BUILD: Project I / Design & Implement a Relational Database

Project proposal:

We're going to build a restaurant review management system that lets users submit reviews for a restaurant and lets the restaurant/user manage these reviews.
The application will support:

- CRUD operations of the reviewers
- CRUD operations of the restaurants
- CRUD operations of the reviews
- Query/filter reviews based on cost, service, parking, waiting time, and overall rating
- Query/filter restaurants based on name, food type, working hours, accepted payment methods, Facilities
- Query/filter users based on food preferences and available payment methods

Requirements of the problem:

In today's age **restaurants** lean on services like yelp to help them advertise their **restaurant**. These services try to summarize the **customer's** experience by allowing them to *rate and review* the restaurants. Restaurants use the services to build credibility by *asking* customers for positive reviews when they feel that the customer is happy. The Project consists of the following parts: The ability to add reviews, customer and restaurants.

The customer should be able to *create* an **account** by providing the following details: Information on if they are a **smoker**, if they are **heavy drinkers**, the **budget**, the **ambiance** the customer prefers, the preferred **dress code**, what **cuisine** they like and what **payment methods** they have available.

After creating an account, a customer can *write* reviews for an already registered restaurant. A restaurant has the following information provided: The **address**, **city, state, country**, **price range**, the **cuisine** they are serving. The restaurant also provides certain **facilities** like **parking space** available, **ambience, seating area** and **services** like **alcohol, dress code, smoking area.** They have a set of available **payment methods** and **working hours.**

When *creating* a review, the customer *provides* a **rating** of 0-5 for the **cost**, **food**, **service**, **parking**, **waiting time**, and if they want a written **review**.

Nouns in bold and verbs in Italic

Nouns:

Restaurants
- Name
- Address
- Price Range
- Cuisine

Facilities
- Ambience
- Seating Area
- Parking

Services
- Alcohol
- dress code
- smoking area
- payment methods
- working hours

Reviews
- Cost
- Food
- Service
- Parking
- waiting time

Account/customer
- smoker
- heavy drinkers
- Budget
- Ambiance
- dress code
- Cuisine
- payment methods
- name

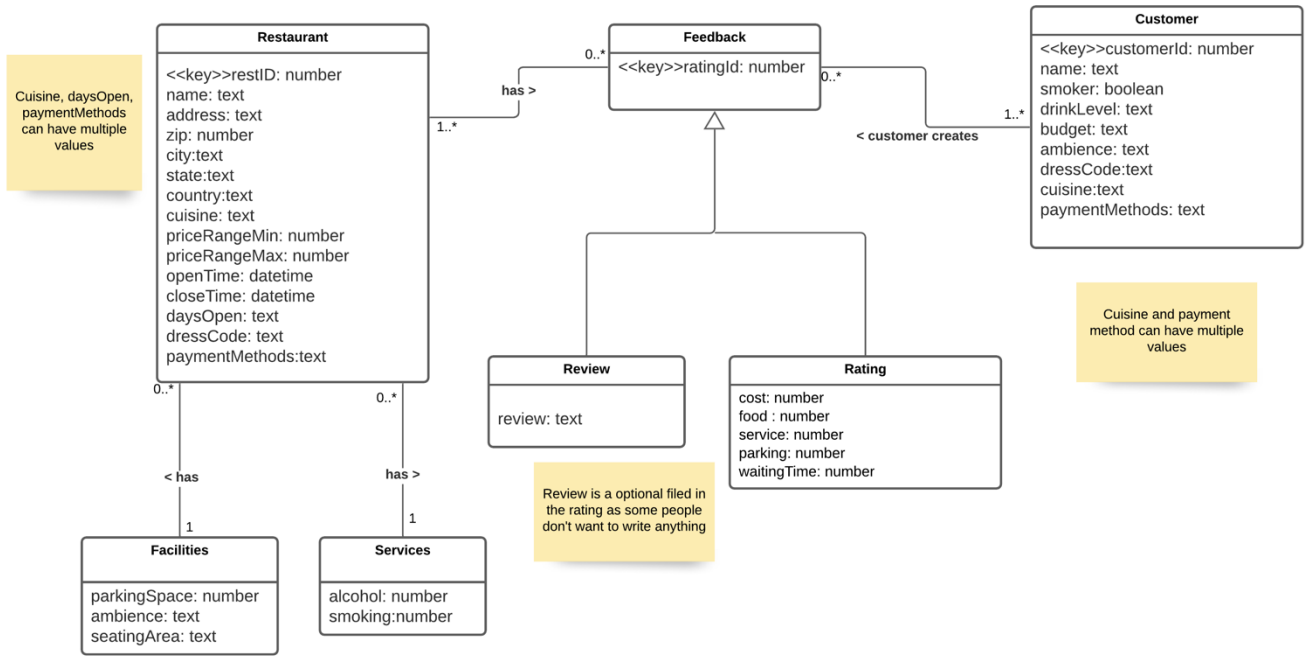
Verbs:

