Final Project

Arunima Sen
Instructor: Prof. Sudmanns

1 Introduction and Methods

I chose the Coachella Valley Music and Arts Festival(or simply, Coachella) for my project. In designing the database for Coachella, I first identified key entities essential for managing the event and these are: `Stages`, `Food Areas`, `Facilities`, `Vendors`, `Emergency Exits`, `Visitors`, and `Artist Lineups`.

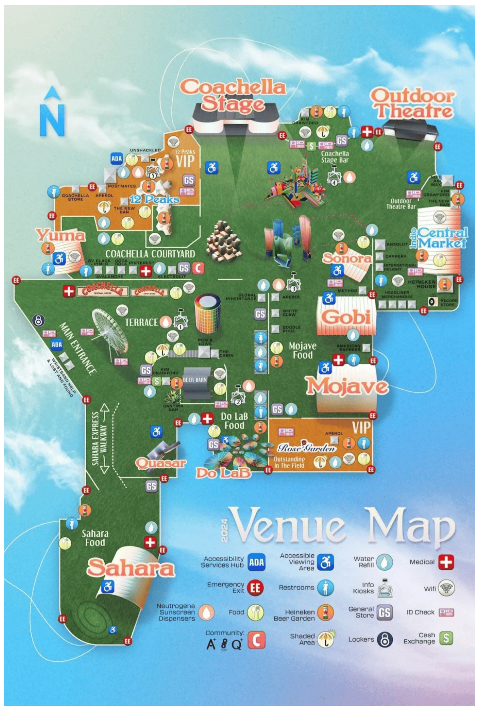

Figure 1: Layout

I then defined attributes for each entity to capture relevant information. For example, `Stages` have attributes like `capacity` and `music type`, while `Food Areas` include `cuisine type` and `seating capacity`. I included foreign keys to establish some (dependent) relationships, such as linking stages to their nearest food areas and emergency exits. Then, I established relationships to reflect real-world interactions - for instance, `Stages` host multiple `Artist Lineups`, and `Food Areas` contain multiple `Vendors`. Also, `Visitors` interact with multiple `Stages`, `Food Areas`, and `Facilities`. I used dbdiagram.io for coming up with the UML diagram and I was unable to visually represent the many-to-many relationships involving visitors on this diagram. These M:N relationships are conceptually important, as we discussed in class.

I used ChatGPT to allocate spatial data. I gave it the bounding coordinates of the venue and instructed it to place emergency exits on the boundary and randomly allocate points within the boundary for all other entities. This helped me overcome the problem I faced while manually creating shapefiles in QGIS. I also asked ChatGPT to generate realistic pseudodata as instructed in the project. For the artist lineup, I used some data entries from this website that had the past schedule.

1.1 Database Replication

To replicate my database, please run the following files in order:

- `1create_tables.sql`
- `2pseudodata.sql`
- `3artists.sql`
- `4sample_queries.sql`

2 Model

Entity	**Attributes**
Stage	stage_id (PK), name, capacity, music_type, emergency_exit_id (FK), nearest_food_area_id (FK)
Food area	food_area_id (PK), name, cuisine_type, seating_capacity, emergency_exit_id (FK)
Facility	facility_id (PK), type, emergency_exit_id (FK)
Vendor	vendor_id (PK), name, type, opening_time, closing_time, food_area_id (FK), emergency_exit_id (FK)
Emergency exit	exit_id (PK), zone
Visitor	visitor_id (PK), access_level
Artist lineup	lineup_id (PK), artist_name, performance_time, stage_id (FK)

Table 1: Entities and attributes

Class	**Relationship**	**Class**
Stage	hosts/has (1:N)	Artist lineup
Food area	contains (1:N)	Vendor
Stage	near (N:1)	Food area
Stage	near (N:1)	Emergency exit
Food area	near (N:1)	Emergency exit
Facility	near (N:1)	Emergency exit
Vendor	near (N:1)	Emergency exit
Vendor	located in (N:1)	Food area
Visitor	visits (M:N)	Stage
Visitor	visits (M:N)	Food area
Visitor	uses (M:N)	Facility

