

Spatial Databases | Final Project

Database Backend for a WebGIS

29th March 2023

Rama Kamala Rajeswari, PARASA



Table of Contents

Introduction	3
Modelling entities and relationships	4
Identifying entities	4
Building relationships	4
Data tables	6
Usage of the database	12
Requirements and Scenarios	12
Making queries accessible and efficient	14
Indexes	14
Views	14
References	15

Introduction

Visakha Utsav is an annual festival hosted by the coastal city of Visakhapatnam, India. It is held in the month of December and is usually 3 days long. The event displays the best city has to offer including folk music and dance and local delicacies. Several local and tribal artisans set up stalls in the exhibition arena and their works are the charm of the festival. The festival is filled with performances and games. If you happen to be in the city around that time of the year, make sure to not miss it!

In this project, I build a database prototype that can potentially serve as a backend for the WebGIS application built for visitors for easy information access. The app would enable the visitors to easily navigate the festival arena and also to make the most of their visit.

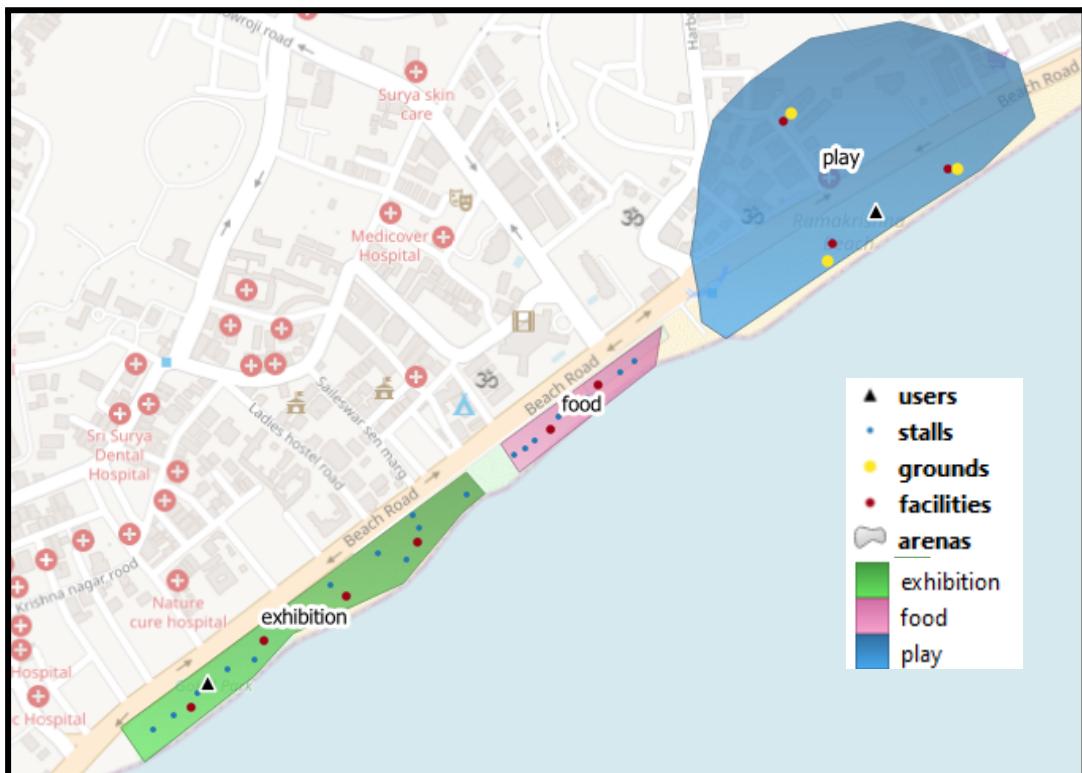


Figure 1: Festival area and places of interest

Modelling entities and relationships

Identifying entities

To start with identifying the different entities, I look for information about the festival online to get ideas.

