,OpisSkladnikow) VALUES(1,56,1,5,'Masło
ucierać, dodając po 1 żółtku i po trochu cukru, wymieszać
z kawą i proszkiem spulchniającym. Ubić białka na pianę,
wymieszać z masą żółtkową i mąką. Biszkopt wyłożyć do długiej,
wąskiej, dobrze wysmarowanej masłem formy keksowej i upiec w
piekarniku niezbyt gorącym (ok. 40 min.).
  ','Masło ucierać, dodając po 1 żółtku i po trochu cukru,
wymieszać z kawą i proszkiem spulchniającym. Ubić białka na
pianę, wymieszać z masą żółtkową i mąką. Biszkopt wyłożyć do
długiej, wąskiej, dobrze wysmarowanej masłem formy keksowej i
upiec w piekarniku niezbyt gorącym (ok. 40 min.).
  ','Biszkopt chiński',null,null,89,null,'20 dag masła
  4 żółtka
  12 dag cukru pudru
  4 białka (piana)
  13 dag mąki pszennej
  1 dag masła do smarowania formy
  12 dag mąki ziemniaczanej
  1/2 proszku spulchniającego
  2 kopiaste łyżki mielonej kawy palonej
```

```
   ');
   INSERT INTO RecipeCategory(PrzepisID,KatName)
VALUES(1,'Ciasta');
```

2. Słownik danych.

Tabela "users" – zawiera dane logowania użytkowników

user_name – klucz główny, długość maksymalnie 64 znaki, dozwolone polskie znaki, litery i cyfry, musi być w formie adresu e–mail. Typ : String.

user_pass – hasło, dozwolone wszystkie znaki alfanumeryczne i znaki specjalne. Typ : String.

Tabela "user_roles" – zawiera role (uprawnienia) poszczególnych użytkowników

user_name – część złożonego klucza głównego, klucz obcy z tabeli "users". Typ : String

role_name – część złożonego klucza głównego, dozwolone wartości : "user", "mod". Typ : String.

Tabela "Moderator" – zawiera dodatkowe dane o moderatorach

ModID – klucz główny. Typ : Integer

user_name – klucz obcy do tabeli "users". Typ : String

Imie – imię moderatora, długość maksymalnie 64 znaki, musi się zaczynać dużą literą, dozwolone tylko litery włącznie z polskimi znakami.

Nazwisko – nazwisko moderatora, długość maksymalnie 64 znaki, musi się zaczynać dużą literą, dozwolone tylko litery włącznie z polskimi znakami oraz znak "–".

Tabela ServiceUser – zawiera niezbędne dane o użytkownikach serwisu.

UserID – klucz główny. Typ : Integer

user_name – klucz obcy do tabeli "users". Typ : String

Imie – imię użytkownika, długość maksymalnie 64 znaki, musi się zaczynać dużą literą, dozwolone tylko litery włącznie z polskimi znakami.

Nazwisko – nazwisko użytkownika, długość maksymalnie 64 znaki, musi się zaczynać dużą literą, dozwolone tylko litery włącznie z polskimi znakami oraz znak "–".

Plec – płeć, pojedynczy znak "k" lub "m". Typ : Character

DataUrodzenia – data urodzenia. Typ : Date, format DD–MM–RRRR

Opis – opis użytkownika, maksymalna długość 512 znaków, dozwolone wszystkie znaki aflanumeryczne. Typ : String

Tabela Ulubione – tabela zawierająca ID przepisów oznaczonych jako ulubione. Zarezerwowane do przyszłego użycia

Tabela OstatnioObejrzane – tabela zawierająca ID przepisów ostatnio obejrzanych. Zarezerwowane do przyszłego użycia

Status – status użytkownika, pojedynczy znak "r" – zarejestrowany, "c" – potwierdzona rejestracja lub "b" – zbanowany

DataRejestracji – data rejestracji. Typ : Date, format DD–MM–RRRR

DataBan – data zniesienia bana. Typ : Date, format DD–MM–RRRR

OcenyCache – srednia ocen za wszystkie przepisy. Typ : Double

Tabela ExtendedUserData – zawiera rozszerzone dane uzytkownika, zarezerwowana na dodatkowe funkcjonalności.

