

# **CSCI 5408**

## **Distributed Database Builder Project**

### **Final Report Group 13**

#### **Team Members**

Himanshi Verma – B00976966

Bhavya Mukesh Dave – B00982225

Jay Jagdishbhai Patel – B00981520

Poojitha Mummadi - B00951877

**Gitlab Link:** <https://git.cs.dal.ca/himanshi/csci5408-sprint4>

## Contents

<b>Introduction.....</b>	<b>3</b>
<b>Summary of the research .....</b>	<b>6</b>
<b>Initial Entity Relationship Diagram.....</b>	<b>8</b>
<b>Design Issues in Initial ERD.....</b>	<b>9</b>
<b>Final Entity Relationship Diagram .....</b>	<b>11</b>
<b>Logical Phase.....</b>	<b>12</b>
<b>DDL Statements for the tables in two fragments .....</b>	<b>16</b>
<b>Structure of Global Data Catalog.....</b>	<b>34</b>
<b>Structure of Distributed Database .....</b>	<b>35</b>
<b>Novelty .....</b>	<b>36</b>
<b>Code execution snapshots.....</b>	<b>37</b>
<b>References .....</b>	<b>38</b>

## Introduction

We chose “**Hospitality Industry (Hotel)**” for its broad scope and the significant role it plays in global economics and customer service. This industry's complex data needs present a unique opportunity to develop a distributed database system. Understanding this, we have embarked on a thorough research phase, investigating a variety of sources ranging from official hotel and tourism websites, industry-specific research papers, insightful survey analyses, authoritative blogs, and current news articles to gather a comprehensive dataset relevant to the hospitality sector. Below is a summary that encapsulates our findings and the insightful conclusions drawn.

**Table 1:** Collected data and their sources

Collected Data	Name of Source	URL
Trends prevailing in the demanding tourist market determines trends in the modern hotel industry which can be summarized in the following general trends, changing needs of consumers, conditions of work and life, lifetime extension, growth of level of information and computerization, greater need to safeguard health (wellness, spa, organic food), the emphasis on ecology and healthy food, stay in the pure nature, growing demand for adventure facilities and excitement, visiting major events (sporting, cultural, religious, business, etc.), and new travel motivations. Hotel offers should constantly study these trends and adapt to the demands, wishes, and needs of modern consumers (customers). The breadth of the range of hotel services includes accommodation services, food and beverage services, services of recreation and sport, cultural-entertainment services, merchant services, trades, and services.	Research Gate: HOTEL MANAGEMEN T AND QUALITY OF HOTEL SERVICES (Journal of Process Managemen t – New Technolog ies, International Vol. 4, No.1, 2016.)	<a href="https://www.researchgate.net/publication/293479361_Hotel_management_and_quality_of_hotel_services">https://www.researchgate.net/publication/293479361_Hotel_management_and_quality_of_hotel_services</a>

<p>This hotel offers diverse services under one hood like different types of rooms, different types of restaurants featuring the culture of a different nation and their food, a buzzing nightlife scene, different bars with different themes, a Spa with a different type of rooms, fitness and wellness center, meeting and conference halls, seasoned planning and catering services, exquisite wedding planning, business center for private meetings, different types of Lounges, outdoor pool, housekeeping, gift shop, hair saloon, laundry, dry cleaning, convenience store, free Wi-Fi, service request, offers traveling to explore the city with full of adventures, etc.</p>	<p>JW Marriott Marquis Hotel Dubai Website</p>	<p><a href="https://www.marriott.com/en-us/hotels/dxbjw-jw-marriott-marquis-hotel-dubai/overview/?gclid=CjwKCAiA29auBhBxEiwAnKcSsqkWBChDeXrX0Q-WWcm0-QxsagRYh7yyiLNX5w89DB5AnxlcQDMh3SBoCp7AQAuD-BwE&amp;gclidsrc=aw.ds&amp;cid=PAIGLB0004YXD_GLE000BIM5_GLF000OET_A">https://www.marriott.com/en-us/hotels/dxbjw-jw-marriott-marquis-hotel-dubai/overview/?gclid=CjwKCAiA29auBhBxEiwAnKcSsqkWBChDeXrX0Q-WWcm0-QxsagRYh7yyiLNX5w89DB5AnxlcQDMh3SBoCp7AQAuD-BwE&amp;gclidsrc=aw.ds&amp;cid=PAIGLB0004YXD_GLE000BIM5_GLF000OET_A</a></p>
<p>This blog suggests how important it is to publish blogs regularly on the hotel website by hotel management. In the hospitality industry, blogging is vital to speaking to your targeted customer base. Hotel-related topics for blog posts are plentiful, and you're able to provide quick answers and additional support to your guests. Hotel blog ideas are cheaper than advertising and they last for years. In addition, hotel blogs can be interactive and fun for your guests as long as you keep coming up with creative blog topics that inspire your guests to participate in the fun.</p> <p>Hotel marketing blog ideas increase traffic to your business sites. When you are consistent with your hotel blog posts, you answer many questions before your guest can even pose the question.</p>	<p>The Content Panel</p>	<p><a href="https://thecontentpanel.com/blog-post-ideas/hotel-topics/">https://thecontentpanel.com/blog-post-ideas/hotel-topics/</a></p>

<p>The hotel also has a career website, and they advertise the open positions available to fill. So that people with the necessary skills can apply Positions like Bar attender, Security personnel, Chefs, Singers.</p>	<p>Marina Bay Sands Hotel Website</p>	<p><a href="https://www.marinabaysands.com/careers.html">https://www.marinabaysands.com/careers.html</a></p>
<p>Hotels are typically referred to by hotel type or category. The type of hotel is determined primarily by the size and location of the building structure, and then according to the function, target market, service level, other amenities, and industry standards.</p>	<p><u>BCcampus Open Publishing</u></p>	<p><a href="https://opentextbc.ca/introtourism2e/chapter/hotels/">https://opentextbc.ca/introtourism2e/chapter/hotels/</a></p>
<p>The modern hospitality landscape demands a shift towards prioritizing guest well-being in its entirety. This necessitates investments in upgraded spa facilities, state-of-the-art gyms, and comprehensive relaxation services.</p> <p>Recognizing the growing importance of mental health, hotels are incorporating programs like yoga and personalized training, while expanding menus with healthier, organic, and gluten-free options. Embracing this holistic approach fosters a guest experience that caters to both physical and mental rejuvenation, ultimately enhancing loyalty and solidifying a competitive edge within the industry.</p>	<p>Lightspeed blogs</p>	<p><a href="https://www.lightspeedhq.com/blog/top-10-hospitality-industry-trends-to-watch-in-2024/">https://www.lightspeedhq.com/blog/top-10-hospitality-industry-trends-to-watch-in-2024/</a></p>
<p>Serviced residences in the hospitality sector now prioritize modern home conveniences, offering essential appliances and kitchenware on demand to ensure a homelike experience. Wellness treatments, particularly customized therapies by providers like Kaya Kalp, are also becoming a standard offering, focusing on personalized guest rejuvenation and health.</p>	<p>ITC Kolkata</p>	<p><a href="https://www.itchotels.com/en/itcroyalbengalkolkata">https://www.itchotels.com/en/itcroyalbengalkolkata</a></p>