ACTIVITIES

What You Should Not Miss At Visakha Utsav:

Games including beach soccer, volleyball, treasure hunt
Exhibition stalls with traditional handlooms and local art and crafts
Food courts serving local and world delicacies
Culture and entertainment shows with local music, dance and stand up comedy

Source: [Ministry of Tourism, Govt of India](#)

Based on the information online, and my personal experience of visiting the festival, I come up with the following entities and their attributes:

1. Events (name, type, start and end times, venue name and location)
2. Facilities (name, type [- bathroom, helpdesk, etc], location)
3. Vendor (name, stall location)
4. Stall (name, type [-food, exhibition], assigned vendor)
5. Grounds (name, location)
6. Arenas (name, type [- exhibition, play, food], arena extent)
7. Users (first name, last name, live location, current time)
 - Note: It is not necessary to maintain a user database in the scope of this database, because the app will be used only for going over general information about the festival. However, for demonstration of queries, this entity is modelled. In actual implementation, live device location and time can be used.

Building relationships

In the entities identified above, two prominent relationships are necessary to be considered. One, the relationship between stalls and the vendors that will be set up there. A vendor can set up at multiple stall locations throughout the exhibition arena. Two, the relationship between events and the grounds where they will be held at. The grounds can host different events at different times. However, one event is held at only one ground location. Based on the entities identified, and the relationships defined, I build the entity-relationship diagram using the entity-relationship tool of pgAdmin.

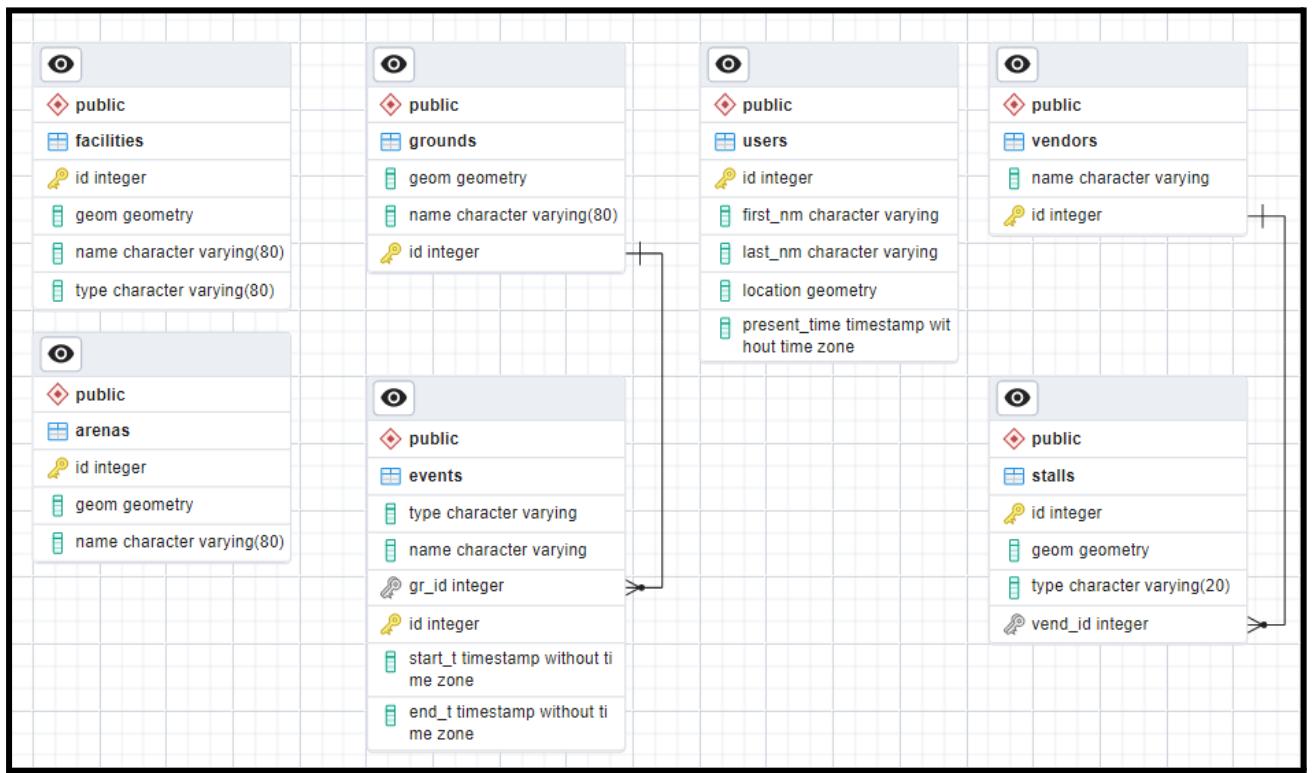


Figure 2: Design diagram of the proposed database prototype

Data tables

1. Arenas

	id [PK] integer	geom geometry	name character varying (80)
1		0103000020110...	exhibition
2		0103000020110...	food
3		0103000020110...	play

