MINI-WORLD

Team Outlaws: Kartik Garg(2019101060), Harshit Sharma(2019101083)

We have picked a mini world similar to zomato. We have 5 relations in this setting which are “Food_Item”, “Restaurants”, “User”, “Order” and “Order_Items”.

Note: Our mini world does not mimic zomato to its full extent. It is a minimal environment that tries to mimic zomato in every aspect posible.

1. User Entity has 4 attributes: ‘Name’, ’Email’, ‘Address’ and ‘Phone_Number’.
    a. All of them are intentionally kept to be single valued variables thereby reducing complexity of the project such that we can focus more on Distributed Database Concepts rather than Database Concepts.
2. Restaurant Entity has 6 attributes: ’Name’, ‘Address’, ‘Email’, ‘Rating’, ‘Specialty’ and ‘Num_Reviews’.
    a. Specialty: Specialty is a single word that will be given by the vendor. For instance one can give some cuisine, for example chinese, as its specialty.
    b. Again, all the variables are single valued.
3. Food_Item Entity has 5 attributes: ‘Name’, ‘Type’, ‘Price’, ‘Category’ and ‘FK_Restaurant’
    a. Type: This attribute’s domain is {Chinese, Indian, Italian} that is of size 3
    b. Category: This attribute’s domain is of size 2 that is {Veg, Non_Veg}
    c. FK_Restaurant: This attribute is a foreign key referencing some entry in Restaurant Relation.

Note: In each of the above relations we will add another variable named PK_Custom which will be a custom generated id for each entry in the table. This will serve the

purpose of the primary key for all the database tables. Basically similar to object_id in mongoDB.

4. Order Entity has 3 attributes: 'User_ID', 'Restaurant_ID', 'Amount'
    a. User_ID: A foreign Key referencing to some entry in User Relation
    b. Restaurant_ID: A foreign key referencing to some entry in Restaurant Relation
    c. Amount: Total amount of the transaction that was involved in this order

Note: Order Entity will also be given a PK_Custom

5. Order_Items Entity has 3 attributes: 'Order_ID', 'Item_ID' and 'Quantity'
    a. Order_ID: Referencing to some entity in Order Relation
    b. Item_ID: Referencing to some entity in Food_Item Relation

QUERY DETAILS

1. Fetch All Food Items: In the Food Items tab, one has to show all the food items that are available along with the restaurant name that serves that particular food item. [A SQL SELECT * statement will be generated, JOIN operation will also be involved]
    a. Further in this tab, one will have filters for selecting Type/Cuisine that is Chinese, Indian or Italian. [A SQL SELECTstatement with where clause will be generated here]
    b. In this tab, one will also have a filter to select the category that is Veg/Non-Veg. [A SQL query using AGGREGATE/GROUP_BY operators will be generated here]

2. Fetch All Restaurants: In the Restaurants tab, one has to show all the restaurants available along with some minimal details like avg rating, specialty and the name of the restaurant. [A SQL SELECT* statement with PROJECTION operator]
    a. In this tab, one can select any restaurant and further in that restaurant we will show all the food items for that particular restaurant [A SQL statement involving JOIN operation will be issued here]
    b. Filters that were mentioned in QUERY 1 will also be present here and similar kinds of queries will be generated here.

3. Fetch User: In User's Tab, one will show the minimal detail for the logged in user like his name and email address. [SQL SELECT statement with PROJECTION operator]
    a. In this tab, we will have a further option to 'View More Details and Edit', in this window one will be able to view all the details of the user and there will be an option to edit them. [SQL SELECT with JOIN and SQL UPDATE kind of Query]

4. All the above mentioned are the main queries that our application environment will require. There may be some more queries which we might consider while building our application in a more deeper and analytic manner but we assure that all the above mentioned queries will be present and will be handled appropriately.