## Summary of the research

The vast hospitality industry encompasses a wide range of businesses, including theme parks, cruise lines, hotels, restaurants, bars, cafes, resorts, casinos, and event organizing. Basically, any company that provides services intending to meet clients' wants and preferences.

We have limited our research for the group project to the hotel business. It is evident from looking at different websites that this sector provides a broad range of services under one roof to meet the needs of different types of guests. The hotel offers a wide range of amenities, such as different room types, restaurants serving international cuisine, themed bars, a lively nightlife, a comprehensive spa, fitness center, and wellness center, meeting and conference spaces, event planning services, a business center, lounges, an outdoor pool, and many other conveniences like laundry, housekeeping, and gift shop. Furthermore, the hotel's provision of complimentary WIFI, service requests, and city adventure trips underscores its dedication to delivering a comprehensive and fulfilling guest experience.

Over time, the emphasis has shifted more toward what customers want, changes in work and life, longer lifespans, increased computer use, increased emphasis on health and fitness (including organic food and spa treatments), the value of being environmentally conscious and healthy, the desire for natural experiences, an increased need for adventure and fun, and various justifications for traveling due to significant events. Companies adapt to these shifts by providing services above and beyond what customers request. Hotels often enter strategic alliances with various stakeholders, including other hotels, travel agencies, airlines, and event management companies.

Additionally, we learned that a hotel's human resources strategy must include hotel employment portals to draw in and hire the talent it requires. A hotel can list the roles it is hiring for on these career portals, along with all the information a prospective candidate needs to know about the position—including what qualifications and abilities are required and how to apply. The idea is that just as a hotel can obtain the best applicant for a certain vacancy, job seekers can grasp what kinds of roles are available that fit their talents when they land on a career website for hotels.

Typically, a hotel career website would provide details on roles like Bar attendants, Security personnel, Chefs, Singer.

A hotel's marketing strategy includes blogging and newsletters since they provide channels for connecting with prospective customers, building brand awareness, and encouraging reservations. Hotels may become recognized by industry experts by writing interesting and educational blog posts. This draws readers in naturally through SEO and provides them useful advice on where to go and what to see in the area. Newsletters function as direct lines of communication with previous and prospective visitors, giving hotels the opportunity to provide exclusive content, advertise discounts, and cultivate a sense of community and loyalty. When used in tandem, these technologies help hotels stay at the forefront of guests' minds, build stronger bonds with them, and eventually increase revenue in the cutthroat hospitality sector.

Following our research we have finalized the entities below:

1. Guest
2. Employee
3. Room
4. Restaurant & Bar
5. Service (spa, tour, transport, business centre)
6. Amenity (laundry, fitness centre, conference room, Game Zone, pool, housekeeping, parking, lounge, WIFI, etc.)
7. Newsletter
8. Reservation
9. PaymentMethod
10. Feedback
11. Social Platform
12. Careers
13. Offers/Promotion
14. Inventory
15. Transaction
16. Departments
17. Frequent/Loyal Customers
18. Product
19. Invoice
20. Travel Agent
21. Event
22. Supplier
23. Incident
24. Partners
25. Hotel

Using the above entities we drafted our first ERD [1].



## Design Issues in Initial ERD

### Fan Trap

We discovered a fan trap in our initial Entity-Relationship Diagram (ERD) involving the entities Guest, Reservation, Room, and Service (like spa, tour, transport, business center), based on the following assumptions:

A Guest can make multiple Reservations. Each reservation has a single room, but a single room can be reserved for more than one time over time. A guest may, during the period of his reservation, consume more than one service, and a service may be consumed by one or more guests.

This is where the Fan Trap occurs [3]. The entity "Guest" has a direct relationship with the entities "Reservation" and "Service". The entity "Reservation" has a relationship with the entity "Room".

The Fan Trap arises because there is some ambiguity in the model about how "Services" are related to a specific "Reservation" or "Room" of a "Guest". It can even give misleading answers when trying to answer questions like "Which services did a guest use for a specific reservation?" Since there is a direct relationship between Guest and Service, not taking into consideration that it was associated with a Reservation or a Room. Let's imagine this setup:

**Guest --(makes)--> Reservation --(is for) --> Room    Guest --(utilizes)--> Service**

This creates a scenario where it is not clear how Service directly relates to each Guest's Reservation or Room, something that makes it unclear in case Service relates with Reservation or Room, hence again giving room for ambiguous information in queries related to services used per reservation.

This creates a scenario where it's unclear how services relate directly to each guest's reservation or room because there's no direct path from `Reservation` or `Room` to `Service`, creating potential confusion in queries related to services used per reservation [6].

### Solving the Fan Trap:

This problem is overcome by changing the relationship paths. The connection between entities will have to reflect their real-world participation. For example:

From 'Service' directly to 'Reservation', instead of 'Guest' Should be 'Reservation' connecting to 'Guest' in the first place

Creating a new entity 'ReservationServices' which establishes a relationship between 'Reservation' and 'Service' to say which services were used for which reservation.

## Time Variant Design Issue

This issue occurs when one of the values of attributes of an entity keeps on changing over time. So, to solve this problem, we have created another entity which is in relationship of 1:M with the original entity. The new entity which is created contains the new value and any other related attributes to that entity.