- Asking – restaurants ask customers to review them
- Creates - customer creates reviews on restaurant
- Provides – restaurants provide certain facilities and services

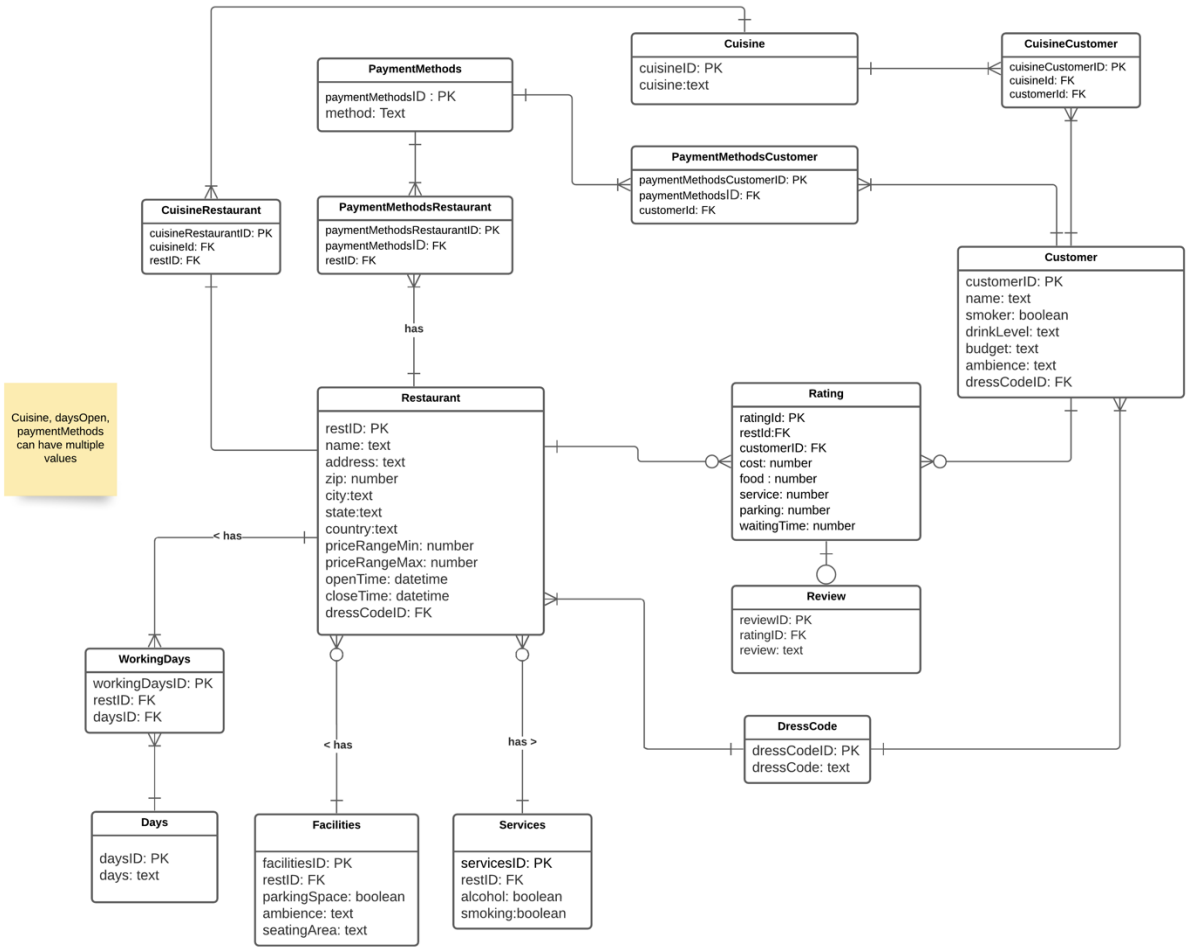
Business Rules:

1. Facilities can be offered by many restaurants while each restaurant provides a set of facilities.
2. Services can be offered by many restaurants while each restaurant provides a set of services.
3. A restaurant can get none to many feedbacks while feedback has 1 or more restaurants
4. Customer can create 0 or more feedbacks while feedbacks have 1 or more customers
5. Cuisine, Days Open and Payment methods can have multiple values
6. There are about 19 cuisines in total to choose from
7. There are 5 total dress code types
8. There are 5 different payment methods
9. 7 working days

Conceptual model:



Logical model:



https://lucid.app/lucidchart/dfef3e2d-917a-4e03-b69d-3cdab87b2dab/edit?page=0_0&invitationId=inv_24237a88-360d-4f8c-a0d9-cafe2d86331f#

1. Many-to-Many relation between Restaurant and feedback & customer and feedback has been broken down.
2. Multivalued attributes like Cuisine, Working Days, Payment Methods have been eliminated
3. Dress code can have a certain range of 5 values like Informal, formal, casual,etc and hence is separated.

