Domain Types

Outline here the domain types you created, their names, types, the reason you included them and an example how you used them e.g.

Domain Type Name	Type Definition	Reason Introduced	Example
Address	VARCHAR(50)	For attributes which contain address strings	address in entity staffTable
Age	NUMERIC(3)	For attributes which contain a persons age at precision 3 digits	age in entity custbooktable
Time	TIMESTAMP WITH LOCAL ZONE	Date and time attributes which require a very clearly defined time element	timearrival in entity customer
ID	Numeric(8)	For attributes which contain the ID of the specific table.	tableID in entity Table
Count	Numeric(3)	For attributes which require a number of specific elements.	guestcount in entity bookingreceipt
Amount	Numeric(5,2)	For attributes which require a fee or price for something.	fee in entity penaltybill
Email	VARCHAR(50)	For attributes which require an email address.	email in entity custbooktable
Phone Number	Numeric(9)	For attributes which require a telephone or mobile number of length 9.	phonenumber in entity customer
Location	VARCHAR(20)	For attributes which require the name of a specific place.	tablelocation in entity Table

Name	VARCHAR(20)	For attributes which require the name of a person.	fname in entity staffTable

Major Decisions

Have a section per decision. This can include decisions such as where to place particular attributes, whether to have an entity, what type of relationship to use. Anything you had to discuss you should include in this section.

Entities and foreign keys

We created a table called "customers" for all customers that entered the restaurant including the people booking. We used a separate table to identify the people that booked in "custbooktable". Here we use a foreign primary key from "customer" table and this table could contain details of the person that booked the table. This way we can view who booked without the confusion of many different ID's for customer who booked and a regular customer. In the "custbooktable" each customer must declare their age when they book, we think that information was only a necessity for the people that are booking tables as stated in the case study and not for all customers. In our booking receipt we included the tableID as a foreign key that way when we form a join, we can see what table was booked, which type, location and which restaurant branch the booking took place, many customers can book the same table at different times. "bookingreceipt" also contains the foreign key of customerID (from the customer that booked the table). The reason being that if a customer wishes to book more than once their costumerID which identifies their details is kept the same. We also created two tables to find the "manager" and "waiter". Within each table contains the primary key of a unique of either MangerID or WaterID and its respective StaffID with a unique constraint. We created it this way so we can display both the waiter and manager serving the table that was booked.

We feel it was important to link the customerID from customer table as a foreign key to the penaltybill table, we used a penaltybillID to identify this fee. We believe that it was easier to know which customer receives a fee because the CustomerID is a unique identifier to that customer and the penaltybillID is assigned to customersID. Time of arrival and time leaving is also included in that table.

We also separate the name from first name (fname) and last name (lname) in any case the user wants to view the tables based on either parts instead of a the full name.

Table relationships

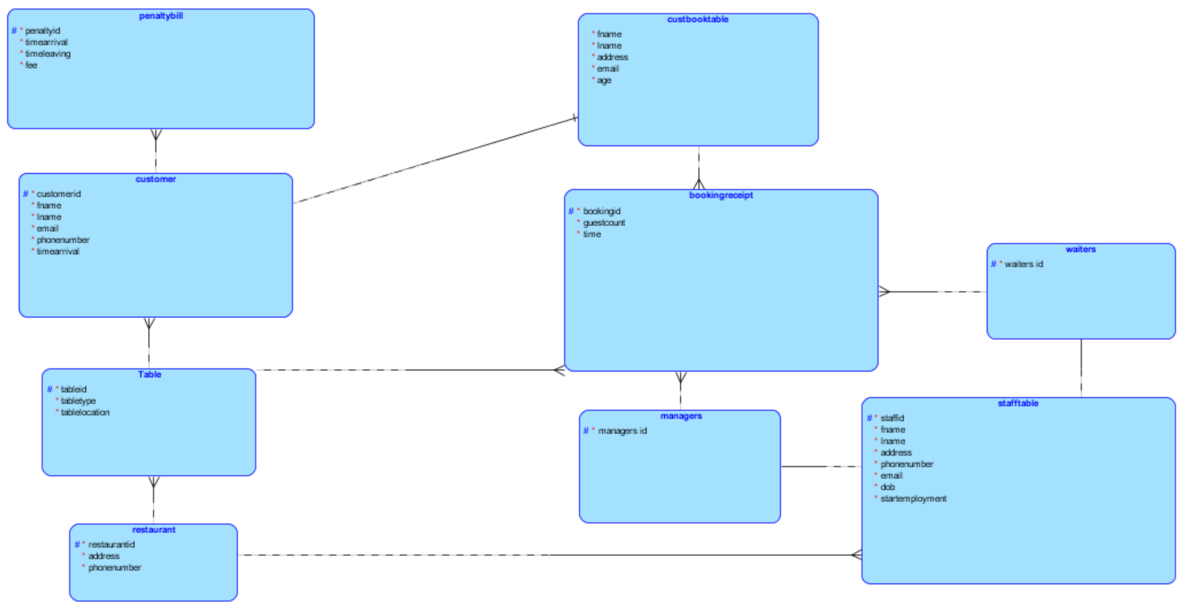
Customer table and custbooktable has a one-to-one relationship because we are reusing the primary key of customer table as the same primary key to identify customers who booked the table. This type of relationship is similar with stafftable and managers table or waiter's table. Because each managerID or waiterID belongs to only one staff ID. Customer table has a one-to-many relationship with "table", because multiple customers could be using the same table. This relationship and reason are the same with the booking receipt as values from those tables (including manager table and waiter table) may appear more than once. Similarly, other examples include restaurant with stafftable. Custbooktable has a one-to-many relationship with bookingreceipt as the same customer can have many bookings within the restaurant.

