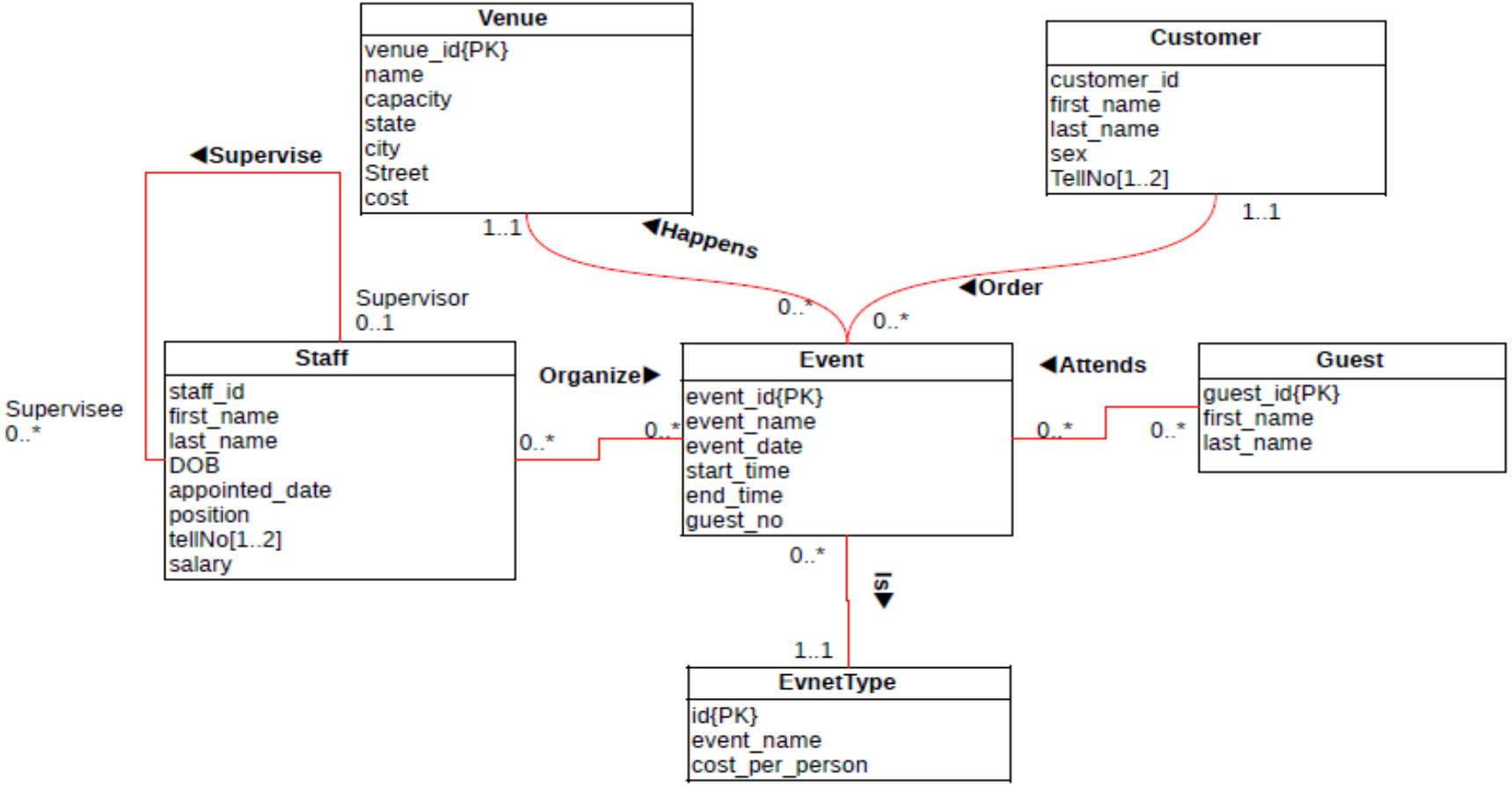
Event Organizing RDBMS

Advanced Database

Use of the database

- The purpose of this project is to develop and implement an event organizing database system that can provide user-friendly solution that can facilitate and improve event organizing. The system will allow users to create and manage events, book and allocate venues, register and communicate with attendees, and perform other related tasks. Additionally, on the staff side of the program, staff members will have the capability to view their assigned tasks, access and update their personal information. This comprehensive solution aims to streamline event management processes, enhance efficiency, and provide a seamless experience for both organizers and staff members.


