Problem Statement

Badafone India entrusts us with developing a business-to-user platform for their application. Here are the Instructions given

Each customer may have only and only one SIM Card. We need to know their personal information, like name, aadhar number, birthdate, and e-mail. Each SIM Card will be connected to a phone number. It may or may not be activated.

Each SIM Card will be issued with a "Home Tower" within its own city. If the SIM Card wanders outside of the home tower's radius, then it will be considered to be in "roaming range". Towers are used to transmit signal between the caller and callee.

Communication can be done via two means. One is through an audio call. Each call starts and ends at some time, and occurs between two phone numbers. We also want to know which towers the calls started in for bookkeeping purposes. The second mode is via SMS. An SMS is also transmitted from one phone to another, but we don't care about the tower that an SMS travels through. SMS contains some text in it. We also want to know if the receiver read the message, and may provide Read Receipts in the future.

Of course, none of this is possible without plans. Each subscription plan has a fixed validity (duration in days). It also has a price, and some name. Plans can be either calling plans or data plans, but not both. A data plan provides local, national, and international calling minutes. They also put an upper limit on calls and SMSes. Alternatively, a user can buy a data plan. They will receive high speed up till a fixed cap, and a much lower speed afterward. The data limit may also reset every fixed duration, like daily or weekly.

We also want to store the amount that a user has used up their plan, along with the plan itself. This way we can check if a user's plan has run out, and notify them to renew it. Speaking of renewal, all recharging is done through an in-app wallet. This can be connected to any payment method of the user's choice, like Credit Card, Net Banking, or UPI. All transactions should also be logged with the time at which they happened, so the users know we aren't scamming them.

If, however, they still think we're scamming them, they can contact the support team. To do so, they can issue a support ticket. Each support ticket will belong to one of the following categories: Calling Help, Data Help, Connectivity Issues, Plan Enquiry, Payment Enquiry. The query can also contain some text in it. This query

will be attended to by an employee, who will close it after talking with the customer.

Assumptions

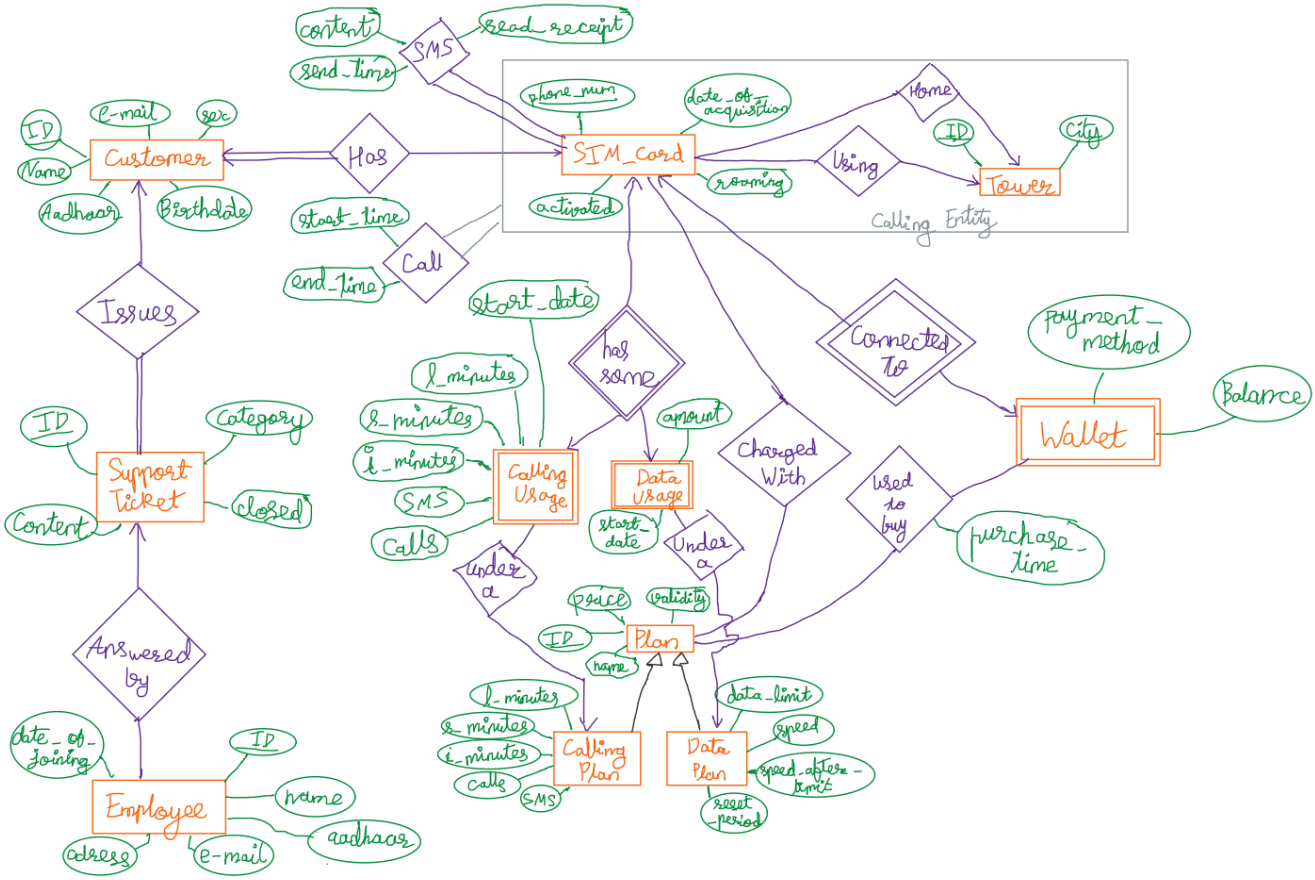
- A SIM Card may not be owned by anyone.
- Users would rather be referred to by a customer ID than their Aadhaar number, for privacy's sake.
- There will be only one tower in a city.
- Connecting Towers between the starting and ending tower are irrelevant.
- Limits on usage minutes and call count are independent of each other. Call Count limit does not care about the category of calling.
- A SIM Card can be loaded with two plans so user can have both calling and data facilities.
- Every plan MUST be a calling plan OR a data plan. It cannot be both or neither.
- Every support ticket is filed by a Customer. A ticket may not be answered by any employee, and will be answered by up to one employee.
- The company is not concerned about the responses employees send to support tickets.