UserID – klucz obcy do tabeli ServiceUser. Typ : Integer.

Tabela Recipe – zawiera podstawowe dane o każdym przepisie

PrzepisID – klucz główny. Typ : Integer

UserID – odniesienie do użytkownika który dodał przepis,klucz obcy do tabeli ServiceUser. Typ : Integer.

Ocena – średnia ocen, zakres 0–5. Typ Double.

PoziomTrudnosci – poziom trudności zakres 1–5. Typ Integer.

Opis – opis przepisu, tekst nielimitowanej długości, dozwolone wszystkie znaki alfanumeryczne. Typ : String

OpisPrzygotowania – opis przygotowania przepisu, tekst nielimitowanej długości, dozwolone wszystkie znaki alfanumeryczne. Typ : String

Nazwa – nazwa przepisu, maksymalna długość 128 znaków, dozwolone tylko litery. Typ : String

ShortDesc varchar(256) – krótki opis przepisu, zarezerwowany do użycia w przyszłości. Typ String.

Status – status przepisu, pojedynczy znak "w" – oczekujące na akceptacje, "a" – zaakceptowany lub "d" – usunięty

IloscOcen – ilość wystawionych ocen. Typ : Integer.

CzasPrzygotowania czas przygotowania w podany w minutach. Typ : Integer.

OpisSkladnikow – lista składników w formie plaintext, w celu zachowania kompatybilności z początkowymi danymi w bazie, nielimitowana ilość znaków, dozwolone wszystkie alfanumeryczne. Typ : String

Tabela Comment – zawiera komentarze do przepisów

CommentID – klucz główny. Typ : Integer

Tresc – treść komentarza, nielimitowana długość, dozwolone wszystkie znaki alfanumeryczne. Typ : String.

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer.

UserID – klucz obcy do tabeli ServiceUser. Typ : Integer.

Status – pojedynczy znak "v" – widoczny, "r" – zgłoszony lub "d" – usunięty

Tabela Photo – zawiera zdjęcia przyporządkowane do przepisów.

PhotoID – klucz główny. Typ : Integer.

Zdjecie – ścieżka do pliku ze zdjęciem. Typ : String

Opis – krótki opis zdjęcia, maksymalnie 256 znaków, dozwolone wszystkie alfanumeryczne. Typ : String

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer.

Tabela RecipeExtendedData – rozszerzone informacje o przepisie, zarezerwowane do użycia w przyszłości

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer.

Tabela Category – zawiera nazwy kategorii.

KatName –klucz główny, nazwa kategorii, tekst maksymalnie 128 znaków, dozwolone tylko litery. Typ : String

Tabela Region – zawiera nazwy regionów

RegName – klucz główny, nazwa regionu, tekst maksymalnie 128 znaków, dozwolone tylko litery. Typ : String

Tabela Tag – zawiera wszystkie tagi wraz z ich popularnością

TagName – klucz główny, nazwa tagu, maksymalna długość 128 znaków. Typ : String

Popularnosc – liczba wystąpień tagu w wynikach wyszukiwania. Typ Integer.

Tabela IngredientCat – zawiera kategorie składników

KatName – klucz główny, nazwa kategorii, maksymalnie 128 znaków, dozwolone tylko litery. Typ : String.

Tabela Ingredient – zawiera składniki do przepisów.