### Solving the Time Variant Design Issue:

**Reservation** - For solving the time variant design issue in Reservation entity, we created another entity named **ReservationHistory** which has **RHistoryID** as primary key and **ReservationID** as the foreign key. The other attributes of the entity are **TimeStamp** and **Status**. There is one to many relationships between Reservation and ReservationHistory entity.

**Offers and promotion** - For solving the time variant design issue in Offers/Promotion entity, we created another entity named **OfferHistory** which has **OfferHistoryID** as primary key and **OfferID** as the foreign key. The other attributes of the entity are **ChangeReason**, **StartDate** and **EndDate**. There is one to many relationships between Offer/Promotion and OfferHistory entity.

**Incident** - For solving the time variant design issue in Incident entity, we created another entity named **IncidentHistory** which has **IncidentHistoryID** as primary key and **IncidentID** as the foreign key. The other attributes of the entity are **TimeStamp** and **Status**. There is one to many relationships between Incident and IncidentHistory entity.

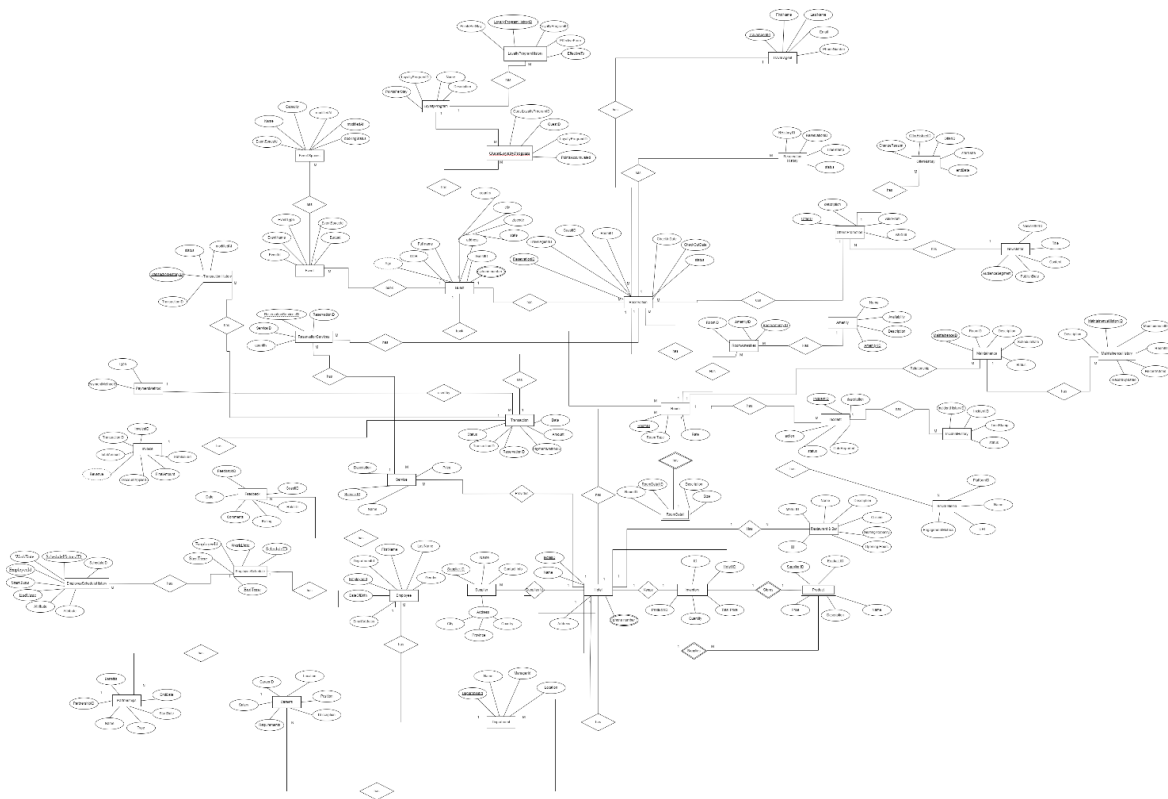
**Transaction** - For solving the time variant design issue in Transaction entity, we created another entity named **TransactionHistory** which has **TransactionHistoryID** as primary key and **TransactionID** as the foreign key. The other attributes of the entity are **Status** and **ModifiedAt**. There is one to many relationships between Transaction and TransactionHistory entity.

## Final Entity Relationship Diagram

**Note:** We have identified new entities in the later stages of our project.

After overcoming all design issues mentioned above, we have created issue free ERD as below.

This ERD also contains the newly added entities which also have all the design issues resolved.



**Fig. 2:** Final ERD [5].

## Logical Phase:

### Partial Dependency

- Upon analysis, it appears that all non-key attributes, such as `Description` and `Size`, exhibit full functional dependence on the entire composite key rather than on any of its individual components. For instance, in our database schema, the `RoomDetail` entity contains composite primary keys consisting of `RoomDetailID` and `RoomID`. This confirms that there are no partial dependencies within the RoomDetail table's structure, ensuring adherence to the principles of normalization in database design [7].

### Transitive Dependency

- A transitive dependency in a database occurs when a non-prime attribute is dependent on another non-prime attribute, rather than being directly dependent on the primary key. This situation violates the Third Normal Form (3NF).
- In the provided database schema, foreign keys establish direct relationships between tables, such as `Reservation` to `Guest`, `Room`, `travel\_agent`, and `OffersPromotion`. This setup ensures that the non-key attributes in each table are dependent on the primary key of their respective table, and not on other non-key attributes within the same table. Consequently, this structure adheres to the Third Normal Form (3NF) by preventing transitive dependencies, where a non-prime attribute is wrongly dependent on another non-prime attribute instead of the primary key.
- For example, in the `Reservation` table, attributes like `check\_in\_date`, `check\_out\_date`, and `status` depend on the `reservation\_id` (primary key) and not on other non-prime attributes. Foreign key dependencies on other tables (such as `Guest`, `Room`, etc.) also don't introduce transitive dependencies because they are linking primary keys of related tables to foreign keys, which is a normalization practice designed to eliminate redundancy and maintain data integrity, not a sign of a transitive dependency within the table itself.

### Conclusion

- The justification for the claim that there are no partial or transitive dependencies in the schema lies in the observation that all attributes within each table are functionally dependent on the primary key of their own table (satisfying 2NF) and there are no dependencies of non-prime attributes on other non-prime attributes (satisfying 3NF). We prioritized ensuring data consistency and ease of management by adhering to 2NF and 3NF principles. We structured each table so that all attributes depend solely on the primary key, preventing partial dependencies, and ensured no non-key attribute depends on another non-key attribute, maintaining 3NF. This careful approach was aimed at avoiding data anomalies and facilitating a clear, manageable database structure.