Figure 3: Snapshot of festival arenas table

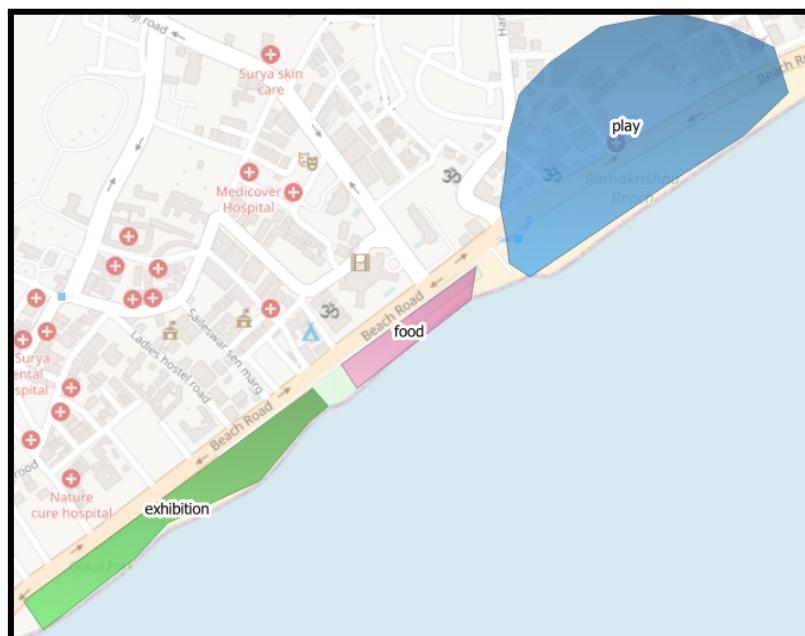


Figure 4: Map of the festival arenas

2. Facilities

	id [PK] integer	geom geometry	name character varying (80)	type character varying (80)
1	1	0101000020110...	Toilets 1	toilet
2	2	0101000020110...	Helpdesk 1	helpdesk
3	3	0101000020110...	Toilets 2	toilet
4	4	0101000020110...	Drinking Water 1	free drinking water
5	5	0101000020110...	Helpdesk 2	helpdesk
6	6	0101000020110...	Drinking Water 2	free drinking water
7	7	0101000020110...	Helpdesk 3	helpdesk
8	8	0101000020110...	Toilets 3	toilet
9	9	0101000020110...	Drinking Water 3	free drinking water