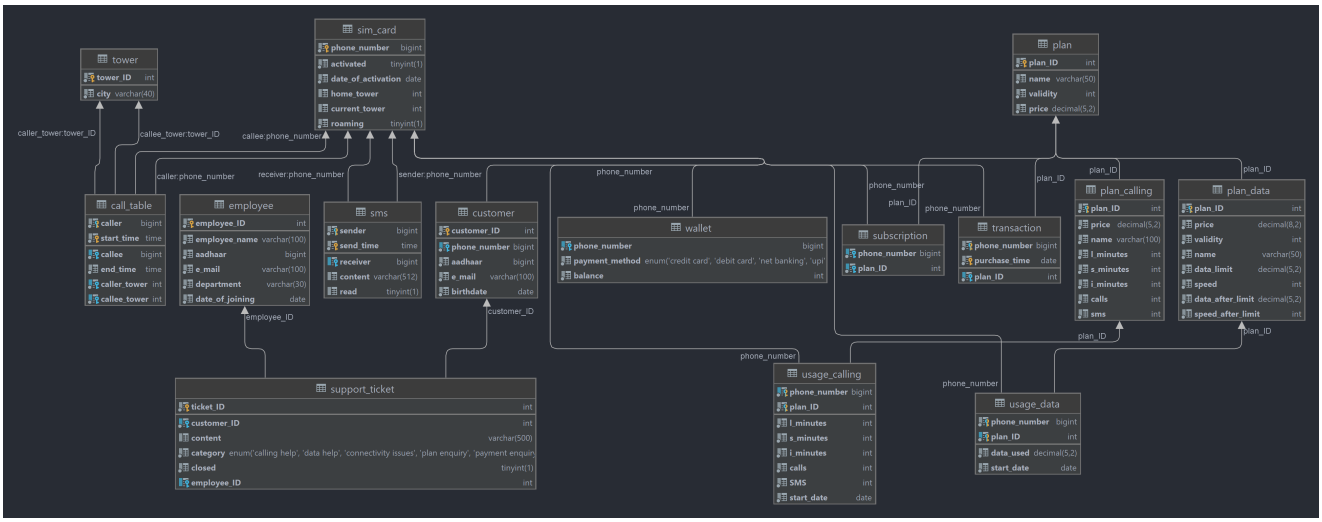
Stakeholders

Our Stakeholders are

- Users who want to make calls, send SMSes, recharge their plan, check validity of their calling or data plans, or send support tickets.
- Employees who want to see support tickets sent by users.
- Company accountants who want to see how much money was rolled in by sales.
- Sales Department who wants to know which the highest selling plans are.
- Administrators who want to see which users have finished their usage limits.