## **Distributed Database:**

We have analysed that our system can be fragmented on database level.

### **Database 1: Hotel and Guest Management**

This database will focus on the core operations related to hotel management, guest interactions, room bookings, and on-premises services and amenities.

#### **Core Hotel Operations**

EventSpaces

Guest

reservation (Reservation)

OffersPromotion

Newsletter

ReservationServices

Employee

Room

RoomDetail

Departments

Event

Incident

Feedback

Careers

ReservationHistory

IncidentHistory

RoomAmenities

EmployeeSchedule

EmployeeScheduleHistory

LoyaltyProgram

LoyaltyProgramHistory

GuestLoyaltyProgram

Maintenance

MaintenanceHistory

DiningReservation

RestaurantBar

Service

Amenity

## Supporting Tables

Address (as it is central to both Hotel and Guest, it's crucial to have it here to avoid cross-database joins)

## Database 2: Business Operations and External Relations

This database focuses on supply chain management, financial transactions, marketing promotions, career opportunities, and external partnerships.

Inventory

Product

Supplier

hotel (Hotel)

travel\_agent (Travel Agent)

Payment\_Method

Transaction

Invoice

TransactionHistory

OfferHistory

Partnerships

Social\_Media

## Key Considerations:

- **Reservation and Offer Promotions:** The Reservation table includes a foreign key to OffersPromotion, which may suggest a cross-database relationship. However, this link is primarily for tracking which promotions are applied to reservations. Depending on our application logic, we might handle this at the application level to minimize direct database cross-references or replicate necessary promotion details into the Hotel and Guest Management database for local reference during reservation processing.
- **Cross-database Relations:** For functionalities like linking reservations with promotions (from the Business Operations database) or managing inventory that affects both supply chain and hotel operations, application-level logic might be used to bridge data from both databases as needed, rather than direct database links.
- **Address Table Replication:** If the Address information is frequently accessed and updated from entities in both databases, consider replicating this table or using a microservice architecture for address management to reduce cross-database dependencies.
- This approach aims to segregate operational aspects (direct hotel and guest management) from broader business operations (supply chain, financials, marketing, and partnerships).

## **DDL Statements for the tables in two fragments :**

### **Hotel Guest Management -**

#### **-- DDL to create Address table**

```
CREATE TABLE Address (  
    AddressID INT PRIMARY KEY,  
    City VARCHAR(100),  
    ZipCode VARCHAR(20),  
    Country VARCHAR(100),  
    State VARCHAR(100)  
);
```

#### **-- DDL to create EventSpaces table**

```
CREATE TABLE EventSpaces (  
    EventSpaceId INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Capacity INT,  
    modifiedAt DATETIME,  
    bookingStatus VARCHAR(50)  
);
```

#### **-- DDL to create Inventory table**

```
CREATE TABLE Inventory (  
    InventoryID INT PRIMARY KEY,  
    HotelID INT,  
    ProductID INT,  
    Quantity INT,  
    TotalPrice DECIMAL(10, 2),  
    LastUpdated DATETIME,
```



```
FOREIGN KEY (HotelID) REFERENCES Hotel(hotel_id),  
FOREIGN KEY (ProductID) REFERENCES Product(ProductID)  
);
```

**-- DDL to create Product table**

```
CREATE TABLE Product (  
    ProductID INT PRIMARY KEY,  
    SupplierID INT,  
    Name VARCHAR(255),  
    Description TEXT,  
    Price DECIMAL(10, 2),  
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)  
);
```

**-- DDL to create Supplier table**

```
CREATE TABLE Supplier (  
    SupplierID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    ContactInfo VARCHAR(255),  
    AddressID INT,  
    FOREIGN KEY (AddressID) REFERENCES Address(AddressID)  
);
```

**-- DDL to create Address table**

```
CREATE TABLE Address (  
    AddressID INT PRIMARY KEY,  
    City VARCHAR(100),  
    ZipCode VARCHAR(20),  
    Country VARCHAR(100),  
    State VARCHAR(100)
```

);

**-- DDL to create Service table**

```
CREATE TABLE Service (  
    ServiceID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Description TEXT,  
    Price DECIMAL(10, 2)  
);
```

**-- DDL to create Amenity table**

```
CREATE TABLE Amenity (  
    AmenityID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Description TEXT,  
    Availability BOOLEAN,  
    Price DECIMAL(10, 2)  
);
```

**-- DDL to create RestaurantBar table**

```
CREATE TABLE RestaurantBar (  
    RestaurantID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Description TEXT,  
    Cuisine VARCHAR(255),  
    SeatingCapacity INT,  
    OpeningHours VARCHAR(255)  
);
```

**-- DDL to create hotel table**

```
CREATE TABLE hotel (  
    hotel_id INT PRIMARY KEY,  
    name VARCHAR(255),  
    phone_number VARCHAR(20),  
    AddressID INT,  
    FOREIGN KEY (AddressID) REFERENCES Address(AddressID)  
);
```

**-- DDL to create travel\_agent table**

```
CREATE TABLE travel_agent (  
    travel_agent_id INT PRIMARY KEY,  
    first_name VARCHAR(255),  
    last_name VARCHAR(255),  
    email VARCHAR(255),  
    phone_number VARCHAR(20)  
);
```

**-- DDL to create Guest table**

```
CREATE TABLE Guest (  
    Age INT,  
    DOB DATE,  
    FullName VARCHAR(255),  
    GuestID INT PRIMARY KEY,  
    PhoneNumber VARCHAR(20),  
    AddressID INT,  
    FOREIGN KEY (AddressID) REFERENCES Address(AddressID)  
);
```

**-- DDL to create reservation table**

```
CREATE TABLE reservation (  
    reservation_id INT PRIMARY KEY,  
    travel_agent_id INT,  
    guest_id INT,  
    room_id INT,  
    Offer_id INT,  
    check_in_date DATE,  
    check_out_date DATE,  
    status VARCHAR(50),  
    FOREIGN KEY (travel_agent_id) REFERENCES travel_agent(travel_agent_id),  
    FOREIGN KEY (guest_id) REFERENCES Guest(GuestID),  
    FOREIGN KEY (room_id) REFERENCES Room(RoomID),  
    FOREIGN KEY (offer_id) REFERENCES OffersPromotion(OfferID)  
);
```

