Modyfikacje

0. Dana jest poniższa tabela:

```
CREATE table ankiety(
  id        int  PRIMARY KEY,
  nazwisko  varchar(100) NOT NULL,
  wiek      int NOT NULL
);
```

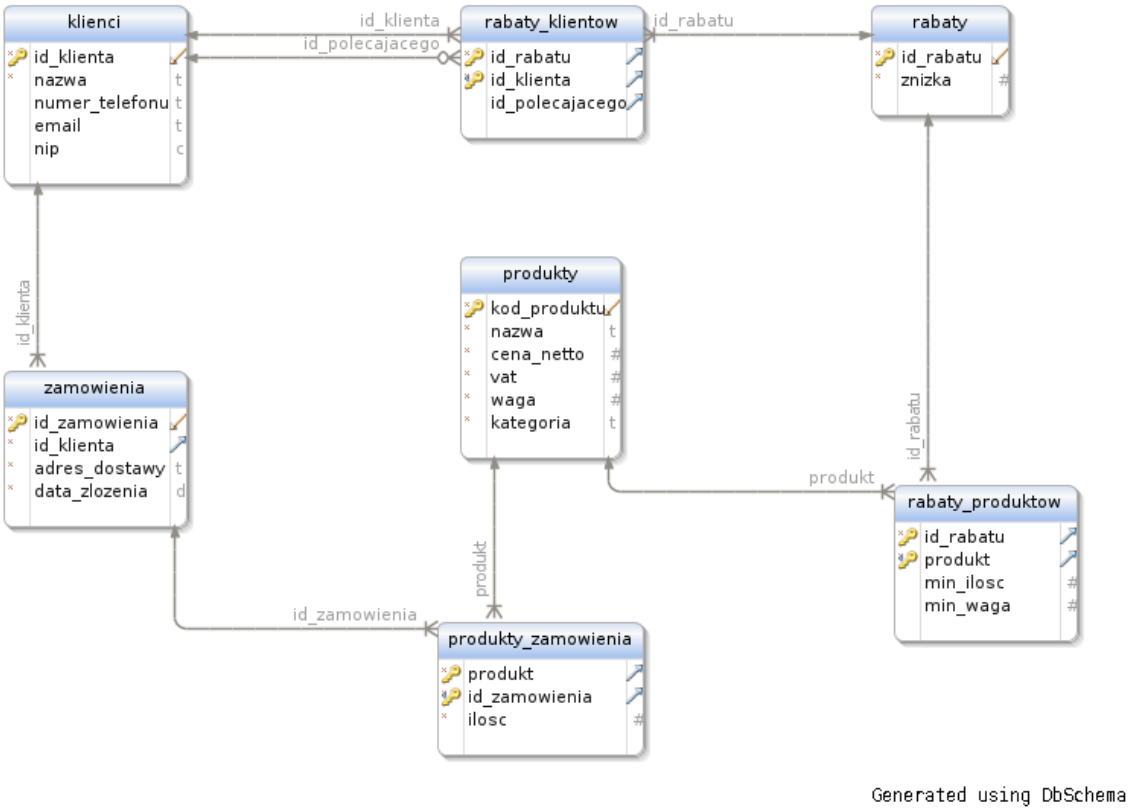
Napisz program, który na wyjściu zwróci zapytania wstawiające do tabeli *ankiety* 5000000 losowych krotek. Wygeneruj w ten sposób 3 pliki z różnymi sposobami wstawiania krotek:

- poprzez ciąg instrukcji **INSERT INTO ankieta(id,nazwisko,wiek) VALUES (?,?,?);**
- poprzez jedną instrukcję INSERT: **INSERT INTO ankieta (id,nazwisko,wiek) VALUES (?,?,?), (?,?,?), ...;**
- z użyciem instrukcji **COPY ankiety (id, nazwisko, wiek) FROM stdin;**

Zbadaj, która z powyższych metod wstawiania krotek jest najszybsza. Czym różnią się te podejścia?

To zadanie nie jest sprawdzane na Satori.

Poniższy schemat bazy sklep nie jest prawidłowy.



Wprowadź następujące zmiany:

1. Utwórz nową tabelę *kategorie(#id_kategoria, *nazwa, *vat)* do przechowywania różnych rodzajów kategorii (pole *nazwa* powinno być niepowtarzalne, pole *id_kategoria* powinno mieć typ serial). Następnie przenieść dane z kolumn *kategorie* i *vat* z tabeli *produkty* do nowoutworzonej tabeli. Można przyjąć, że istnieje zależność funkcyjna *kategoria -> vat*. Uwaga: typy kolumn *nazwa* i *vat* w tabeli *kategorie* powinny odpowiadać odpowiednim typom z tabeli *produkty*.