5. Analytics Tab, one can view all the orders grouped by cuisine type.

FRAGMENTATION DETAILS

The Database Setting supports all the three kinds of Fragmentation, that is

1. **Horizontal Fragmentation**
    a. Food_Item relation is horizontally fragmented into 3 relations based on the type, that is Chinese/Indian/Italian. (one relation for each Type/Cuisine stored separately on one of the sites.)
    b. There was an idea to further fragment each of these relations into two relations based on Category, that is Veg/Non-Veg, but since our database is to support aggregate/group-by operator so we did not consider this fragmentation so as to illustrate Aggregate Operators.
    c. This fragmentation exists to aid the filter functionality of food items.
    d. Name of fragments are:
        i. Food_Item_Chinese
        ii. Food_Item_Indian
        iii. Food_Item_Italian
2. **Vertical Fragmentation**
    a. Restaurants relation is vertically fragmented into 2 relations
        i. One of the relation will store columns named as 'Name', 'Rating' and 'Specialty'
        ii. Second relation will store columns named as 'Address', 'Email' and 'Num_Reviews'
        iii. Both the fragments will also have PK_Custom as the primary key.

iv. This fragmentation exists to aid the UI design of Restaurants tab where we are supposed to show all the restaurants with minimal detail.
v. Name of Fragments are:
    1. Restaurants_Minimal
    2. Restaurants_Remaining

b. User Relation is fragmented into 2 relation
i. First one stores Name and Email, used to display minimal user details.
ii. Second one stores Address and Phone Number, will be used to display all the details in 'View More Details and Edit' Tab
iii. Name of Fragments are:
    1. User_Minimal
    2. User_Remaining

3. **Derived Horizontal Fragments**
a. Since we have HFs for food items we can derive the food items of a particular type/cuisine for a particular restaurant. This will let us implement the filter mentioned in 2.b of query details sections.
i. Derived Dynamically
b. Tables for Analytics tab will be derived horizontally using
i. Order_Items SEMIJOIN Food_Item_Chinese
ii. PROJECT Order_ID FROM a
iii. Order SEMIJOIN b
iv. We have the DHF which gives us all the orders of all users who had at least one item belonging to the chinese cuisine.
v. Let's name the fragments as:
    1. User_Restaurant_Order_Amount_Chinese

2. User_Restaurant_Order_Amount_Indian
3. User_Restaurant_Order_Amount_Italian

4. All the reasons behind choosing that specific fragmentation along with relating it with the query are mentioned alongside the fragmentation details.
5. One can more intuitively relate to these features while observing the actual zomato app.

ALLOCATION DETAILS

1. Food_Item_Chinese: Site 1
2. Food_Item_Indian: Site 2
3. Food_Item_Italian: Site 3
4. Restaurants_Minimal: Site 4
5. Restaurants_Remaining: Site 4
6. User_Minimal: Site 4
7. User_Remaining: Site 2
8. Order_Items: Site 2
9. User_Restaurant_Order_Amount_Chinese: Site 1
10. User_Restaurant_Order_Amount_Indian: Site 2
11. User_Restaurant_Order_Amount_Italian: Site 3

1,2,3 are randomly allotted, based on this allotment 9,10,11 are decided so as to reduce the number of data transfer operations from one site to another.
I have tried to impart a characteristic property to Site 4 which is it deals with the primary data that will be accessed on a very frequent basis for both restaurants and users. Since User_Remaining is the one which appears in the case where we also edit our data which is not gonna happen very frequently if we talk in general, that's why it is kept in a different site other than Site 4

Order_Items, is kept in site 2 in consideration of the fact that it will be a fast growing table when compared to the other relations. It will be the one which will have a lot of data entries happening in a single day if we consider daily life applications. So, I have tried to keep Site 2 mostly for this purpose. Other details like how we are performing aggregates on the databases were also kept in consideration such that we can reduce our data transfer operations. In this way Allocation is done