Constraints.

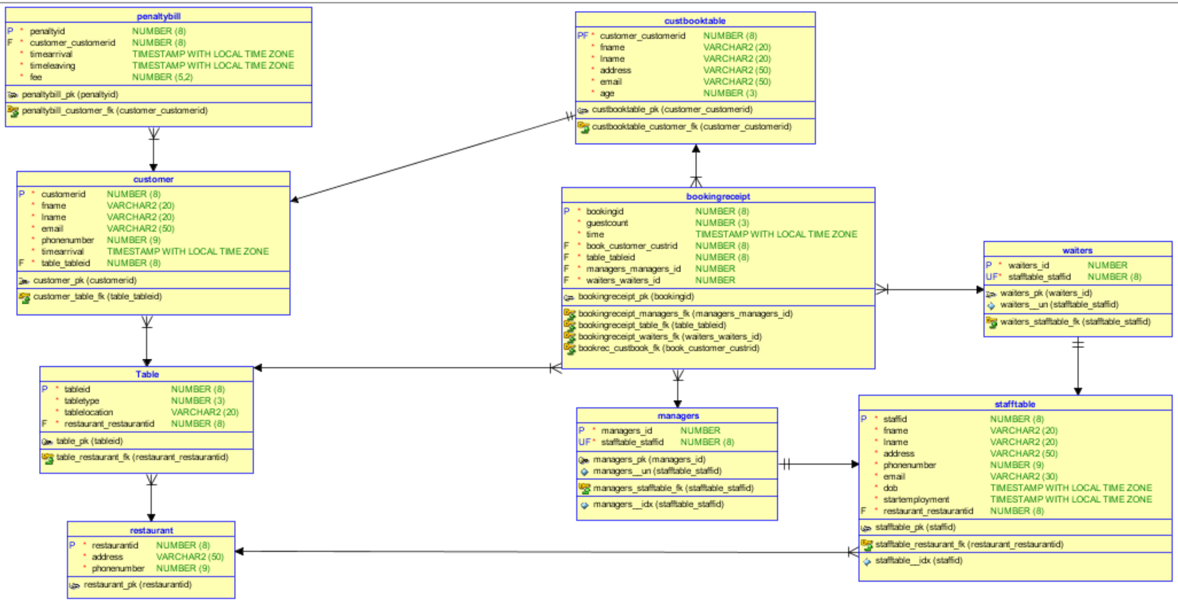
It was a major decision to include constraints in our database. For example, we had to ensure that there are people under 18 were allowed to book. This is done to meet the criteria that only people at a certain age can book. We also added constraints that require the symbol ‘@’ for any column that require emails. This is to ensure that the User does not forget to add this symbol and provide extra error checking. For staff table its compulsory for all staff members to use the staff email and so we added constraints that require “@burgershack.com”. It is important that the number of people in the table was booked was no more than 6 (guestcount).