Figure 5: Snapshot of the facilities table

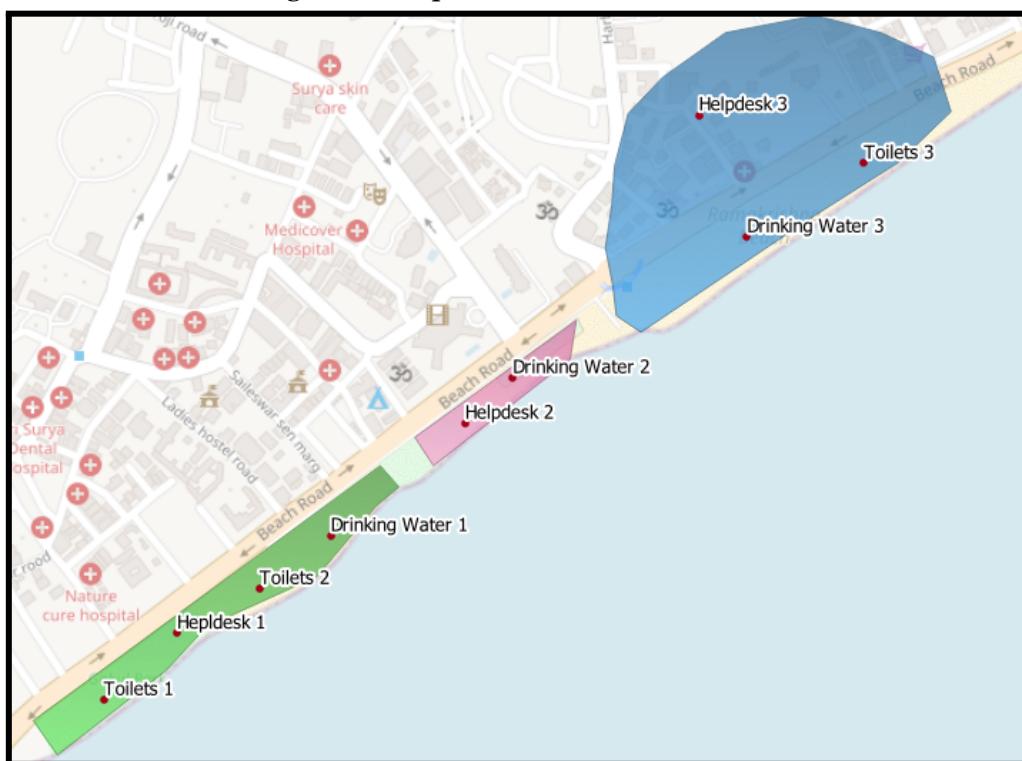


Figure 6: Map of the facilities in the festival area

3. Grounds

	geom geometry	name character varying (80)	id [PK] integer
1	0101000020110F0000039F...	Ground A	1
2	0101000020110F00000862...	Ground B	2
3	0101000020110F0000115...	Ground C	3

Figure 7: Snapshot of the grounds table

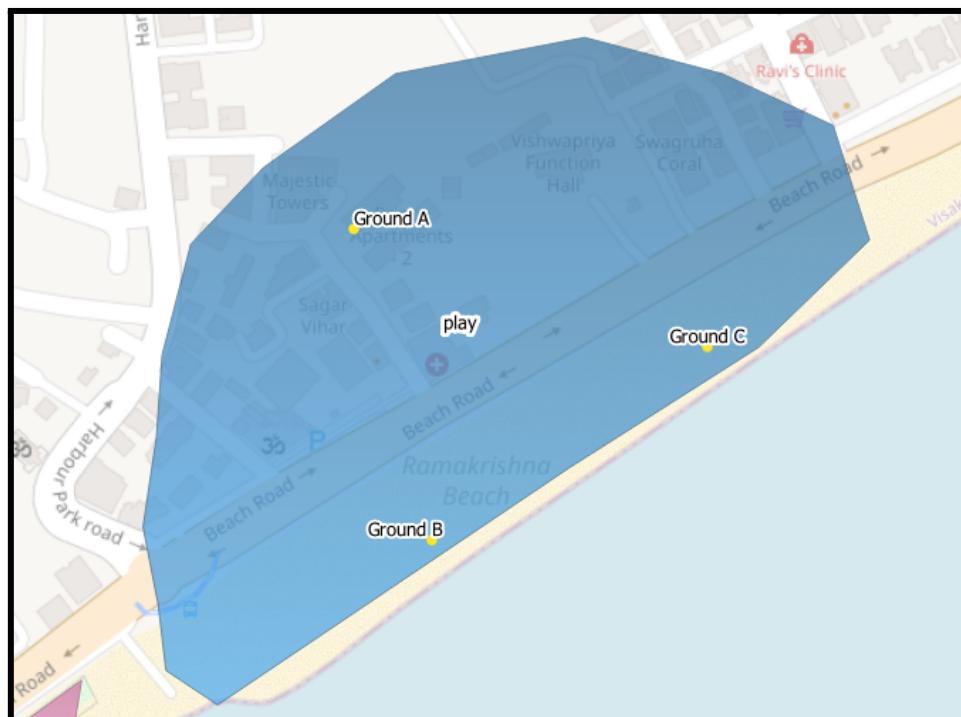


Figure 8: Map of the grounds/venues for events

4. Stalls

	id [PK] integer	geom geometry	type character varying (20)	vend_id integer
1	1	0101000020110F0000C46B1D6A60B06141213DF23ABF903E...	exhibition	1
2	2	0101000020110F0000092C956964B0614109A55D9ED5903E41	exhibition	2
3	3	0101000020110F0000854F399C68B0614130A71B9AF5903E41	exhibition	3
4	4	0101000020110F0000B58C40686EB06141C1DC9FC818913E...	exhibition	4
5	5	0101000020110F0000B3D966773B0614120C41B2D27913E41	exhibition	1