SkladnikID – klucz główny. Typ : Integer

KatName – klucz obcy do tabeli IngredientCat. Typ : String

Nazwa – nazwa składnika, maks 128 znaków, dozwolone tylko litery. Typ : String

Opis – opis składnika, nielimitowa ilość znaków, dozwolone wszystkie znaki alfanumeryczne. Typ : String

Zdjecie – ścieżka do pliku ze zdjęciem. Typ : String

Jednostka – jednostka w jakiej podawana jest ilość danego składnika, tekst maks. 16 znaków. Typ : String

Status – pojedynczy znak "a" – składnik dostępny lub "d" – składnik usunięty

Tabela AGDCat – zawiera kategorie sprzętu AGD

KatName – klucz główny, nazwa kategorii, maksymalnie 128 znaków, dozwolone tylko litery. Typ : String

Tabela AGD

SprzetID – klucz główny. Typ : Integer

KatName – klucz obcy do tabeli AGDCat. Typ : String

Nazwa – nazwa sprzętu, maks 128 znaków, dozwolone tylko litery. Typ : String

Opis – opis sprzętu, nielimitowa ilość znaków, dozwolone wszystkie znaki alfanumeryczne. Typ : String

Zdjecie – ścieżka do pliku ze zdjęciem. Typ : String

Jednostka – jednostka w jakiej podawana jest ilość danego sprzętu, tekst maks. 16 znaków. Typ : String

Status – pojedynczy znak "a" – sprzęt dostępny lub "d" – sprzęt usunięty

Tabela UsersAGD – tabela zarezerwowana do przyszłego użytku. Docelowo będzie odpowiadać za zapisanie na stałe informacji o posiadanym sprzęcie przez każdego użytkownika.

UserID – klucz obcy do tabeli ServiceUser. Typ : Integer

SprzetID – klucz obcy do tabeli AGD. Typ : Integer

ilosc – ilosc posiadanego sprzętu. Typ : Integer

Tabela RecipeAGD – łącznik między tabelą Recipe i AGD

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer
SprzetID – klucz obcy do tabeli Sprzet. Typ : Integer
ilosc – wymagana ilość sprzętu. Typ : Integer

Tabela RecipeIngredient – łącznik między tabelą Recipe i Ingredient

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer
SkladnikID – klucz obcy do tabeli Ingredient . Typ : Integer
ilosc integer – wymagana ilość składnika. Typ : Integer
sposob – sposób przyrządzenia składnika w danym przepisie. Typ : String

Tabela RecipeRegion – łącznik między tabelą Recipe i Region

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer
RegName – klucz obcy do tabeli Region. Typ : String

Tabela RecipeTag – łącznik między tabelą Recipe i Tag

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer
TagName – klucz obcy do tabeli Tag. Typ : String

Tabela RecipeCategory – łącznik między tabelą Recipe i Category

PrzepisID – klucz obcy do tabeli Recipe. Typ : Integer
KatName – klucz obcy do tabeli Category. Typ : String

3. Analiza zależności funkcyjnych i normalizacja tabel.

Pierwsza postać normalna.

1NF – baza danych spełnia 1NF ponieważ wszystkie pola są atomowe. Teoretycznie istnieje możliwość rozbicia atrybutów "DataUrodzenia", "DataBan" itp. na osobne pola rok, miesiac i dzień, jednak nie ma to żadnego praktycznego zastosowania, więc należy uznać te atrybuty za atomowe.

Druga postać normalna.

2NF – baza danych spełnia 2NF. Od samego początku baza była projektowana, tak aby wszystkie atrybuty danej relacji były w pełni zależne od klucza głównego.

Trzecia postać normalna.

3NF – baza danych nie spełnia 3NF między innymi z powodu wprowadzenia celowej nadmiarowości danych w niektórych tabelach. Np. w relacjach "Moderator" i "ServiceUser", atrybut "user_name" w pełni wystarczałby jako klucz główny. Wprowadzenie atrybutów "UserID" i "ModID" eliminuje potrzebę aktualizacji wszystkich powiązanych tabel w przypadku kiedy użytkownik zmieni swój login.

4. Denormalizacja struktury tabel.

Bazę danych normalizujemy głównie w celu zaoszczędzenia miejsca zajętego przy przechowywaniu danych w bazie, dzięki rozbiciu większych tabel na mniejsze, nie zawierające nadmiarowych danych. Ponadto w nieznormalizowanych bazach istnieje ryzyko wystąpienia anomalii (nadmiarowe dane w tabeli powodują wiele problemów przy usuwaniu lub edycji pewnej części z nich, np. adresu ze zbyt dużej tabeli zawierającej jednocześnie dane firm, ich adresów, zamówień, itp.). Znormalizowanie bazy danych eliminuje anomalie.