Table 2: Relationships

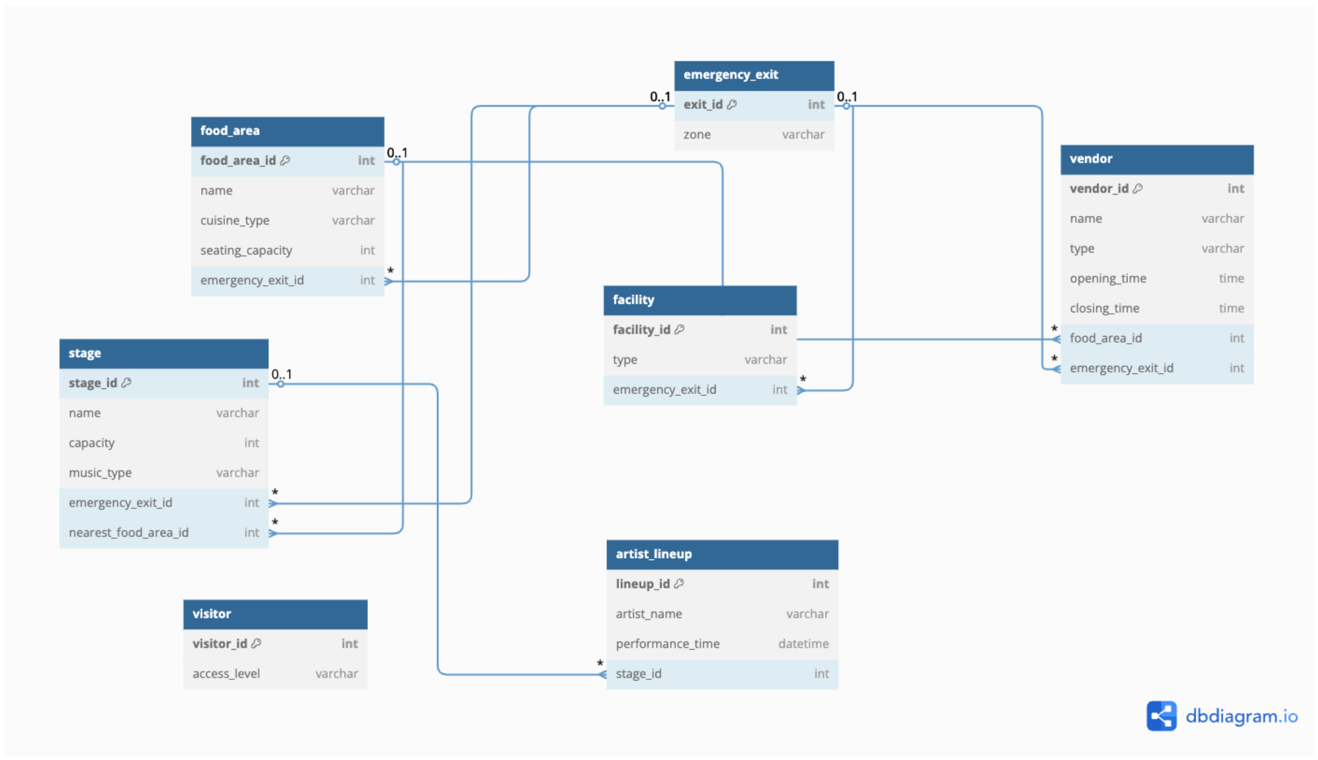
emergency_exit
exit_id int
zone varchar
food_area
food_area_id int
name varchar
cuisine_type varchar
seating_capacity int
emergency_exit_id int
vendor
vendor_id int
name varchar
type varchar
opening_time time
closing_time time
food_area_id int
emergency_exit_id int
facility
facility_id int
type varchar
emergency_exit_id int
stage
stage_id int
name varchar
capacity int
music_type varchar
emergency_exit_id int
nearest_food_area_id int
artist_lineup
lineup_id int
artist_name varchar
performance_time datetime
stage_id int
visitor
visitor_id int
access_level varchar
0..1
0..1
0..1
0..1
*
*
*
*
*
*
*
dbdiagram.io

Figure 2: UML