Figure 9: Snapshot of a sample of the stalls table

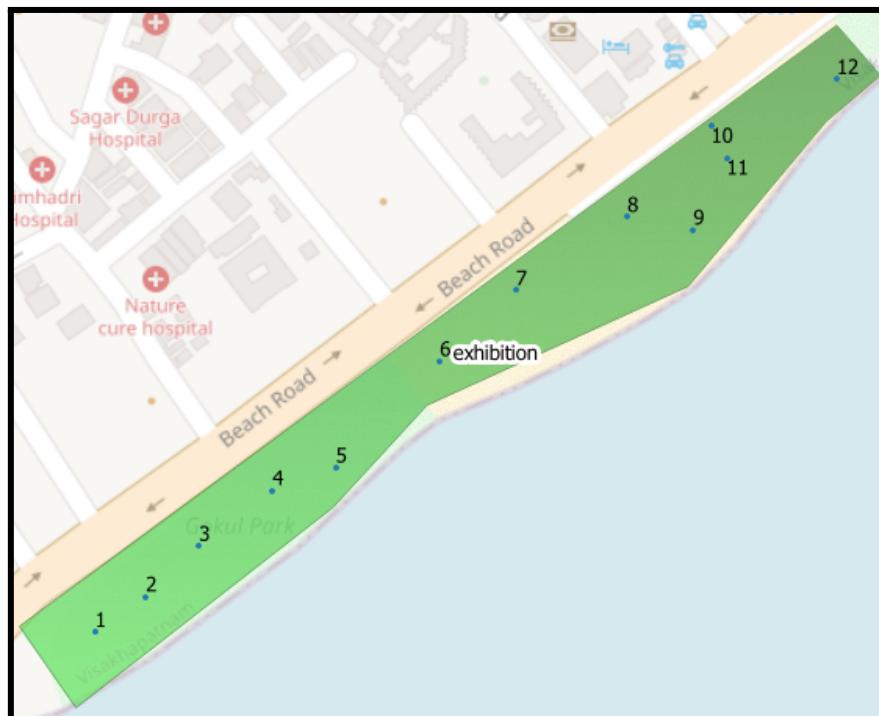


Figure 10: Map of the stalls in the festival

5. Users

	<code>id [PK] integer</code>	<code>first_nm character varying</code>	<code>last_nm character varying</code>	<code>location geometry</code>	<code>present_time timestamp without time zone</code>
1	1	Dead	Shapefiles	0101000020110F000040E106906AB06141C11192CF04913E41	2023-12-01 10:00:00
2	2	Ecstatic	Elephant	0101000020110F000012A988E4AFB0614126AF8B8AA5923E41	2023-12-02 15:00:00.736
3	3	Brilliant	Bumblebee	0101000020110F0000DF49CE5AE7B06141D91AF37CC3933E41	2023-12-03 11:00:00.32