Relational scheme:

1. Restaurant(restID,dressCodeID,name,address, zip, city, state, country, priceRangeMin, priceRangeMax, openTime, closeTime)

2. WorkingDays(restID, daysID)

3. Days(daysID, day)

4. Facilities(restID, parkingSpace,ambience,seatingArea)

5. Services(restID, servicesID , alcohol,smoking)

6. PaymentMethodsRestaurant(paymentMethodsRestaurantID, paymentMethodsID, restID)

7. CuisineRestaurant(cuisineCustomerID, cuisineId, restID)

8. DressCode(dressCodeID, dressCode)

9. Cuisine(cuisineID, cuisine)

10. Rating(ratingId, cost, food, service, parking, waitingTime)

11. Review(reviewID, ratingID, review)

12. Customer(customerID, restId, customerID name, smoker, drinkLevel, budge, ambience, dressCodeID)

13. CuisineCustomer(cuisineCustomerID, cuisineId, customerId)

14. PaymentMethodsCustomer(paymentMethodsCustomerID, paymentMethodsID, customerId)

15. PaymentMethods(paymentMethodsID, method)

Relational schema in at least BCNF

SQL data definition statements

Cuisine:

```
CREATE TABLE "Cuisine" (
"cuisineID"	INTEGER NOT NULL UNIQUE,
"cuisine"	TEXT NOT NULL,
PRIMARY KEY("cuisineID" AUTOINCREMENT)
)
```

CuisineCustomer:

```
CREATE TABLE "CuisineCustomer" (
"cuisineCustomerID"	INTEGER NOT NULL UNIQUE,
"customerId"	INTEGER,
"cuisineId"	INTEGER NOT NULL,
PRIMARY KEY("cuisineCustomerID" AUTOINCREMENT),
FOREIGN KEY("customerId") REFERENCES "Customer"("customerID"),
FOREIGN KEY("cuisineId") REFERENCES "Cuisine"("cuisineID")
)
```

CuisineRestaurant:

```
CREATE TABLE "CuisineRestaurant" (
"cuisineRestaurantID"	INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"cuisineId"	INTEGER NOT NULL,
PRIMARY KEY("cuisineRestaurantID" AUTOINCREMENT),
FOREIGN KEY("restID") REFERENCES "Restaurant"("restID"),
FOREIGN KEY("cuisineId") REFERENCES "Cuisine"("cuisineID")
)
```

Restaurant:

```
CREATE TABLE "Customer" (
"customerID"    INTEGER NOT NULL UNIQUE,
"name" TEXT NOT NULL,
"smoker"        INTEGER NOT NULL,
"drinkLevel"    TEXT NOT NULL,
"dressCodeID"   INTEGER NOT NULL,
"ambience"      TEXT NOT NULL,
"budget"        TEXT NOT NULL,
PRIMARY KEY("customerID" AUTOINCREMENT),
FOREIGN KEY("dressCodeID") REFERENCES "DressCode"("dressCodeID")
)
```

Days:

```
CREATE TABLE "Days" (
"daysID"        INTEGER NOT NULL UNIQUE,
"days"   TEXT NOT NULL,
PRIMARY KEY("daysID" AUTOINCREMENT)
)
```

DressCode:

```
CREATE TABLE "DressCode" (
"dressCodeID"   INTEGER NOT NULL UNIQUE,
"dressCode"     TEXT NOT NULL,
PRIMARY KEY("dressCodeID" AUTOINCREMENT)
)
```

Facilities:

```
CREATE TABLE "Facilities" (
"facilitiesID"      INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"parkingSpace"  INTEGER NOT NULL,
```

"ambience"	TEXT NOT NULL,

"seatingArea"	TEXT NOT NULL,

PRIMARY KEY("facilitiesID" AUTOINCREMENT),

FOREIGN KEY("restID") REFERENCES "Restaurant"("restID")

)

PaymentsMethods:

CREATE TABLE "PaymentMethods" (

"paymentMethodsID"	INTEGER NOT NULL UNIQUE,

"method"	BLOB NOT NULL,

PRIMARY KEY("paymentMethodsID" AUTOINCREMENT)

)

PaymentMethodsCustomer:

CREATE TABLE "PaymentMethodsCustomer" (

"paymentMethodsCustomerID" INTEGER NOT NULL UNIQUE,

"customerId"	INTEGER NOT NULL,

"paymentMethodsID"	INTEGER NOT NULL,

PRIMARY KEY("paymentMethodsCustomerID" AUTOINCREMENT),

FOREIGN KEY("paymentMethodsID") REFERENCES "PaymentMethods"("paymentMethodsID"),

FOREIGN KEY("customerId") REFERENCES "Customer"("customerID")