SYSTEM CATALOG RELATIONS WITH DATA FOR FRAGMENTATION AND ALLOCATION

```
mysql> SELECT * FROM Allocation;
+---------------------------------------+------+
| Fragment_Name                         | Site |
+---------------------------------------+------+
| Food_Item_Chinese                     |    1 |
| User_Restaurant_Order_Amount_Chinese  |    1 |
| Food_Item_Indian                      |    2 |
| Order_Items                           |    2 |
| User_Remaining                        |    2 |
| User_Restaurant_Order_Amount_Indian   |    2 |
| Food_Item_Italian                     |    3 |
| User_Restaurant_Order_Amount_Italian  |    3 |
| Restaurants_Minimal                   |    4 |
| Restaurants_Remaining                 |    4 |
| User_Minimal                          |    4 |
+---------------------------------------+------+
11 rows in set (0.00 sec)
```

^Allocation Schema

```
mysql> SELECT * FROM Horizontal_Fragments;
+--------------------+------------+-----------------+
| Fragment_Name      | Table_Name | Clause          |
+--------------------+------------+-----------------+
| Food_Item_Chinese | Food_Item  | Type == Chinese |
| Food_Item_Indian  | Food_Item  | Type == Indian  |
| Food_Item_Italian | Food_Item  | Type == Italian |
+--------------------+------------+-----------------+
3 rows in set (0.00 sec)
```

^Horizontal Fragments Schema

```
mysql> SELECT * FROM Vertical_Fragments;
+------------------------+-------------+
| Fragment_Name          | Table_Name  |
+------------------------+-------------+
| Restaurants_Minimal    | Restaurants |
| Restaurants_Remaining | Restaurants |
| User_Minimal           | User        |
| User_Remaining         | User        |
+------------------------+-------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM VF_Columns;
+------------------------+--------------+
| Fragment_Name          | Column_Name  |
+------------------------+--------------+
| Restaurants_Minimal    | Name         |
| Restaurants_Minimal    | PK_Custom    |
| Restaurants_Minimal    | Rating       |
| Restaurants_Minimal    | Specialty    |
| Restaurants_Remaining | Address      |
| Restaurants_Remaining | Email        |
| Restaurants_Remaining | Num_Reviews  |
| Restaurants_Remaining | PK_Custom    |
| User_Minimal           | Email        |
| User_Minimal           | Name         |
| User_Minimal           | PK_Custom    |
| User_Remaining         | Address      |
| User_Remaining         | Phone_Number |
| User_Remaining         | PK_Custom    |
+------------------------+--------------+
14 rows in set (0.00 sec)
```