Figure 11: Snapshot of the users table

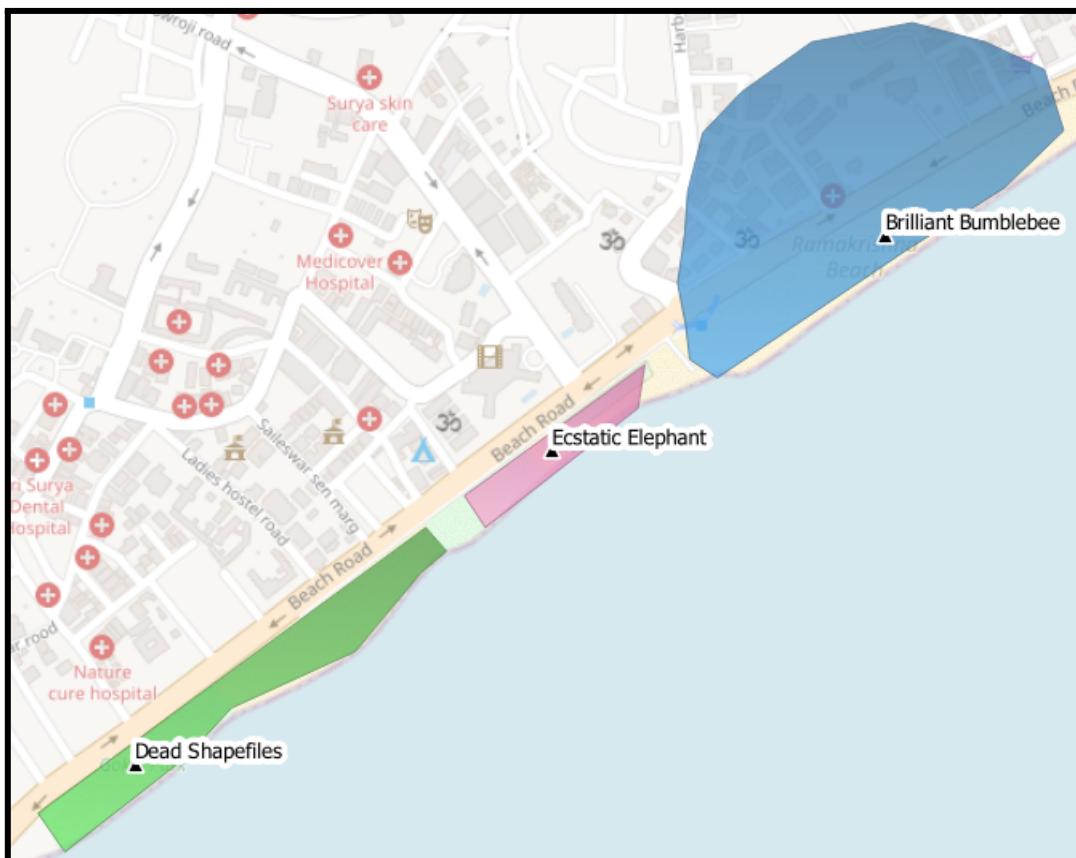


Figure 12: Map of the user locations

6. Events

	type character varying	name character varying	gr_id integer	id [PK] integer	start_t timestamp without time zone	end_t timestamp without time zone
1	Game	Beach Soccer	2	1	2023-12-01 08:00:00	2023-12-01 11:59:00
2	Game	Volleyball	1	2	2023-12-01 14:00:00	2023-12-01 18:00:00
3	Show	Folk Music	1	3	2023-12-02 09:00:00	2023-12-02 11:59:00
4	Show	Folk Dance	3	4	2023-12-02 14:00:00	2023-12-02 17:00:00
5	Game	Treasure Hunt	3	5	2023-12-03 10:00:00	2023-12-03 16:00:00

Figure 13: Snapshot of the events table

7. Vendors

	name character varying	id [PK] integer
1	vendor1	1
2	vendor2	2
3	vendor3	3
4	vendor4	4
5	vendor5	5

Figure 14: Snapshot of a sample of the vendors table

Usage of the database

Requirements and Scenarios

Requirement 1: The database should serve as a backend to organise events and venues

Scenario: Let's say that the organisers need to organise an award function on the 2nd day of the event for the winners of beach soccer and volleyball matches held on the previous day. So they want to find out which venues are available for an hour from 12.30 pm to 1.30 pm. This is the time when the chief guest will be available to present the awards.