E-R Diagram





Relational Schemas

Customer (customer_ID, username, aadhar_ID, birthdate, phone_number)
SIM_Card (phone_number, date_of_activation, roaming, home_tower, current_tower)
Tower (tower_ID, city)

Call (caller, start_time, end_time, callee, caller_tower, callee_tower)
SMS (sender, send_time, receiver, read)

Wallet (phone_number, payment_method, balance)
Transaction (phone_number, purchase_time, plan_ID)

Subscription (phone_number, plan_ID)
Plan(plan_ID, price, validity, name)
Calling_Plan (plan_ID, price, validity, name, l_minutes, s_minutes, i_minutes, calls, SMS)

Data_Plan (plan_ID, price, validity, name, data_limit, speed, data_after_limit, reset_period, speed_after_limit)
Calling_Usage (phone_number, plan_ID, l_minutes, s_minutes, i_minutes, calls, SMS, start_date)
Data_Usage (phone_number, plan_ID, data_used, start_date)

Support_Ticket (ticket_ID, customer_ID, content, category, closed, employee_ID)
Employee (employee_ID, name, aadhar_ID, e-mail, date_of_joining)

Weak Entities

- The Wallet is defined as a weak entity, as it has no relevance without a Sim Card to be connected to.
- Calling and Data Usage information-entities are weak entities. These are entities that store information about the usage done by a customer. So, they have no independent existence without a user.

Ternary Relationship

- There is a ternary relationship between SIM Card, Calling Usage, and Data Usage. This is because Usage under a SIM card contains both calling and data usage. This ternary relationship is broken down into two binary relations in the relational schemas.

Miscellaneous Intricacies

- SMS and Call are relations that connect an entity set to itself. This showcases the usage of **Roles** in a relation.
- SIM_Card and Tower are **Aggregated** as a calling entity. This is because the Call relation needs both of their attributes. This aggregation could also be expressed as a ternary relationship.
- Plan is **Generalised** as either a Calling Plan or a Data Plan. This Generalisation is **Disjoint**, **Total** Generalisation. Each Plan *has to be* a Calling or Data Plan, but not both.

Relevant Queries

1. **Select all phone numbers that have dialled at a particular hour**

```
SELECT caller
FROM call_table
WHERE HOUR(starting_time) = 4
```

2. **Find the number of phone numbers in the same tower region as a given phone number**

```
SELECT COUNT(phone_number)
FROM SIM_Card
WHERE current_tower = (
    SELECT current_tower
        FROM SIM_Card
    WHERE phone_number = 78191502196
)
```

3. Select all customers with an open support ticket asking for calling help

```
SELECT customer.username
FROM customer
    INNER JOIN support_ticket
    ON customer.customer_ID = support_ticket.customer_ID
WHERE category = 'Calling Help' and closed = 0
```

4. Select all calling plans that cost under 500 rupees and last for more than 31 days

```
SELECT *
FROM plan_calling
WHERE price < 500 AND validity > 31
```

5. Find the number of plans of the "Maha" series in each validity category, except the fortnightly one.

```
SELECT COUNT(*)
FROM plan
WHERE name like 'maha%'
```

```
GROUP BY validity
HAVING validity > 15
```

6. Find all customers with names starting with J who purchased their SIM Card in Mumbai

```
SELECT username
FROM (Customer
 INNER JOIN SIM_Card AS T ON Customer.phone_number = T.phone_number)
 INNER JOIN Tower ON Tower.tower_ID = T.home_tower
WHERE username LIKE 'J%'
```

7. Make a view that stores all calling plans that have more than 25 international minutes, and offer SMS facility

```
CREATE VIEW international_special AS
SELECT name
FROM plan_calling
WHERE i_minutes > 25 AND SMS > 0
```

8. Select all users who have used up their data plan. Put this into a view called defaulters. Allow administrators to see this view. Also store the amount of extra data that has been used bty the defaulter.

```
CREATE VIEW defaulters AS
 SELECT sim_card.phone_number, Data_Usage.data_used - Data_Plan.data_l
    FROM usage_data as Data_Usage, plan_data as Data_Plan, SIM_Card
 WHERE Data_Usage.phone_number = SIM_Card.phone_number
        AND Data_Usage.plan_ID = Data_Plan.plan_ID
        AND Data_Usage.data_used >= Data_Plan.data_limit;

CREATE ROLE administrator;

GRANT SELECT ON defaulters TO administrator;
```

9. Find the names of top 10 plans that have rolled in the most users, and this valuable information with the sales team.

```sql
CREATE ROLE sales;

CREATE VIEW top_plans AS
SELECT plan.name, COUNT(*)
FROM subscription INNER JOIN plan ON subscription.plan_ID = plan.plan_
GROUP BY subscription.plan_ID
ORDER BY COUNT(*) DESC
LIMIT 10;

GRANT SELECT ON top_plans TO sales;
```

10. Find the total amount of money rolled in by Badafone Inc. from these top plans after deducting 18% GST, and share this information with the accounts team.

```sql
CREATE ROLE accounts;

CREATE VIEW revenue AS
SELECT SUM(val) FROM (
 SELECT COUNT(*) * plan.price AS val
   FROM subscription
 INNER JOIN plan ON subscription.plan_ID = plan.plan_ID
   GROUP BY subscription.plan_ID
   ORDER BY COUNT(*) DESC
 LIMIT 10
) as spv;

GRANT SELECT ON revenue TO accounts;
```