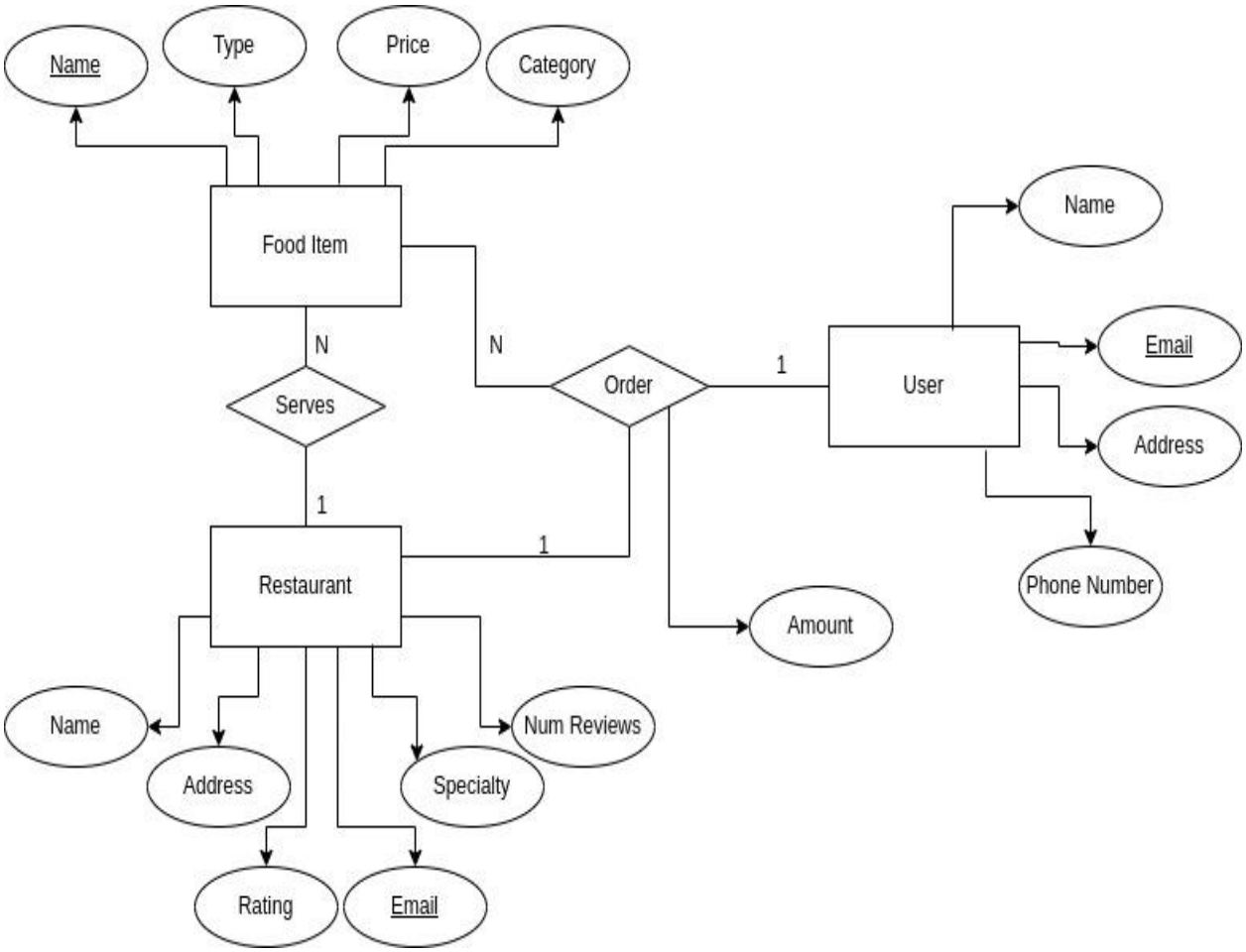
^Vertical Fragments Schema

```
mysql> SELECT * FROM Derived_Horizontal_Fragments;
+-----------------------------------+-------------------------+---------------------------+
| Fragment_Name                     | Horizontal_Fragment_Name | Relation                  |
+-----------------------------------+-------------------------+---------------------------+
| User_Restaurant_Order_Amount_Chinese | Food_Item_Chinese       | Order_Items JOIN Food_Item |
| User_Restaurant_Order_Amount_Indian  | Food_Item_Indian        | Order_Items JOIN Food_Item |
| User_Restaurant_Order_Amount_Italian | Food_Item_Italian       | Order_Items JOIN Food_Item |
+-----------------------------------+-------------------------+---------------------------+
3 rows in set (0.00 sec)
```