The following query allows the organisers to find the grounds that don't have an ongoing event during the mentioned time. *Refer file: q1.sql*

The screenshot shows a SQL query editor interface. The top pane displays the following SQL code:

```
1 --Query to find grounds that are unoccupied for a given duration
2 select grounds.name from grounds
3
4 where grounds.name not in
5 (select a.name
6 from grounds as a, events as b
7 where a.id= b.gr_id and start_t>'2023-12-02 12:30:00' and end_t<'2023-12-02 13:30:00') ;
```

The bottom pane shows the results of the query:

name
Ground A
Ground B

Figure 15: Query to identify which venues are free at a given time (i.e. no prescheduled event)

Requirement 2: Visitors should be able to find the nearby facilities and ongoing events

Scenario 1: Mr Dead Shapefiles is in the festival area and is looking for a toilet facility less than 50m from him. The following query can help him find the facility. *Refer file q2.sql*

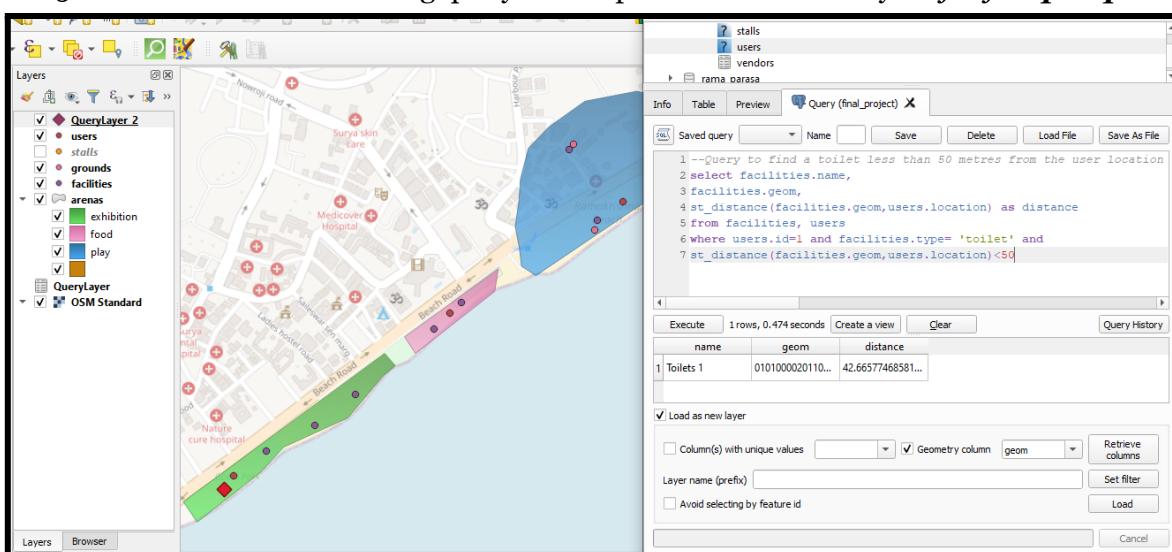


Figure 16: Query to find and display a toilet facility less than 50 m away.

Scenario 2: Ms Ecstatic Elephant is interested in finding information about the stalls less than 50m from her. So she can decide which stall to go to and plan where to spend more time. Refer file: **q3.sql**