**-- DDL to create OffersPromotion table**

```
CREATE TABLE OffersPromotion (  
    OfferID INT PRIMARY KEY,  
    Description TEXT,  
    ValidFrom DATE,  
    ValidTill DATE,  
    NewsletterID INT,  
    FOREIGN KEY (NewsletterID) REFERENCES Newsletter(NewsletterID)  
);
```

**-- DDL to create Newsletter table**

```
CREATE TABLE Newsletter (  
    NewsletterID INT PRIMARY KEY,  
    Title VARCHAR(255),
```

```
Content TEXT,  
PublishedDate DATE,  
AudienceSegment VARCHAR(255)  
);
```

**-- DDL to create ReservationServices table**

```
CREATE TABLE ReservationServices (  
    ReservationServicesID INT PRIMARY KEY,  
    ReservationID INT,  
    ServiceID INT,  
    Quantity INT,  
    FOREIGN KEY (ReservationID) REFERENCES reservation(reservation_id),  
    FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID)  
);
```

**-- DDL to create Employee table**

```
CREATE TABLE Employee (  
    DateOfBirth DATE,  
    EmployeeId INT PRIMARY KEY,  
    DepartmentId INT,  
    FirstName VARCHAR(255),  
    LastName VARCHAR(255),  
    Gender CHAR(1),  
    FOREIGN KEY (DepartmentId) REFERENCES Departments(DepartmentId)  
);
```

**-- DDL to create Room table**

```
CREATE TABLE Room (  
    RoomID INT PRIMARY KEY,  
    RoomType VARCHAR(255),
```

```
Rate DECIMAL(10, 2)
);
```

**-- DDL to create RoomDetail table**

```
CREATE TABLE RoomDetail (
    RoomDetailID INT,
    RoomID INT,
    Description TEXT,
    Size DECIMAL(10, 2),
    PRIMARY KEY (RoomDetailID, RoomID),
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID)
);
```

**-- DDL to create Departments table**

```
CREATE TABLE Departments (
    DepartmentId INT PRIMARY KEY,
    Name VARCHAR(255),
    ManagerId INT,
    Location VARCHAR(255)
);
```

**-- DDL to create Event table**

```
CREATE TABLE Event (
    EventId INT PRIMARY KEY,
    EventName VARCHAR(255),
    EventType VARCHAR(255),
    EventSpaceId INT,
    Budget DECIMAL(10, 2),
    FOREIGN KEY (EventSpaceId) REFERENCES EventSpaces(EventSpaceId)
);
```

**-- DDL to create Incident table**

```
CREATE TABLE Incident (  
    IncidentID INT PRIMARY KEY,  
    Description TEXT,  
    Action TEXT,  
    Status VARCHAR(255),  
    DateReported DATE  
);
```

**-- DDL to create Feedback table**

```
CREATE TABLE Feedback (  
    FeedbackID INT PRIMARY KEY,  
    Date DATE,  
    Comments TEXT,  
    Rating INT,  
    HotelID INT,  
    GuestID INT  
);
```

**-- DDL to create Careers table**

```
CREATE TABLE Careers (  
    CareerID INT PRIMARY KEY,  
    Position VARCHAR(255),  
    Requirements TEXT,  
    Description TEXT,  
    Salary DECIMAL(10, 2),  
    Location VARCHAR(255)  
);
```

**-- DDL to create ReservationHistory table**

```
CREATE TABLE ReservationHistory (  
    RHistoryID INT PRIMARY KEY,  
    ReservationID INT,  
    TimeStamp TIMESTAMP,  
    status VARCHAR(50),  
    FOREIGN KEY (ReservationID) REFERENCES reservation(reservation_id)  
);
```

**-- DDL to create IncidentHistory table**

```
CREATE TABLE IncidentHistory (  
    IncidentHistoryID INT PRIMARY KEY,  
    IncidentID INT,  
    TimeStamp TIMESTAMP,  
    status VARCHAR(255),  
    FOREIGN KEY (IncidentID) REFERENCES Incident(IncidentID)  
);
```

**-- DDL to create RoomAmenities table**

```
CREATE TABLE RoomAmenities (  
    RoomAmenitiesID INT PRIMARY KEY,  
    RoomID INT,  
    AmenityID INT,  
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID),  
    FOREIGN KEY (AmenityID) REFERENCES Amenity(AmenityID)  
);
```

**-- DDL to create EmployeeSchedule table**

```
CREATE TABLE EmployeeSchedule (  

```



```
ScheduleID INT PRIMARY KEY,  
EmployeeId INT,  
WorkDate DATE,  
StartTime TIME,  
EndTime TIME,  
FOREIGN KEY (EmployeeId) REFERENCES Employee(EmployeeId)  
);
```

**-- DDL to create EmployeeScheduleHistory table**

```
CREATE TABLE EmployeeScheduleHistory (  
    ScheduleHistoryID INT PRIMARY KEY AUTO_INCREMENT,  
    ScheduleID INT,  
    EmployeeId INT,  
    WorkDate DATE,  
    StartTime TIME,  
    EndTime TIME,  
    EffectiveDate DATETIME,  
    EndDate DATETIME DEFAULT NULL,  
    FOREIGN KEY (EmployeeId) REFERENCES Employee(EmployeeId),  
    FOREIGN KEY (ScheduleID) REFERENCES EmployeeSchedule(ScheduleID)  
);
```

**-- DDL to create LoyaltyProgram table**

```
CREATE TABLE LoyaltyProgram (  
    LoyaltyProgramID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Description TEXT,  
    PointsPerStay INT  
);
```

**-- DDL to create LoyaltyProgramHistory table**

```
CREATE TABLE LoyaltyProgramHistory (  
    LPHistoryID INT PRIMARY KEY AUTO_INCREMENT,  
    LoyaltyProgramID INT,  
    Name VARCHAR(255),  
    Description TEXT,  
    PointsPerStay INT,  
    EffectiveFrom DATETIME,  
    EffectiveTo DATETIME DEFAULT NULL,  
    FOREIGN KEY (LoyaltyProgramID) REFERENCES LoyaltyProgram(LoyaltyProgramID)  
);
```