   Kolumny *kategorie* i *vat* w tabeli *produkty* powinny być ostatecznie usunięte, a zamiast nich powinien pojawić się niepusty klucz obcy (o nazwie *id_kategoria*) wskazujący na tabelę *kategorie*.

2. Utwórz tabelę *historia_cen(#kod_produktu, *cena_netto, #data_wprowadzenia)*. Wszystkie ceny z tabeli *produkty* przenieś do nowej tabeli przyjmując datę wprowadzenia '2000-01-01' (typ kolumny *cena_netto* powinien odpowiadać typowi z tabeli *produkty*). Usuń kolumnę *cenna_netto* z tabeli *produkty*. Kolumna *kod_produktu* z tabeli *historia_cen* powinna być kluczem obcym na tabelę *produkty*. Zabronione jest wykonywanie instrukcji INSERT.

3. Usuń tabelę *klienci* w taki sposób aby wszystkie wartości klucza obcego *id_klienta* w tabeli *zamowienia* zostały ustawione na null. Natomiast wszystkie krotki w tabeli *rabaty_klientow* odwołujące się do tabeli *klienci* powinny zostać usuniete. Nie wolno modyfikować danych

w tabelach *zamowienia* i *rabaty_klientow*. Można jedynie wprowadzać zmiany do struktury tych tabel.

Dane są następujące tabele:

```
CREATE TABLE etaty (
  etat  varchar(100) PRIMARY KEY,
  widelki numeric(8,2)[]  NOT NULL
);

CREATE TABLE pracownicy (
  id  int PRIMARY KEY,
  imie  varchar(100) NOT NULL,
  nazwisko varchar(100) NOT NULL,
  etat  varchar(100) NOT NULL REFERENCES etaty,
  pensja  numeric(8,2)  NOT NULL
);
```

Następujące zadania 4-9 dotyczą powyższych tabel. Dla tych zadań należy przyjąć, że rozwiązanie i-tego zadania korzysta z rozwiązania zadań {4,..,i-1}. Oznacza to, że do rozwiązania zadania i należy dołączyć rozwiązania zadań {4,..,i-1}.

W poniższych zadaniach wymagane jest użycie instrukcji UPDATE.

4. Tabela *etaty* nie jest w I postaci normalnej. Aby to naprawić, zamień kolumnę *widelki* na dwie kolumny *pensja_od* i *pensja_do*. Można założyć, że *widelki* są dwuelementową tablicą ale niekoniecznie uporządkowaną. Dodaj warunek $pensja\_od < pensja\_do$.

5. Podnieś pensje do minimalnego możliwego poziomu wszystkim pracownikom, którzy zarabiają mniej niż wskazują na to widełki płacowe ich etatów.

6. Okazało się, że są jeszcze pracownicy, którzy zarabiają więcej niż wskazują na to widełki płacowe ich etatów. Szef postanowił utworzyć nowe etaty dodając do nazwy każdego istniejącego etatu słowo 'starszy' zachowując dolną granicę widełek, a górną podwajając. Wszyscy pracownicy, o których mowa w pierwszym zdaniu tego zadania powinni przejść na nowy etat (odpowiadający im etat starszy).

7. Ponieważ szef zaczyna zmieniać nazwy etatów, kolumna ta nie nadaje się na klucz podstawowy. Dodaj do tabeli *etaty* kolumnę *id_etatu* typu liczbowego (int), która powinna pełnić funkcję klucza podstawowego (kolumna etaty nie powinna być już kluczem podstawowym). Kolumna *etat* w tabeli *pracownicy* powinna być nadal prawidłowym kluczem obcym na tabelę *etaty* i wskazywać kolumnę *id_etatu*. Pracownicy nie powinni zgubić swoich etatów. Wartości w kolumnie *id_etatu* powinny być kolejnymi, różniącymi się o 10 liczbami i zaczynać się od wartości 10.

8. Okazuje się, że nadal są pracownicy zarabiający więcej niż górne widełki ich etatów. Aby to naprawić, utwórz kolumnę *dodatki* typu numeric(8,2) w tabeli *pracownicy*. Wszystkim pracownikom, o których mowa w pierwszym zdaniu przenieś nadwyżkę pensji ponad górne widełki ich etatu do pola *dodatek*. Pozostali pracownicy powinni mieć to pole ustawione na NULL.

9. Dla każdego pracownika (podaj *imie*, *nazwisko*) podaj etat (nazwa etatu), który ma najwyższą górna granicę widełek i pensja pracownika z uwzględnieniem dodatków spełnia widełki tego etatu. Należy uwzględnić wszystkie etaty spełniające powyższe warunki (wypisać w kolejnych krotkach: *imie*, *nazwisko*, *etat*). Nie wypisywać pracownika, dla którego nie istnieje odpowiedni etat.