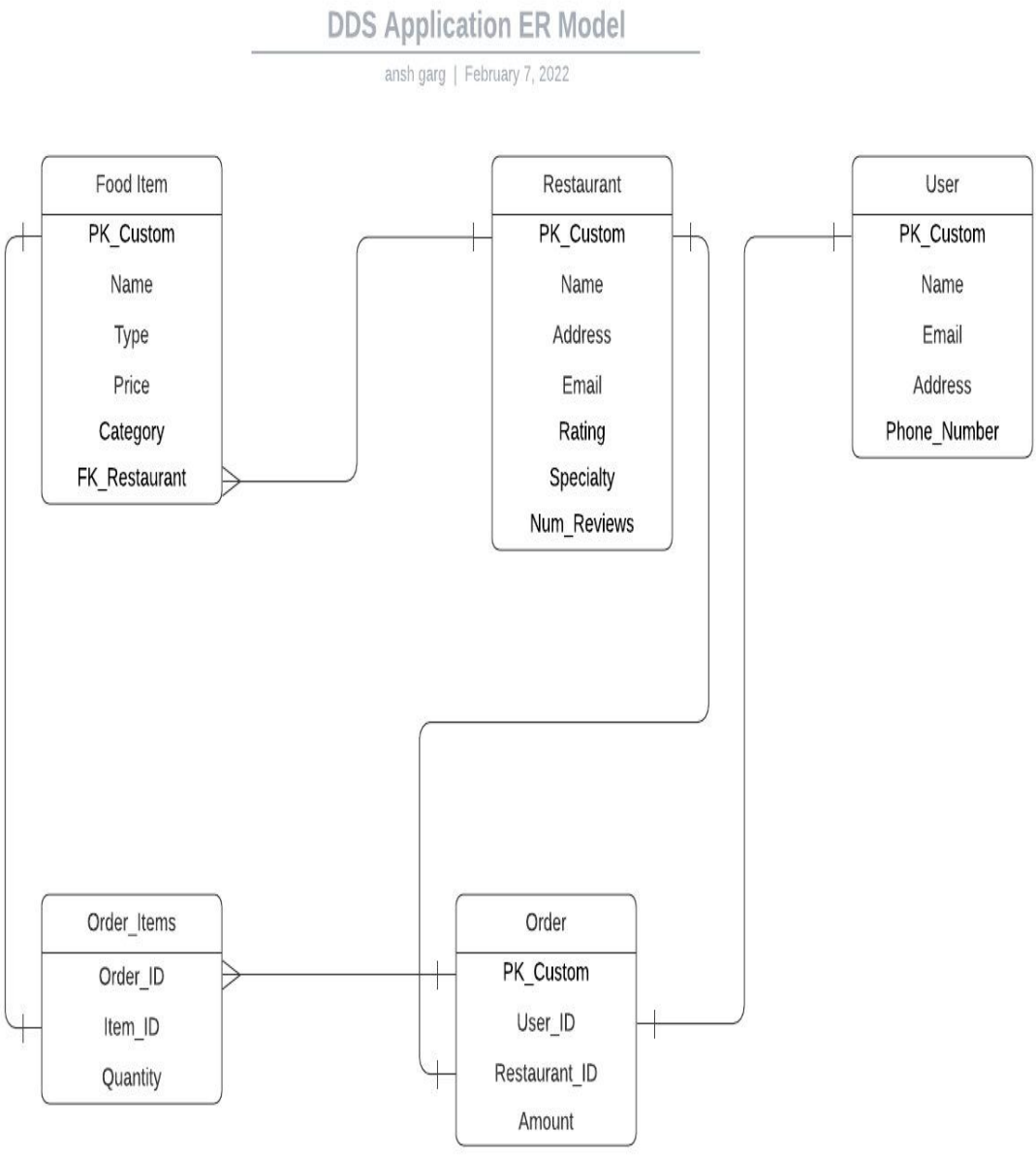
^Derived Horizontal Fragments Schema

ER and RELATIONAL MODELS

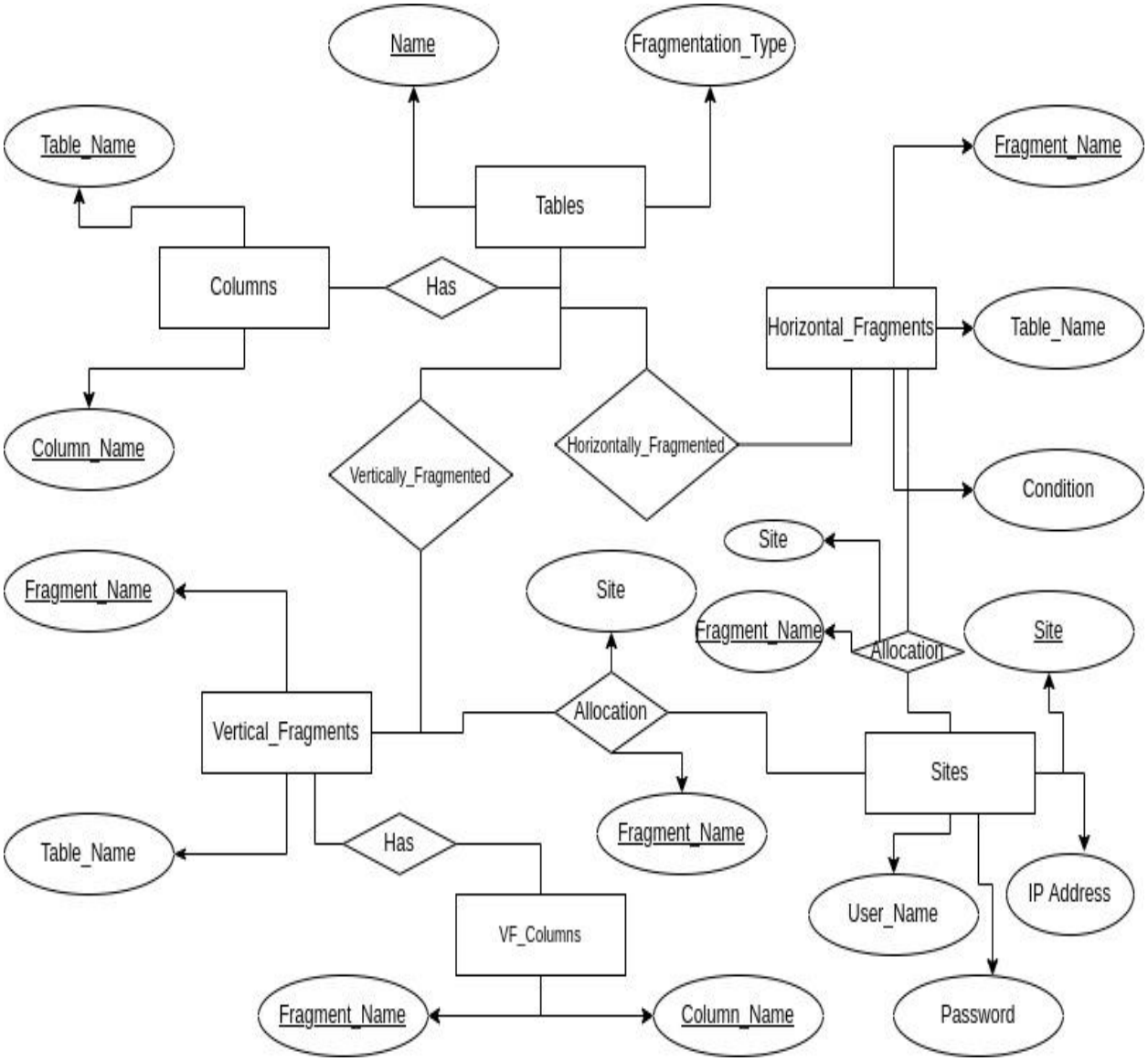
1. Application DBMS:
    a. Relational Diagram



b. ER Model


DDS Application ER Model
ansh garg | February 7, 2022
Food Item
PK_Custom
Name
Type
Price
Category
FK_Restaurant
Restaurant
PK_Custom
Name
Address
Email
Rating
Specialty
Num_Reviews
User
PK_Custom
Name
Email
Address
Phone_Number
Order_Items
Order_ID
Item_ID
Quantity
Order
PK_Custom
User_ID
Restaurant_ID
Amount