**-- DDL to create GuestLoyaltyProgram table**

```
CREATE TABLE GuestLoyaltyProgram (  
    GLPID INT PRIMARY KEY,  
    GuestID INT,  
    LoyaltyProgramID INT,  
    PointsAccumulated INT,  
    FOREIGN KEY (GuestID) REFERENCES Guest(GuestID),  
    FOREIGN KEY (LoyaltyProgramID) REFERENCES LoyaltyProgram(LoyaltyProgramID)  
);
```

**-- DDL to create Maintenance table**

```
CREATE TABLE Maintenance (  
    MaintenanceID INT PRIMARY KEY,  
    RoomID INT,  
    Description TEXT,  
    ScheduledDate DATE,  
    Status VARCHAR(50),  
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID)
```

);

**-- DDL to create MaintenanceHistory table**

```
CREATE TABLE MaintenanceHistory (  
    MaintenanceHistoryID INT PRIMARY KEY AUTO_INCREMENT,  
    MaintenanceID INT,  
    RoomID INT,  
    Description TEXT,  
    ScheduledDate DATE,  
    Status VARCHAR(50),  
    RecordAdded DATETIME,  
    RecordUpdated DATETIME DEFAULT NULL,  
    FOREIGN KEY (MaintenanceID) REFERENCES Maintenance(MaintenanceID),  
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID)  
);
```

**-- DDL to create DiningReservation table**

```
CREATE TABLE DiningReservation (  
    ReservationID INT PRIMARY KEY,  
    GuestID INT,  
    RestaurantID INT,  
    ReservationDate DATE,  
    ReservationTime TIME,  
    NumberOfGuests INT,  
    FOREIGN KEY (GuestID) REFERENCES Guest(GuestID),  
    FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID)  
);
```

## **Business Operation fragment DDLs**

### **-- DDL to create Address table**

```
CREATE TABLE Address (  
    AddressID INT PRIMARY KEY,  
    City VARCHAR(100),  
    ZipCode VARCHAR(20),  
    Country VARCHAR(100),  
    State VARCHAR(100)  
);
```

### **-- DDL to create Supplier table**

```
CREATE TABLE Supplier (  
    SupplierID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    ContactInfo VARCHAR(255),  
    AddressID INT,  
    FOREIGN KEY (AddressID) REFERENCES Address(AddressID)  
);
```

### **-- DDL to create Product table**

```
CREATE TABLE Product (  
    ProductID INT PRIMARY KEY,  
    SupplierID INT,  
    Name VARCHAR(255),  
    Description TEXT,  
    Price DECIMAL(10, 2),  
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
```

);

**-- DDL to create Hotel table**

```
CREATE TABLE Hotel (  
    hotel_id INT PRIMARY KEY,  
    name VARCHAR(255),  
    phone_number VARCHAR(20),  
    AddressID INT,  
    FOREIGN KEY (AddressID) REFERENCES Address(AddressID)  
);
```

**-- DDL to create Inventory table**

```
CREATE TABLE Inventory (  
    InventoryID INT PRIMARY KEY,  
    HotelID INT,  
    ProductID INT,  
    Quantity INT,  
    TotalPrice DECIMAL(10, 2),  
    LastUpdated DATETIME,  
    FOREIGN KEY (HotelID) REFERENCES Hotel(hotel_id),  
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)  
);
```

**-- DDL to create Payment Method table**

```
CREATE TABLE Payment_Method (  
    payment_method_id INT PRIMARY KEY,  
    type VARCHAR(50)  
);
```

**-- DDL to create Reservation table**

```
CREATE TABLE Reservation (  
    reservation_id INT PRIMARY KEY,  
    check_in_date DATE,  
    check_out_date DATE,  
    status VARCHAR(50)  
);
```

**-- DDL to create Transaction table**

```
CREATE TABLE Transaction (  
    transaction_id INT PRIMARY KEY,  
    status VARCHAR(50),  
    reservation_id INT,  
    payment_method_id INT,  
    amount DECIMAL(10,2),  
    date DATE,  
    FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id),  
    FOREIGN KEY (payment_method_id) REFERENCES  
Payment_Method(payment_method_id)  
);
```

**-- DDL to create Invoice table**

```
CREATE TABLE Invoice (  
    invoice_id INT PRIMARY KEY,  
    transaction_id INT,  
    total_amount DECIMAL(10,2),  
    revenue DECIMAL(10,2),  
    discount_applied DECIMAL(10,2),  
    final_amount DECIMAL(10,2),
```

```
date_issued DATE,  
FOREIGN KEY (transaction_id) REFERENCES Transaction(transaction_id)  
);
```

**-- DDL to create TransactionHistory table**

```
CREATE TABLE TransactionHistory (  
TransactionHistoryID INT PRIMARY KEY,  
TransactionID INT,  
status VARCHAR(50),  
modifiedAt TIMESTAMP,  
FOREIGN KEY (TransactionID) REFERENCES Transaction(transaction_id)  
);
```

**-- DDL to create OffersPromotion table**

```
CREATE TABLE OffersPromotion (  
OfferID INT PRIMARY KEY,  
Description TEXT,  
ValidFrom DATE,  
ValidTill DATE,  
NewsletterID INT,  
FOREIGN KEY (NewsletterID) REFERENCES Newsletter(NewsletterID)  
);
```

**-- DDL to create OfferHistory table**

```
CREATE TABLE OfferHistory (  
OfferHistoryID INT PRIMARY KEY,  
OfferID INT,  
ChangeReason TEXT,
```

```
startDate DATE,  
endDate DATE,  
FOREIGN KEY (OfferID) REFERENCES OffersPromotion(OfferID)  
);
```

**-- DDL to create Careers table**

```
CREATE TABLE Careers (  
    CareerID INT PRIMARY KEY,  
    Position VARCHAR(255),  
    Requirements TEXT,  
    Description TEXT,  
    Salary DECIMAL(10, 2),  
    Location VARCHAR(255)  
);
```

**-- DDL to create Partnerships table**

```
CREATE TABLE Partnerships (  
    PartnershipID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Type VARCHAR(255),  
    Benefits TEXT,  
    StartDate DATE,  
    EndDate DATE  
);
```

**-- DDL to create Newsletter table**

```
CREATE TABLE Newsletter (  
    NewsletterID INT PRIMARY KEY,
```



```
Title VARCHAR(255),  
Content TEXT,  
PublishedDate DATE,  
AudienceSegment VARCHAR(255)  
);
```

**-- DDL to create Social Media table**

```
CREATE TABLE Social_Media (  
    PlatformID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    URL VARCHAR(255),  
    EngagementMetrics TEXT  
);
```

## Structure of Global Data Catalog

A Global Data Catalog (GDC) plays a crucial role in managing and facilitating access to data distributed across various locations. The GDC serves as a centralized repository that contains detailed metadata about the data stored in the system, helping to optimize queries and maintain consistency across the network.