Denormalizacja jest odwrotnym, celowym procesem, polegającym na wprowadzeniu nadmiarowości do bazy danych. Poświęca ona zajęte miejsce w zamian za uzyskanie większej wydajności, gdyż przy pobieraniu danych z wielu połączonych tabel będzie wolniejsze niż pobieranie danych z jednej tabeli, która jednak może zawierać nadmiarowe dane. Jest więc to kompromis pomiędzy zajętym miejscem przez bazą danych, a wydajnością skomplikowanych zapytań pobierających dane z wielu połączonych tabel.

Baza danych Serwisu Kucharskiego została od początku zaprojektowana w optymalny sposób, z uwzględnieniem obiektów przechowywanych w serwisie, i jej obecna postać jest wystarczająca dla potrzeb serwisu, nie ma więc potrzeby jej dodatkowej denormalizacji.

5. Projektowanie operacji na danych.

Do przechowywania danych w aplikacji została użyta baza danych PostgreSQL. Dostęp do takiej bazy z wewnątrz aplikacji można najprościej uzyskać za pomocą natywnych zapytań w języku SQL. Autorzy zdecydowali się jednak na skorzystanie z pomocy narzędzia do mapowania relacyjno-obiektowego Hibernate.

Rozwiązanie to zapewnia programiście znaczną pomoc przy tworzeniu zapytań do bazy danych, bowiem wynikiem zapytania jest gotowy obiekt (lub lista obiektów). Kolejną zaletą są wszystkie operacje modyfikujące dane w bazie, gdyż znów odwołujemy się bezpośrednio do obiektów a nie tabel w bazie danych. Kod źródłowy jest dzięki temu znacznie bardziej przejrzysty i prostszy do zrozumienia (zwłaszcza w przypadku bardziej skomplikowanych zapytań, na przykład z wielu połączonych tabel).

Przykładowo aby utworzyć TODO: w języku SQL należałoby:

```
Class.forName("org.postgresql.Driver");
Connection connection = DriverManager.getConnection(
"jdbc:postgresql://localhost/kuchnia", "username",
"password");
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery("SELECT * from
Uzytkownicy");
while( resultSet.next() ) {
  String data = resultSet.getString( "imie" ) + " " +
resultSet.getString( "nazwisko" );
  System.out.println( data );
}
statement.close();
```

Dzięki zastosowaniu technologii Hibernate można ten sam efekt uzyskać w następujący sposób:

```
session = HibernateUtil.getSessionFactory().openSession();
sessionFactory = HibernateUtil.getSessionFactory();
session.beginTransaction();
List result = session.createQuery("from Uzytkownicy").list();
session.getTransaction().commit();
```

Otrzymujemy listę obiektów Uzytkownik ze wszystkimi polami, jakie znajdują się w bazię danych, możemy się odwoływać do nich jak w każdym obiekcie, np.:

```
Uzytkownik uzytkownik = result.get(0);
```

```
System.out.println(uzytkownik.getImie + " " +
uzytkownik.getNazwisko());
```

Kolejnym przykładem może być modyfikacja danych zawartych w bazie danych, dzięki zastosowaniu frameworka Hibernate odbywa się to w następujący sposób:

```
Uzytkownik user = new Uzytkownik();
user.setImie("Jan");
user.setNazwisko("Kowalski")
Transaction tx = session.beginTransaction();
session.save(user);
tx.commit();
```

Widać tutaj przewagę Hibernate'a nad JDBC - tworzony jest nowy obiekt i po prostu dodawany do bazy danych. Analogicznie można wykonywać modyfikacje danych w bazie — pobieramy dane jako obiekt, modyfikujemy wartość obiektu, po czym zapisujemy rezultat w bazie danych.

Dodatkowo przy tworzeniu bardziej skomplikowanych zapytań (przy zapytaniach łączących dane z wielu tabel) Hibernate pozwala na użycie języka HQL (Hibernate Query Language), mającego nieco uproszczoną składnię w porównaniu do SQL-a. Jeśli istnieje potrzeba użycia natywnego zapytania SQL, również istnieje taka możliwość.

Dzięki zastosowaniu frameworka Hibernate przy tworzeniu serwisu uniknięto żmudnej pracy przy pisaniu skomplikowanych zapytań SQL, przy której znacznie łatwiej o pomyłkę, niż przy tworzeniu obiektów, bądź zapytań HQL.