Providing Access to Database as SaaS(Software as a Service)

Website URL : https://radiant-bayou-9206.herokuapp.com/fakenames/
Source Code : https://radiant-bayou-9206.herokuapp.com/fakenames/

How the data were collected and stored ?

Data is collected from set of fake names and information available on internet in form of text file .C++ program is designed to parse this set of information and write it in form of sql queries of INSERT Command ,inserting each tuple one by one into database table.Another table containing other aspect for individual is created which corresponds to every tuple in first table.
First Database table named **Fakenames** contains different attributes for an individual such as Gender,Title,Given Name,Middle Name, Surname, City , Country ,Email .This table consists of unique fictional individuals . It contains 1110 different entries with each tuple representing a unique individual. Second table represents a one to one mapping of email address of individuals to their unique Social Security Numbers.
with again each row or tuple representing a unique individual.This table too contains 1100 unique entries. Here the SSN number is generated by C random function and same C program is modified to push entries into second table table **FakeSSN.**

PostgreSql Database Server is used to store all the data and connector specific to psql is used to connect database to website.Both tables FakeSSN and Fakenames reside in database DATABASE_URL at heroku and in mydb on my local machine .These tables were initially populated in local database then finally pushed to heroku server using command

heroku db:push mydb DATABASE_URL

This populated the heroku database and entire data is visible on website.
This technique helps to protect data and not being exposed to corruption while developer can continue to operate on database on his local machine.

Design of Relational Database Schema

First table Fakenames defines relation between 8 attributes of individual . These attributes are Gender,Title,Given Name,Middle Name, Surname, City , Country ,Email . Here Email is primary key We can identify each individual uniquely by giving each one a unique email address. Note the asterisk (*) beside this field in the table below: it signals that the email address field is a *key field*, containing a unique value in each record. We can use that field to retrieve any specific record. When you create such a key field in a database program, the program will then prevent you from entering duplicate values in this field, safeguarding the uniqueness of each entry.email address is also the primary key in second table FakeSSN and

also serves the same purpose. It acts as foreign key as both ways as it helps to uniquely specify a tuple in other table using one attribute i.e email address.
Both the tables are linked by key field email address.

Relational Database Schema

Gender	Title	Given name	Middle Initial	Surname	City	Country	Email Address*
Female	Ms.	Lisa	M	Conway	Clandulla	Australia	LisaMConway@fleckens.hu
Female	Ms.	Andrea	M	Luczak	Kooragang	Australia	AndreaMLuczak@cuvox.de

Figure A: Fakenames

Email Address*	Social Security Number
LisaMConway@fleckens.hu	2007
AndreaMLuczak@cuvox.de	2049

Figure B: FakeSSN

Major Components of Application

This Web application is designed using ruby on Rails framework.
It can be mainly divided into three main components:

1. Models
2. Views
3. Controller

1. **Models**

Models consists of backend of application which comprises of database which contains both of our tables Fakenames and FakeSSNs.They reside in Database_URL databse in postgresql database server.

2. **Views**

This defines the HTML/CSS pages which acts as user interface for different workflows.User inputs their query params through this for updation/deletion/insertion. This queries are passed onto controller via params hash.
index views displays both the database tables .
new page user can add entries for new individual and insert into database using submit button.
User can delete by searching database using primary key i.e emailaddress through edit view.
View are spoon fed by controller what to display on interface as controllers receive query params from views and queries to model and send results back to Views for user to see.

3. **Controllers**

It has following functions
Index : it displays all entries from respective table by SELECT * query and display on index view page.
Select : This function receives email address or surname as search query and display tuple or multiple tuples to the edit view page
Update : it receives all entries via params and then queries the tuple via email address updating the particular tuple
Delete: Deleting the tuple by search query using email address as primary key
Create: Inserts a new tuple based on param entries from new View page .

Difficulties and challenges

a. Accumulating such database: Creating such database in specific schema and then pushing in database in form of sql queries. it is not possible manually as it requires lot of work. So I created a script which parses through massive data and arrange them into Insert queries. I used ifstream and instream to read text documents and write them into sql queries.
b. Making insertion and deletion consistent : Addition/Insertion/Deletion in one table also needs to be communicated to the other table . Deleting tuple in Fakenames should also be made sure that it is deleted in Fake SSN . For that deleted entry primary key is recorded and then queried in other table to delete the entries in second table. Same principle was applied to updation and insertion as well.