Venue
venue_id{PK}
name
capacity
state
city
Street
cost
Customer
customer_id
first_name
last_name
sex
TellNo[1..2]
Supervise
Supervisor
0..1
Supervisee
0..*
Staff
staff_id
first_name
last_name
DOB
appointed_date
position
tellNo[1..2]
salary
1..1
Happens
0..*
0..*
Order
1..1
Organize
0..*
0..*
Event
event_id{PK}
event_name
event_date
start_time
end_time
guest_no
Attends
0..*
0..*
Guest
guest_id{PK}
first_name
last_name
0..*
Is
1..1
EvnetType
id{PK}
event_name
cost_per_person

Table's in the DataBase

Customer (Id, first_name, last_name, tellNo1,tellNo2,password)

Event(Id, event_name, type_id, venue_Id, customer_id, event_date, start_time, guest_no)

EventGuests(event_id, guest_id)

EventStaff(event_id, staff_id)

EventType(id, type_name, cost_per_person)

Guest(id, first_name, last_name)

Staff (id, first_name, last_name, sex, position_id, supervisor_id, DOB, supervisor_id)

StaffPosition(id, position, salary)

Venue(Id, name, capacity, state, city, street, price)

Views

- View: is a virtual table based on the result set of a SELECT query.

  There are two view in the database which are

Given_Task：

```
CREATE OR REPLACE VIEW given_tasks AS
SELECT
    staff_id,
    event_id,
    event_name,
    venue.name AS venue_name,
    event_date,
    guest,
    INITCAP(customer.first_name) || ' ' || INITCAP(customer.last_name) AS customer,
    customer.tellNo1
FROM
    staff
JOIN
    eventstaff ON staff.id = eventstaff.staff_id
JOIN
    event ON event.id = eventstaff.event_id
JOIN
    venue ON event.venue_id = venue.id
JOIN
    customer ON event.customer_id = customer.id;
```

Show_Event_Guests: CREATE OR REPLACE view show_event_guests AS SELECT event_id, guest_id, initcap(first_name) ||' '|| initcap(last_name) AS guest_name FROM event JOIN eventGuests on event.id = event_id JOIN guest on guest_id = guest.id;

Sequences

- sequence is a database object used to generate unique sequential values.

  There are

  Staff_position_seq :

```
CREATE SEQUENCE staff_position_seq
  START WITH 1
  INCREMENT BY 1
  NOMAXVALUE;
```

  customer_seq

  event_seq

  guest_seq

  staff_seq

  venue_seq

Triggers

- Triggers : are automatically executed (or fired) in response to specified database events, such as INSERT, UPDATE, DELETE, or DDL (Data Definition Language) statements like CREATE, ALTER, or DROP.

  Triggers in the database

  customer_seq:

```
CREATE OR REPLACE TRIGGER customer_seq
BEFORE INSERT ON customer
FOR EACH ROW
BEGIN
  SELECT customer_seq.NEXTVAL
  INTO :new.id
  FROM dual;
END;
```

  event_seq

  event_type_seq

  guest_seq

  staff_seq

Stored Procedures

- Stored procedures are commonly used to encapsulate business logic, perform complex data manipulation, enforce security policies, and improve performance.

  Stored procedures in the dataBase

  add_customer:

```
CREATE OR REPLACE PROCEDURE add_customer (
    p_first_name IN VARCHAR2,
    p_last_name IN VARCHAR2,
    p_password IN VARCHAR2,
    p_tellNo1 IN VARCHAR2,
    p_tellNo2 IN VARCHAR2,
    p_sex IN VARCHAR2
)
IS
BEGIN
    INSERT INTO customer (first_name, last_name, password, tellNo1, tellNo2, sex)
    VALUES (p_first_name, p_last_name, p_password, p_tellNo1, p_tellNo2, p_sex);
END;
/
```

  add_event

  add_staff

  update_customer

Indexes

- Indexes : are database structures used to improve the performance of queries by providing faster access to rows in tables.

  Indexes in the database

  search_customer:

```
CREATE INDEX search_customer
ON customer(first_name, last_name, password);
```

  search_staff:

```
CREATE INDEX search_staff ON staff(first_name, last_name, password);
```

Functions

- Functions: in Oracle are custom functions created by users to encapsulate specific logic or calculations. Functions in the database

calculate_tax:

```
CREATE OR REPLACE FUNCTION calculate_tax(tax_rate IN NUMBER, taxable_amount IN NUMBER) RETURN NUMBER IS
  tax_amount NUMBER;
BEGIN
  tax_amount := taxable_amount * (tax_rate / 100);
  RETURN tax_amount;
END calculate_tax;
/
```

The End