Present ERD

Screenshot of Logical Model



Screenshot of Physical Model


penaltybill
P * penaltyid NUMBER (8)
F * customer_customerid NUMBER (8)
* timearrival TIMESTAMP WITH LOCAL TIME ZONE
* timeleaving TIMESTAMP WITH LOCAL TIME ZONE
* fee NUMBER (5,2)
penaltybill_pk (penaltyid)
penaltybill_customer_fk (customer_customerid)
custbooktable
PF * customer_customerid NUMBER (8)
* fname VARCHAR2 (20)
* lname VARCHAR2 (20)
* address VARCHAR2 (50)
* email VARCHAR2 (50)
* age NUMBER (3)
custbooktable_pk (customer_customerid)
custbooktable_customer_fk (customer_customerid)
customer
P * customerid NUMBER (8)
* fname VARCHAR2 (20)
* lname VARCHAR2 (20)
* email VARCHAR2 (50)
* phonenumber NUMBER (9)
* timearrival TIMESTAMP WITH LOCAL TIME ZONE
F * table_tableid NUMBER (8)
customer_pk (customerid)
customer_table_fk (table_tableid)
bookingreceipt
P * bookingid NUMBER (8)
* guestcount NUMBER (3)
* time TIMESTAMP WITH LOCAL TIME ZONE
F * book_customer_custrid NUMBER (8)
F * table_tableid NUMBER (8)
F * managers_managers_id NUMBER
F * waiters_waiters_id NUMBER
bookingreceipt_pk (bookingid)
bookingreceipt_managers_fk (managers_managers_id)
bookingreceipt_table_fk (table_tableid)
bookingreceipt_waiters_fk (waiters_waiters_id)
bookrec_custbook_fk (book_customer_custrid)
waiters
P * waiters_id NUMBER
UF * stafftable_staffid NUMBER (8)
waiters_pk (waiters_id)
waiters__un (stafftable_staffid)
waiters_stafftable_fk (stafftable_staffid)
Table
P * tableid NUMBER (8)
* tabletype NUMBER (3)
* tablelocation VARCHAR2 (20)
F * restaurant_restaurantid NUMBER (8)
table_pk (tableid)
table_restaurant_fk (restaurant_restaurantid)
managers
P * managers_id NUMBER
UF * stafftable_staffid NUMBER (8)
managers_pk (managers_id)
managers__un (stafftable_staffid)
managers_stafftable_fk (stafftable_staffid)
managers__idx (stafftable_staffid)
stafftable
P * staffid NUMBER (8)
* fname VARCHAR2 (20)
* lname VARCHAR2 (20)
* address VARCHAR2 (50)
* phonenumber NUMBER (9)
* email VARCHAR2 (30)
* dob TIMESTAMP WITH LOCAL TIME ZONE
* startemployment TIMESTAMP WITH LOCAL TIME ZONE
F * restaurant_restaurantid NUMBER (8)
stafftable_pk (staffid)
stafftable_restaurant_fk (restaurant_restaurantid)
stafftable__idx (staffid)
restaurant
P * restaurantid NUMBER (8)
* address VARCHAR2 (50)
* phonenumber NUMBER (9)
restaurant_pk (restaurantid)

Screenshot of details that you have incorporated into your model such as unique or check constraints.

Constraint for bookingrecipt where guestcount values have to be between 1 to 6

```
ALTER TABLE bookingreceipt
    ADD CONSTRAINT bookingreceipt_ck_1 CHECK ( guestcount BETWEEN 1 AND 6 );
```

Constraint for custbooktable where email must contain @ in between the string and age must be no less than 18

```
ALTER TABLE custbooktable ADD CONSTRAINT custbooktable_ck_2 CHECK ( email LIKE '%@%' );

ALTER TABLE custbooktable ADD CONSTRAINT custbooktable_ck_1 CHECK ( age >= 18 );
```

Constraint for staff table must contain the string Burgershack.com for emails

```
ALTER TABLE stafftable ADD CONSTRAINT stafftable_ck_1 CHECK ( email LIKE '%@burgershack.com%' );
```

Creating a unique key for foreign key staffid in waiters tables

```
ALTER TABLE waiters ADD CONSTRAINT waiters__un UNIQUE ( stafftable_staffid );
```

Creating a unique key for foreign key staffid in waiters table

```
ALTER TABLE managers ADD CONSTRAINT managers__un UNIQUE ( stafftable_staffid );
```