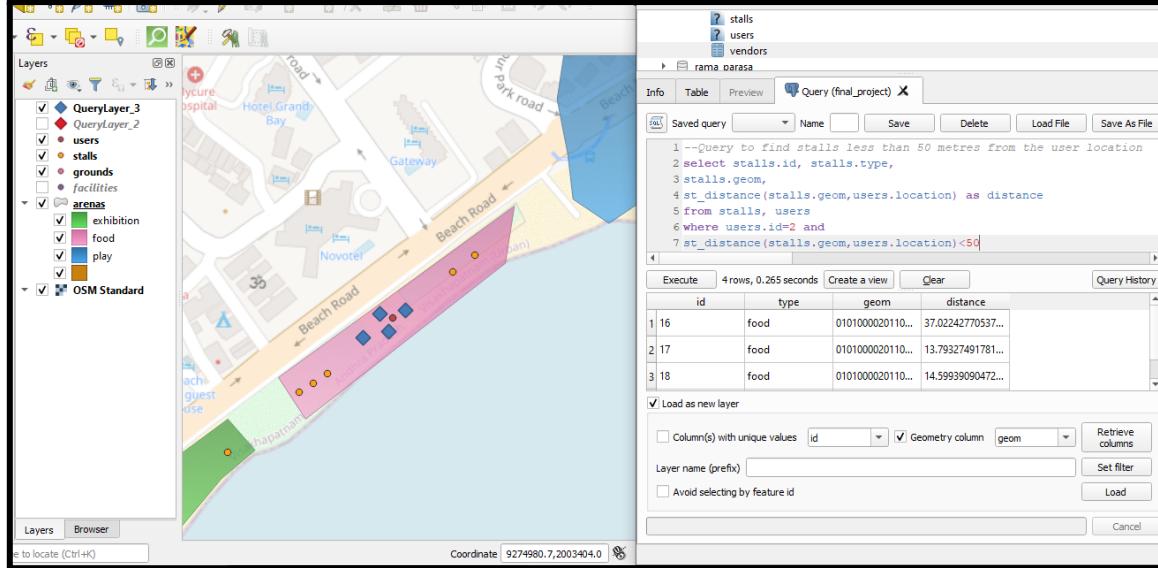


Figure 17: Query to find stalls within 50 m of distance from the visitor

Scenario 3: Ms Brilliant Bumblebee is in the play arena and wants to find out the events happening right now (at the time of her visit). The following query helps find out all the events happening at the time of her visit. Refer file: **q4.sql**

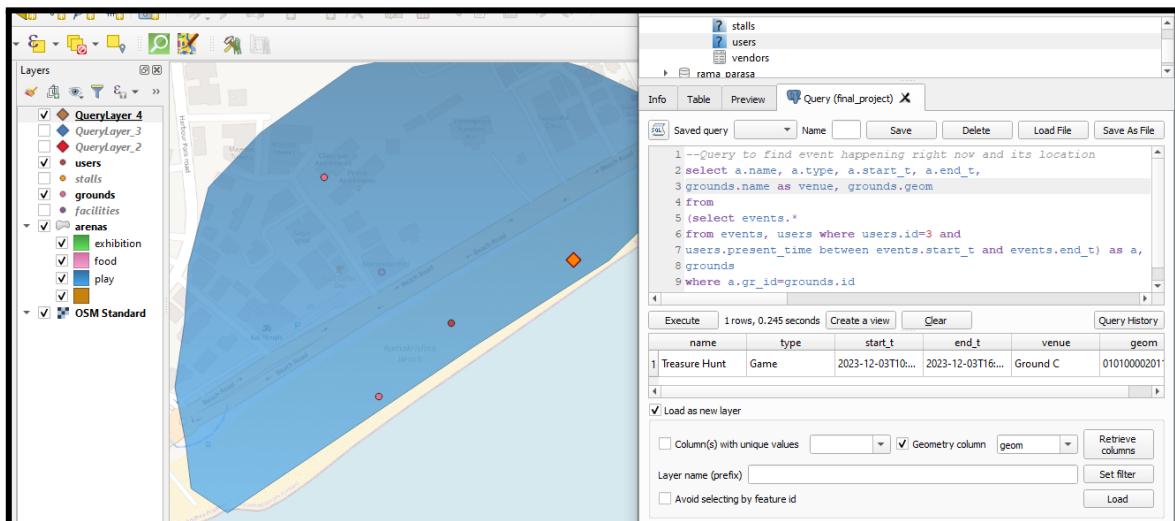


Figure 18: Query to find ongoing events and their location

Making queries accessible and efficient

Requirement 3: Display lists of events or facilities to visitors

This requirement essentially requires easy access and faster processing of

Indexes

Indexes allow for bulky queries to be processed faster. Since visitors will often pull the list of events happening in the festival, it's best to create an index on the events table. Further, visitors will also want to query the locations of various facilities. So I also create a spatial index for the facilities table.

Non-spatial index

```
Query  Query History  
1 CREATE INDEX events_idx  
2   ON events(name);
```

Spatial Index

```
Query  Query History  
1 CREATE INDEX facilities_idx  
2   ON facilities  
3   USING gist(geom);|
```

Views

Since the list of events has to be frequently displayed to visitors, we can store the query as a view. And views essentially query the table, they automatically use the indexes generated on the table.

```
Query  Query History  
1 create view events_list as  
2 select events.*  
3 from events  
4 order by start_t desc;
```

References

- [PostgreSQL Documentation](#)
- [Amit Jha, Stackoverflow](#)
- [Ministry of Tourism, Govt of India](#)
- [Spatial Indexing – Introduction to PostGIS](#)

