Screaming Beckers: Daniel Stanev, Harry Tran

Dr. Eric Becker

CS 4347.501

1 Dec 2023

Data Generation

Table Size

We kept the smaller tables like ‘librarian’ to less than 10 entries. This allows us to keep things simple while showing how employees, or librarians specifically, interact with the library. For the larger tables, such as 'Media', we went with over 200 entries because we wanted to give a good feel for the library's variety of stuff, like books, movies, etc. Our database is not as large as a real library (as that goes beyond the scope of this project). Still, the distinction of sizes between ‘employee and ‘Media’ illustrates the relationship between these entities and makes the database truly feel like a scaled-down version of an actual library.

Population Generation

We used the Python Faker library to create our database data. It's a simple and effective tool that lets us quickly make realistic entries for our tables, like coming up with names for all the people in our database and generating made-up books, movies, etc. Faker was perfect for the larger tables, getting a good mix of user profiles and media items, making our project feel more real without the hassle of having to manually come up with realistic-sounding data. For some of the smaller tables, I entered data by hand to make it easier because it used less brain power than having to configure the script to work around partially filled-in tables.

Data Generation Script

```
from faker import Faker
import mysql.connector
from datetime import date

# Replace these with your database details
host = "localhost"
user = "root"
password = "password"
database = "newschema"

# Establish a connection
connection = mysql.connector.connect(
    host=host, user=user, password=password, database=database
)

cursor = connection.cursor()

# Use Faker to generate mock data
fake = Faker()

# Insert data into the database

# Person
for i in range(220):
    Person_ID = i + 10000
    first_name = fake.first_name()
    last_name = fake.last_name()
    address = fake.street_address()
    city = fake.city()
    state = fake.state_abbr()

    query = "INSERT INTO person (Person_ID, first_name, last_name, Street_Address, City, State) VALUES (%s,
%s, %s, %s, %s, %s)"
    values = (Person_ID, first_name, last_name, address, city, state)

    cursor.execute(query, values)

# Employee
arr = []
for i in range(20):
    Person_ID = fake.random_int(6, 220) + 100000

    while Person_ID in arr:
        Person_ID = fake.random_int(6, 220) + 100000

    arr.append(Person_ID)
    supervisor = fake.random_int(2, 4)
    jobs = ["librarian", "clerk", "volunteer"]
    if supervisor == 2:
        job = jobs[0]
    elif supervisor == 3:
        job = jobs[1]
    elif supervisor == 4:
        job = jobs[2]

    supervisor += 100000
    query = "INSERT INTO employee (Person_ID, Supervisor, Job) VALUES (%s, %s, %s)"
    values = (Person_ID, supervisor, job)

    cursor.execute(query, values)

# Librarian
table_name = "librarian"
```

```
select_query = f"SELECT Person_ID, Policy_Develop FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    Person_ID, Policy_Develop = row
    degree_level = fake.random_element(
        elements=("Associate", "Bachelor", "Master", "Doctorate")
    )
    insert_query = (
        f"UPDATE librarian SET degree = %s WHERE Person_ID = %s AND Policy_Develop = %s"
    )
    cursor.execute(insert_query, (degree_level, Person_ID, Policy_Develop))

# Clerk
table_name = "clerk"
select_query = f"SELECT Person_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    Person_ID = row[0]
    regNum = fake.random_int(1, 5)
    tComp = fake.random_int(0, 1)
    insert_query = f"UPDATE clerk SET Register_No = %s, training_complete = %s WHERE Person_ID = %s"
    cursor.execute(insert_query, (regNum, tComp, Person_ID))

# Volunteer
table_name = "volunteer"
select_query = f"SELECT Person_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    Person_ID = row[0]
    depart = fake.random_int(2, 4)
    courtMan = fake.random_int(0, 1)
    insert_query = f"UPDATE volunteer SET Department = %s, Court_mandated = %s WHERE Person_ID = %s"
    cursor.execute(insert_query, (depart, courtMan, Person_ID))
"""
# Supervision

SELECT e.Person_ID AS FK_Sub_ID, e.Supervisor AS FK_Super_ID
    FROM employee e
    LEFT JOIN supervisor s ON e.Person_ID = s.Person_ID;
"""


# Media

book_genres = [
    "Fiction",
    "Non-Fiction",
    "Mystery",
    "Science Fiction",
    "Fantasy",
    "Thriller",
    "Romance",
    "Historical Fiction",
]
movie_genres = [
    "Action",
    "Comedy",
```

```
    "Drama",
    "Horror",
    "Science Fiction",
    "Fantasy",
    "Documentary",
    "Animation",
]
music_genres = [
    "Rock",
    "Pop",
    "Hip Hop",
    "Jazz",
    "Country",
    "Electronic",
    "Classical",
    "R&B",
]
article_genres = [
    "Technology",
    "Science",
    "Health",
    "Business",
    "Politics",
    "Entertainment",
    "Sports",
    "Travel",
    "Lifestyle",
    "Fashion",
    "Food",
    "Arts",
    "Education",
    "Opinion",
    "History",
    "Environment",
    "Music",
    "Film",
    "Literature",
    "Gaming",
]
genre_arr = [book_genres, movie_genres, music_genres]

for i in range(240):
    medID = i + 20000
    title = fake.sentence(nb_words=5)
    # Pick a random genre category (books, movies, music)
    genre_category = fake.random_element(elements=genre_arr)

    # Pick a random genre from the selected category
    genre = fake.random_element(elements=genre_category)

    pub = fake.company()
    totalStock = fake.random_int(3, 6)
    availStock = fake.random_int(0, 2)
    medType = fake.random_element(elements=("Audiobook", "book", "music", "movie"))

    query = "INSERT INTO media (Media_ID, Title, Genre, Publisher, Total_Stock, Available_Stock, Media_Type)
VALUES (%s, %s, %s, %s, %s, %s, %s)"
    values = (medID, title, genre, pub, totalStock, availStock, medType)

    cursor.execute(query, values)


# Audiobook
```

```
cursor = connection.cursor()
table_name = "music"
select_query = f"SELECT Media_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    medID = row[0]
    art = fake.name()
    alb = fake.sentence(3)
    length = fake.time()
    # prodCo = row[2]

    insert_query = f"UPDATE music SET Artist = %s, Album = %s, Song_Length = %s WHERE Media_ID = %s"
    cursor.execute(insert_query, (art, alb, length, medID))


cursor = connection.cursor()
table_name = "book"
select_query = f"SELECT Media_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    medID = row[0]
    art = fake.name()
    isbn = fake.random_int(1111111, 9999999)
    # prodCo = row[2]

    insert_query = f"UPDATE book SET Author = %s, ISBN = %s WHERE Media_ID = %s"
    cursor.execute(insert_query, (art, isbn, medID))


table_name = "article"
select_query = f"SELECT Media_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    medID = row[0]
    art = fake.company()

    # prodCo = row[2]

    insert_query = f"UPDATE article SET Journal = %s WHERE Media_ID = %s"
    cursor.execute(insert_query, (art, medID))

cursor = connection.cursor()
table_name = "authors"
select_query = f"SELECT FK_Media_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    medID = row[0]
    art = fake.name()

    # prodCo = row[2]

    insert_query = f"UPDATE authors SET Author = %s WHERE FK_Media_ID = %s"
    cursor.execute(insert_query, (art, medID))

cursor = connection.cursor()
```

```
table_name = "customer"
select_query = f"SELECT Person_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()

for row in existing_data:
    medID = row[0]
    numBorrow = fake.random_int(0, 7)

    # prodCo = row[2]

    insert_query = f"UPDATE customer SET Num_Borrowed = %s WHERE Person_ID = %s"
    cursor.execute(insert_query, (numBorrow, medID))

table_name = "borrow"
select_query = f"SELECT FK_Person_ID FROM {table_name}"
cursor.execute(select_query)
existing_data = cursor.fetchall()
end_date = date(2024, 12, 31)
itemID = 1
for row in existing_data:
    medID = row[0]
    itemID += 1
    # fakedate = fake.date_between(start_date='today', end_date=end_date)

    insert_query = "UPDATE borrow SET itemID = %s WHERE FK_Person_ID = %s"
    cursor.execute(insert_query, (itemID, medID))


# Rent
end_date = date(2024, 3, 31)
for i in range(12):
    Person_ID = fake.random_int(100221, 100400)
    first_name = fake.random_int(1, 5)
    fakedate = fake.date_between(start_date="today", end_date=end_date)

    query = "INSERT INTO rent (FK_Person_ID, roomNum, Date) VALUES (%s, %s, %s)"
    values = (Person_ID, first_name, fakedate)

    cursor.execute(query, values)

connection.commit()
cursor.close()
connection.close()
```

