Marcin Apostoluk
Wydział Informatyki I Zarządzania
Politechnika Wrocławska

Wrocław, 11.11.2005

Generator danych
przeznaczony dla relacyjnej bazy danych

Abstract

Istnieje obecnie wiele uniwersalnych generatorów danych przeznaczonych dla relacyjnych baz danych. Programy te potrafią bez problemu wygenerować losowe ciągi znaków lub ciągi z danego zbioru, losowe liczby, czasem nawet odpowiednie zależności. Programy te mogą wygenerować losowe dane, lecz choć dane te spełniać będą reguły w relacyjnej bazie danych, użytkownik chciałby, aby znacznie bardziej odpowiadały one danym rzeczywistym. Niniejsza praca poświęcona jest właśnie sposobom generowania danych tak, by jak najbardziej przypominały dane rzeczywiste.

1. Wprowadzenie

Stworzenie uniwersalnego generatora danych możliwego do użycia w każdym przypadku jest problemem bardzo trudnym i złożonym. Choć istnieje możliwość wprowadzenia predefiniowanego zbioru imion, nazwisk, e-maili itp., w bardziej złożonych przypadkach zawsze pojawi się konieczność uściślenia założeń np. w kolumnie obok imienia wpisywać zawsze płeć tak, by imionom żeńskim odpowiadała płeć żeńska. Owszem istnieje możliwość wygenerowania danych spełniających ograniczenia bazy danych, lecz dane te mogą nie odzwierciedlać wystarczająco danych rzeczywistych. Na potrzeby generowania danych należy więc w każdym przypadku stworzyć osobny program, można jedynie skorzystać z przyjętego szablonu, zgromadzonych wcześniej danych, czy napisanych już gotowych funkcji.

2. Zbieranie danych

Pierwszym krokiem przy tworzeniu generatora bazy danych jest zebranie jak największej ilości gotowych danych, którymi można będzie wypełnić bazę testową. Liczne zbiory imion, nazwisk, miast, ulic, nazw firm itp. są dostępne w Internecie. Nie zawsze są one dostępne w formie gotowej do końcowego przetwarzania. Przykładowo dla zbioru imion pobranego ze strony internetowej:

<TR><TD>Absalon, Achacjusz, Achacy, Achilles, </TD></TR>
<TR><TD>Adalbert, Adam, Adamina, Adaukt, </TD></TR>
<TR><TD>Adela, Adelajda, Adelina, Ado, </TD></TR>

przetworzenie go do formatu wyjściowego:

Absalon;Achacjusz;Achacy;Achilles;Adalbert;Adam;Adamita;Adaukt;Adela;
Adelajda;Adelina;Ado;

nie stanowi żadnego problemu nawet gdy zbiór ten liczy wiele tysięcy danych. Kilkunastoliniowy program wycinający znaczniki tabelowe i izolujący dane jest bardzo prosty do napisania. Posiadając dane w takim formacie bardzo łatwo skorzystać z nich w aplikacji generującej dane.

W przypadku wielu danych, które bardzo łatwo jest wygenerować programistycznie np. numer domu, PESEL, czy e-mail, próba wyszukiwania gotowych danych jest bezcelowa. Znacznie szybsze w takich przypadkach jest napisanie prostej funkcji tworzącej losowo daną.

3. Generowanie danych dla pojedynczych tabel

Mając zebrane niezbędne dane, przystępujemy do tworzenia generatora zaczynając od tabel słownikowych oraz tabel niemających kluczy obcych do żadnej innej tabeli.

Rozważmy relację:
Studenci(indeks, imie, nazwisko, drugieImie, email, pelnyAdres)

We wcześniejszym etapie przygotowano zbiory imion, nazwisk, miast oraz ulic. Pozostałe dane muszą być generowane przez funkcje aplikacji.

Przykładowo dla pola indeks utworzona zostaje funkcja w języku Java:

```
protected String createIndex()
{
     String index = "";
     for ( int i = 0; i < Generator.INDEX_LENGTH;i++)
     {
          index += (int) ( Math.random() * 10 );
     }
     return “’” + index + “’”;
}
```

Funkcja ta tworzy losowy ciąg cyfr o zadanej długości. Wynik funkcji w zupełności wystarcza i może być potraktowany jako indeks studenta.