2. System Catalog
    a. Relational Diagram

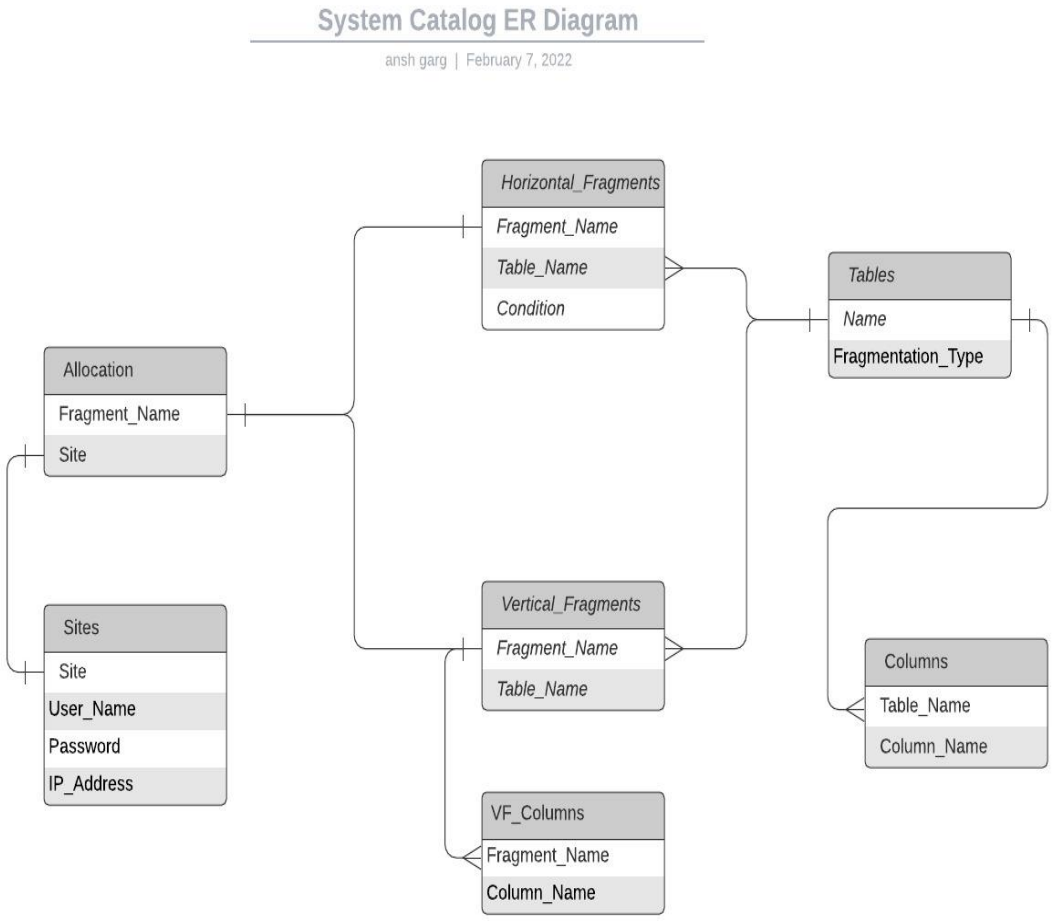


One more table named as “Derived_Horizontal_Fragments” also exists with attributes as:

1. Fragment_Name
2. Horizontal_Fragment_Name : refers to Fragment_Name in Horizontal_Fragments
3. Relation : To be semi joined with fragment mentioned in “Horizontal_Fragment_Name”

Condition in Horizontal_Fragments is renamed to Clause, mysql was giving some error while processing Condition

b. ER Model



DIRECTORY STRUCTURE

1. Report_Outlaws.pdf : Report for this Phase.
2. Codes: A directory which contains the codes that were used to create system catalog and application dbms at various sites.
    a. catalog.sql : File for creating system catalog
    b. application_site1.sql : File to create tables on site 1
    c. application_site2.sql : File to create tables on site 2
    d. application_site3.sql : File to create tables on site 3
    e. application_site4.sql : File to create tables on site 4
3. ER_Model_Application.jpeg : Contains the UML ER diagram for the application dbms
4. ER_Model_System_Catalog.jpeg : Contains the UML ER diagram for the System Catalog
5. Relational_Model_Application.jpg : Contains the relational diagram for the application dbms
6. Relational_Model_System_Catalog.jpg : Contains the relational diagram for the System Catalog