Listing of Tables

{'TABLE_NAME': 'article', 'TABLE_ROWS': 20, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'associations', 'TABLE_ROWS': 8, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'audiobook', 'TABLE_ROWS': 70, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'authors', 'TABLE_ROWS': 20, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'book', 'TABLE_ROWS': 119, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'borrow', 'TABLE_ROWS': 150, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'clerk', 'TABLE_ROWS': 10, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'customer', 'TABLE_ROWS': 180, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'employee', 'TABLE_ROWS': 24, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'keywords', 'TABLE_ROWS': 20, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 16384, 'total_size': 32768}

{'TABLE_NAME': 'librarian', 'TABLE_ROWS': 7, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'media', 'TABLE_ROWS': 240, 'DATA_LENGTH': 65536, 'INDEX_LENGTH': 0, 'total_size': 65536}

{'TABLE_NAME': 'movie', 'TABLE_ROWS': 56, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'music', 'TABLE_ROWS': 45, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'person', 'TABLE_ROWS': 400, 'DATA_LENGTH': 65536, 'INDEX_LENGTH': 0, 'total_size': 65536}

{'TABLE_NAME': 'private_room', 'TABLE_ROWS': 5, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'rent', 'TABLE_ROWS': 12, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 16384, 'total_size': 32768}

{'TABLE_NAME': 'routingnum', 'TABLE_ROWS': 18, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 16384, 'total_size': 32768}

{'TABLE_NAME': 'schedule', 'TABLE_ROWS': 20, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'supervision', 'TABLE_ROWS': 23, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 16384, 'total_size': 32768}

{'TABLE_NAME': 'supervisor', 'TABLE_ROWS': 4, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}

{'TABLE_NAME': 'volunteer', 'TABLE_ROWS': 5, 'DATA_LENGTH': 16384, 'INDEX_LENGTH': 0, 'total_size': 16384}