To effectively organize and retrieve this metadata, our GDC is structured with various columns, each designed to hold specific information about the data entities. Below is an outline of the columns included in the Global Data Catalog structure, which together provide a comprehensive overview of the data's characteristics and location within the distributed database system.

**CatalogID:** This column is of type INT and is set to AUTO\_INCREMENT, meaning it will automatically increment its value for each new row added. It is designated as the primary key, ensuring each entry has a unique identifier [\[2\]](#).

**SiteName:** This column is of type VARCHAR(50), meaning it can hold up to 50 characters. It stores the name of the site associated with the data.

**SiteIP:** This column is of type VARCHAR(15), meaning it can hold up to 15 characters. It stores the IP address of the site associated with the data.

**TableName:** This column is of type VARCHAR(100), meaning it can hold up to 100 characters. It stores the name of the table where the data is stored.

**Description:** This column is of type TEXT, allowing for longer text entries. It stores a description of the data.

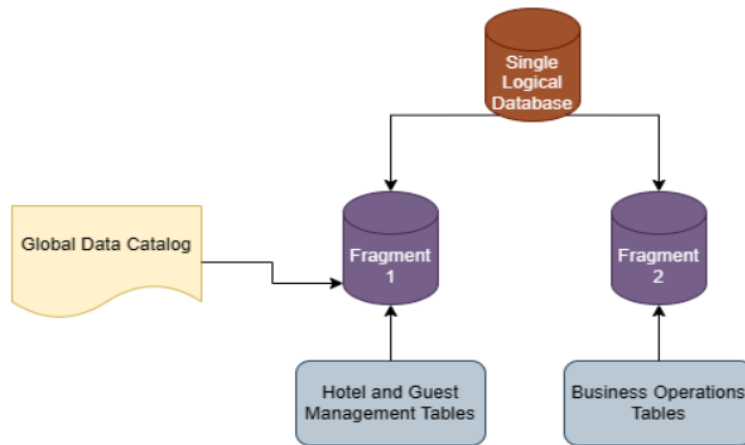
**DataSize:** This column is of type VARCHAR(50), meaning it can hold up to 50 characters. It stores the size of the data.

**LastUpdated:** This column is of type DATE. It stores the date when the data was last updated.

**DataType:** This column is of type VARCHAR(50), meaning it can hold up to 50 characters. It stores the type of data.

**Constraints:** This column is of type TEXT, allowing for longer text entries. It stores any constraints or additional information related to the data.

## Structure of Distributed Database



*Fig. 4:* Database Fragmentation Architecture

- Given that our project does not include multiple hotel chains, we opted not to implement table-level fragmentation. Instead, we selected **Database fragmentation** to divide our system into specialized segments. This structure allows one database to swiftly manage guest bookings, while another separately administers business operations, such as inventory and finances. This targeted approach enhances performance, streamlines system maintenance, and ensures that intensive tasks in one segment do not hinder the efficiency of another.
- We have taken **two MySQL instances** [\[4\]](#) on GCP and divided our entire database into two fragments which focus on a single aspect of the hotel. First Fragment contains all the tables related to **Hotel and Guest Management** and second Fragment contains all the tables related to **Business Operations and Relations**.
- We have placed our **GDC (Global Data Catalog) table**, in Fragment no. 1.

## Novelty

### Global Data Catalog implementation through Java Program -

We have written a java code to check for the server details for respective table that has been queried by the user. Based on the server details, we connect with that database to run the user query. It is transparent as the user does not know about these fragmentation details.

### GitLab URL -

<https://git.cs.dal.ca/himanshi/csci5408-sprint4>

### Code Explanation -

Here's a breakdown of what the code in our Java application that interacts with databases based on user input queries does:

**Sprint2 (Main Class):** This is the main class of your application. It prompts the user to enter a database query, extracts the table name from the query using QueryParser, and then fetches the site name/IP from a local database using SiteIpFetcher. Based on the fetched site name, it determines which remote database to connect to and executes the user query using DatabaseExecutor.

**DatabaseExecutor:** This class is responsible for executing SQL queries on a remote database. It takes the remote database URL, user query, and user details (presumably username and password) as input. It connects to the remote database using the provided details, executes the query, and prints the result set or the number of updated rows.

**QueryParser:** This class extracts the table name from a given SQL query. It uses a regular expression to find the table name after keywords like FROM, UPDATE, or INTO.

**SiteIpFetcher:** This class fetches the site IP from a local database based on the table name. It takes the local database URL, table name, and user details as input. It executes a SQL query to fetch the site IP from the database based on the provided table name.

## Code execution snapshots

After we run the Sprint2.class, user enters the query - “select \* from Product”, parser prints the table name and further the GDC table is hit to fetch the server details. Connection is established with that database server and query is executed and stored in resultset.