Nieco więcej problemów stwarza wygenerowanie losowego adresu e-mail. Na potrzeby tej operacji definiuje się zbiór domen, przykładowo:

```
String[] emails = { "o2.pl", "go2.pl", "yahoo.com",
"google.com", "interia.pl", "poczta.onet.pl", "wp.pl"
};
```

Przyjmijmy, że wygenerowane adresy e-mail będą mieć postać:
`[2 pierwsze litery imion]_[nazwisko][cyfry]@[domena]`

Aby urozmaicić adresy e-mail można wprowadzić więcej wariancji przykładowo dodając losowe cyfry przed znakim '@', co często jest wykorzystywane w rzeczywistości. Załóżmy dodatkowo, że tylko 80% studentów posiada adres e-mail. Napisanie odpowiedniej funkcji w języku Java nie stanowi już problemu:

```
protected String createEmail(
      String name, String surname )
{
      //czy generować email
      if ( Math.random() > this.EMAIL_PROBABILITY )
      {
            return "null";
      }

      String email = name.substring(0,2) + surname;
      //wybierz domenę ze zbioru
      String ending = this.emails[(int) ( Math.random() *
            this.emails.length )];

      //dopisz losowe cyfry
      int digitCount = (int) ( Math.random() * 2 );
      for ( int i = 0; i < digitCount; i++ )
      {
            email += "" + (int) ( Math.random() * 10 );
      }

      email += "@" + ending;
```

```
      //zwróć wynik
      return „'” + email + „'”;
}
```

Utworzenie danych adresowych studenta przebiegać będzie na podobnej zasadzie. Dane ulic i miast wybrane będą ze zbioru natomiast ulice, numery domów i kody pocztowe zostaną odpowiednio wygenerowane przez aplikację.

Posiadając wszystkie funkcje generujące dane dla każdego z pól, wystarczy napisać końcową pętle, która przeiteruje ilość rekordów, którą chce się utworzyć, i wypełni bazę wygenerowanymi danymi.

```
private void createStudents()
{
      for ( int i = 0; i < Generator.STUDENT_COUNT; i++ )
      {
            try
            {
                  String query = "INSERT INTO studenci
                        VALUES(%s,%s,%s,%s,%s)";
                  query = String.format( query,
                        createIndex(),
                        createName(), createSurname(),
                        createSecondName(),createEmail(),
                        createAddress() );
                  this.runQuery( query );
            }
            catch ( Exception e )
            {
                  System.out.println( "exception:" + e );
            }
      }
}
```

gdzie funkcja `runQuery(String query)` uruchamia zapytanie, a funkcje createName(), createSurname(), createSecondName() zwracają losowe dane z przygotowanego wcześniej zbioru.

4. Generowanie danych dla połączonych tabel

Wygenerowanie danych dla pojedynczej tabeli nie stanowi dużego problemu. Wytworzenie danych dla tabel połączonych również nie jest zbytnim problemem. Dla każdej tabeli należy przygotować funkcje tworzące losowe dane dla kolumn. Następnie w każdej iteracji wstawiającej wiersz do tabeli głównej (np. studenci) należy dodać fragment kodu odpowiedzialny za obsługę powiązań. Przykładowo dla tabeli *wpisyNaKursy* odnoszącej się do tabeli *studenci*, należy wygenerować odpowiednią ilość wpisów na kurs danego studenta. Oczywiście wstawiony fragment kodu i jego dalsze powiązania będą zależeć od tego, jaka relacja zachodzi pomiędzy wybranymi tabelami.

Dobrą praktyką jest zachowanie wygenerowanych danych w dowolnej kolekcji, aby możliwe było proste i szybkie późniejsze odwołanie do tych danych. Powiązania nie muszą bowiem być realizowane od razu, dane do tabeli *wpisyNaKursy* mogą zostać wygenerowane w późniejszej części programu, wtedy np. losowo wybierani będą studenci z utworzonej wcześniej kolekcji.

Zazwyczaj przed każdym uruchomieniem procesu generowania danych czyści się dane ze wszystkich tabel (z pominięciem prostych tabel słownikowych), tak by tworzone przez program dane nie powodowały konfliktów z już istniejącymi i aby utworzone dynamicznie w programie kolekcje danych odpowiadały rzeczywistemu stanowi bazy danych.

W przypadku generowania danych jednocześnie dla kilku tabel (dla połączonych relacji) należy fragment kodu odpowiadający jednej iteracji głównej pętli objąć w transakcję, tak, by ewentualny błąd (spowodowany np. wygenerowaniem danych niespełniających ograniczeń unikalności) nie powodował utworzenia tylko częściowych lub nieprawdziwych danych.

Podsumowanie

W niniejszej pracy przedstawiono jeden ze sposobów generowania danych do relacyjnych baz danych. Metoda ta, choć nie najszybsza, pozwala na jak najdokładniejsze wygenerowanie danych odpowiadających danym rzeczywistym, które będą wprowadzane do bazy. Chcąc wygenerować takie dane potrzeba włożyć nieco więcej pracy w proces generacji, lecz utworzone w ten sposób dane są nie do odróżnienia od danych rzeczywistych.

Funkcjonalności tej nie są w stanie zapewnić obecne programy (również komercyjne) generujące dane.