);

PaymentMethodsRestaurant:

CREATE TABLE "PaymentMethodsRestaurant" (

"paymentMethodsRestaurantID"	INTEGER NOT NULL UNIQUE,

"restID"INTEGER NOT NULL,

"paymentMethodsID"	INTEGER NOT NULL,

PRIMARY KEY("paymentMethodsRestaurantID" AUTOINCREMENT),

FOREIGN KEY("restID") REFERENCES "Restaurant"("restID"),

FOREIGN KEY("paymentMethodsID") REFERENCES "PaymentMethods"("paymentMethodsID")

);

Rating:

```
CREATE TABLE "Rating" (
"ratingId"          INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"customerID"    INTEGER NOT NULL,
"cost"    INTEGER NOT NULL,
"Food"  INTEGER NOT NULL,
"Service"           INTEGER NOT NULL,
"parking"           INTEGER NOT NULL,
"waiting"           INTEGER NOT NULL,
"overall"           REAL NOT NULL,
PRIMARY KEY("ratingId" AUTOINCREMENT),
FOREIGN KEY("restID") REFERENCES "Restaurant"("restID"),
FOREIGN KEY("customerID") REFERENCES "Customer"("customerID")
);
```

Restaurant:

```
CREATE TABLE "Restaurant" (
"restID"INTEGER NOT NULL UNIQUE,
"dressCodeID"   INTEGER NOT NULL,
"name" TEXT NOT NULL,
"address"           TEXT NOT NULL,
"zip"     INTEGER NOT NULL,
"city"    TEXT NOT NULL,
"state"  TEXT NOT NULL,
"country"           TEXT NOT NULL,
"priceRangeMin"           INTEGER NOT NULL,
"priceRangeMax"           INTEGER NOT NULL,
"openHours"     TEXT NOT NULL,
"closeHours"     TEXT NOT NULL,
```

PRIMARY KEY("restID" AUTOINCREMENT),

FOREIGN KEY("dressCodeID") REFERENCES "DressCode"("dressCodeID")

);

Review:

CREATE TABLE "Review" (

"reviewID" INTEGER NOT NULL UNIQUE,

"review" TEXT NOT NULL,

"ratingID" INTEGER NOT NULL,

PRIMARY KEY("reviewID" AUTOINCREMENT),

FOREIGN KEY("ratingID") REFERENCES "Rating"("ratingId")

);

Services:

CREATE TABLE "Services" (

"servicesID" INTEGER NOT NULL UNIQUE,

"restID"INTEGER NOT NULL,

"alcohol" INTEGER NOT NULL,

"smoking" INTEGER NOT NULL,

PRIMARY KEY("servicesID" AUTOINCREMENT),

FOREIGN KEY("restID") REFERENCES "Restaurant"("restID")

);

WorkingDays:

CREATE TABLE "WorkingDays" (

"workingDaysID" INTEGER NOT NULL UNIQUE,

"restID"INTEGER NOT NULL,

"daysID" INTEGER NOT NULL,

PRIMARY KEY("workingDaysID" AUTOINCREMENT),

FOREIGN KEY("daysID") REFERENCES "Days"("daysID"),

FOREIGN KEY("restID") REFERENCES "Restaurant"("restID")

);

Define and execute at least five queries that show your database

1. **JOIN OF 3 TABLES**

   select R.restID,R.name,R.address,D.days from Restaurant R

   inner join WorkingDays W on R.restID = W.restID

   inner join Days D on D.daysID = W.daysID

   where D.days = "Sunday" or D.days= "Saturday"

2. **SUBQUERY**

   select restID,name,address,city,state,zip from Restaurant where restID in (

   select restID from CuisineRestaurant where cuisineId in(

   select cuisineId from cuisine where cuisine like "%Chinese%")

)

3. **GROUP BY WITH A HAVING CLAUSE**

   select restID,name,address,priceRangeMax from Restaurant R

   group by priceRangeMax

   having priceRangeMax < 100

4. **COMPLEX SEARCH CRITERION**

   select R.restID,R.name,R.address from Restaurant R

   inner join Facilities F on R.restID = F.restID

   inner join Services S on R.restID = S.restID

   inner join WorkingDays W on R.restID = W.restID

   inner join days D on W.daysID = D.daysID

   WHERE D.days = 'Saturday'

   AND F.seatingArea = "open"

   AND (S.smoking = 1 or S.alcohol = 1)

   AND F.parkingSpace = 1

5. **SELECT CASE/WHEN.**

```
SELECT name, zip, restID , country, state from Restaurant
ORDER BY (CASE WHEN state IS NULL THEN country ELSE state END);
```