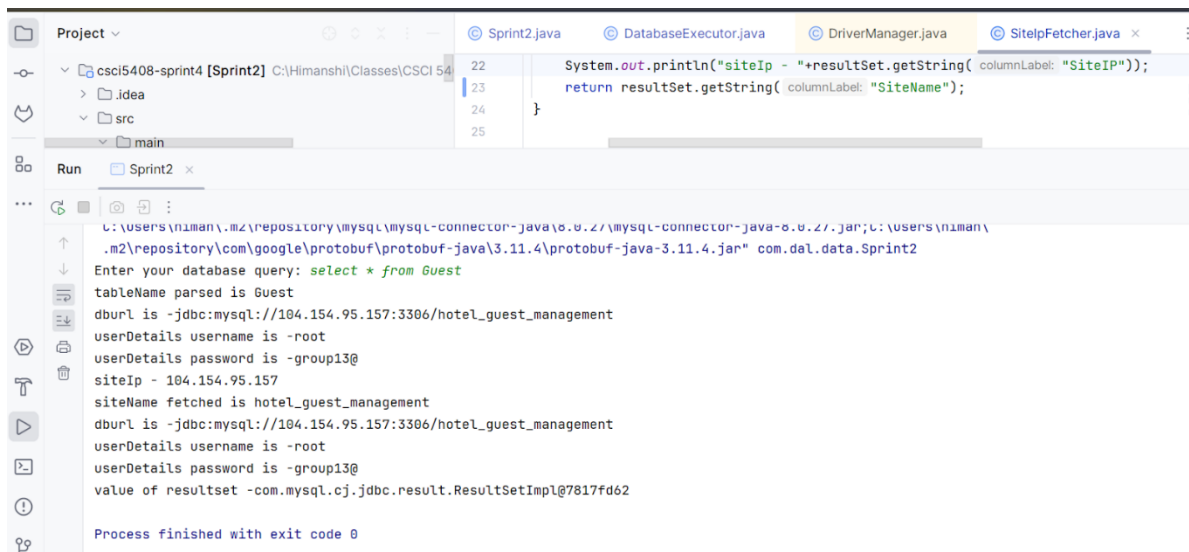


```
Run Sprint2 x
C:\Users\himan\.jdk\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3\lib\idea_rt.jar=55706:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Himanshi\Classes\CSCI 5408\Group Project\csci5408-sprint4\target\classes;C:\Users\himan\.m2\repository\mysql\mysql-connector-java\8.0.27\mysql-connector-java-8.0.27.jar;C:\Users\himan\.m2\repository\com\google\protobuf\protobuf-java\3.11.4\protobuf-java-3.11.4.jar" com.dal.data.Sprint2
Enter your database query: select * from Product
tableName parsed is Product
dburl is -jdbc:mysql://104.154.95.157:3306/hotel_guest_management
userDetails username is -root
userDetails password is -group13@
siteIp - 34.123.28.213
siteName fetched is business_operations
dburl is -jdbc:mysql://34.123.28.213:3306/business_operations
userDetails username is -root
userDetails password is -group13@
value of resultSet -com.mysql.cj.jdbc.result.ResultSetImpl@7817fd62

Process finished with exit code 0
```

**Fig. 5:** Database details fetched for Product table

Similarly, for Guest table different Ip as per its location is fetched and connection is established.



```
Project v
csci5408-sprint4 [Sprint2] C:\Himanshi\Classes\CSCI 5408\Group Project\csci5408-sprint4\target\classes;C:\Users\himan\.m2\repository\mysql\mysql-connector-java\8.0.27\mysql-connector-java-8.0.27.jar;C:\Users\himan\.m2\repository\com\google\protobuf\protobuf-java\3.11.4\protobuf-java-3.11.4.jar" com.dal.data.Sprint2
Enter your database query: select * from Guest
tableName parsed is Guest
dburl is -jdbc:mysql://104.154.95.157:3306/hotel_guest_management
userDetails username is -root
userDetails password is -group13@
siteIp - 104.154.95.157
siteName fetched is hotel_guest_management
dburl is -jdbc:mysql://104.154.95.157:3306/hotel_guest_management
userDetails username is -root
userDetails password is -group13@
value of resultSet -com.mysql.cj.jdbc.result.ResultSetImpl@7817fd62

Process finished with exit code 0
```

**Fig. 6:** Database details fetched for Guest table

## **What Changes we have done in Final ERD after meeting with HeadTA:**

- We have added a few more entities to our Project, as suggested by HeadTA and now our **Total entities count is 41.**
- After adding new Entities, 3 new Time variant design issues come up, we have also solved that.
- 3 new Time variant issues solved (**EmployeeScheduleHistory, MaintainenceHistory, LoyalProgramHistory**).

### **The Final set of Entities are:**

1. Address
2. EventSpaces
3. Inventory
4. Product
5. Supplier
6. Service
7. Amenity
8. RestaurantBar
9. hotel
10. travel\_agent
11. Guest
12. reservation
13. OffersPromotion
14. Newsletter
15. ReservationServices
16. Employee
17. Room
18. RoomDetail
19. Departments
20. Event
21. Incident
22. Feedback
23. Careers
24. ReservationHistory
25. IncidentHistory
26. RoomAmenities
27. EmployeeSchedule
28. EmployeeScheduleHistory
29. LoyaltyProgram

- 30. LoyaltyProgramHistory
- 31. GuestLoyaltyProgram
- 32. Maintenance
- 33. MaintenanceHistory
- 34. DiningReservation
- 35. Payment\_Method
- 36. Transaction
- 37. Invoice
- 38. TransactionHistory
- 39. OfferHistory
- 40. Partnerships
- 41. Social\_Media

## References

- [1] G., "Guide to Entity-Relationship Diagram Notations & Symbols," Gleek, 02-Nov-2021. [Online]. Available: <https://www.gleek.io/blog/er-symbols-notations> [Accessed: Feb. 15, 2024].
- [2] "Types of Keys in Relational Model (Candidate, Super, Primary, Alternate, and Foreign)," *GeeksforGeeks*. [Online]. Available: <https://www.geeksforgeeks.org/types-ofkeys-in-relational-model-candidate-super-primary-alternate-and-foreign/> [Accessed: Feb. 15, 2024].
- [3] "Fan Traps and Chasm Traps," Qlik Community, 08-Sep-2015. [Online]. Available: <https://community.qlik.com/t5/Design/Fan-traps-and-Chasm-traps/ba-p/1463093> [Accessed: Feb. 20 2024]
- [4] "Create and start a VM instance," *Google Cloud* [Online]. Available: <https://cloud.google.com/compute/docs/instances/create-start-instance> [Accessed: Mar. 14, 2024]
- [5] "draw.io - free flowchart maker and diagrams online." [Online]. Available: <https://app.diagrams.net/>. [Accessed: Mar. 15, 2024]
- [6] "Chasm and Fan Traps." [Online]. Available: <https://docs.sisense.com/main/SisenseLinux/chasm-and-fan-traps.htm> [Accessed: Feb. 26, 2024]
- [7] "Partial Dependency in DBMS." [Online]. Available: <https://www.tutorialspoint.com/Partial-Dependency-in-DBMS> [Accessed: March 05, 2024]