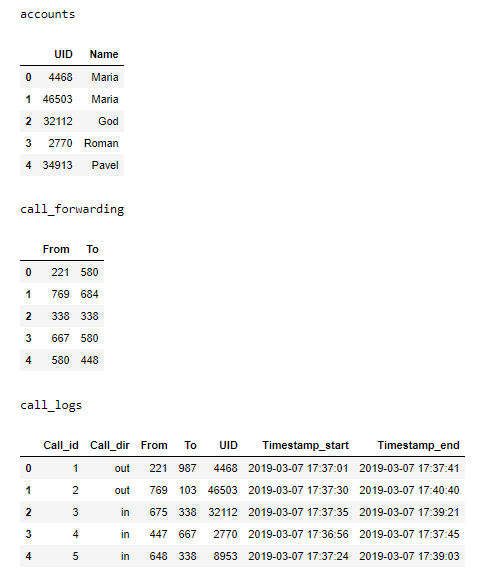
select \* from accounts;

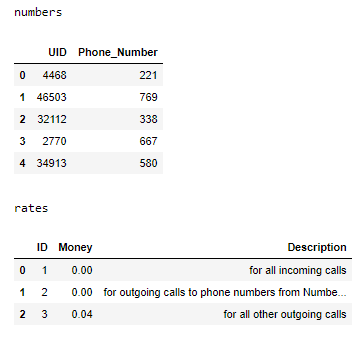
select \* from call\_forwarding;

select \* from call\_logs;

select \* from numbers;

select \* from rates;



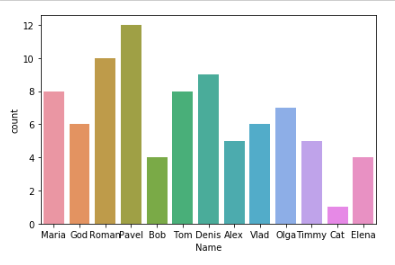


Make a brief look to this tables, we can already say something about database structure.  
Some call center collect data about calls. It has clients. Make calls to this client's numbers is free, according to rates. Also calls can forward to another numbers. Let's explore each table separately.

**Accounts**

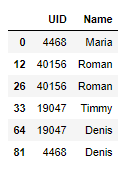
For each UID match a name of person. It doesn't give a lot inforamtion, because one of the reason is that this names is not unique.

Let's look, which names are the most common. I know, it's useless.



Check, is UID unique.

select \* from accounts where UID IN (select UID from accounts group by UID having count(\*)>1);

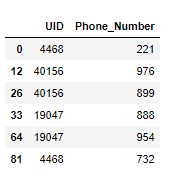


**numbers**

We can guess that numbers in this tables are clients of some company. So outcome calls for this clients is free, according to table rates.

Check uniqueness of UID.

select \* from numbers where UID IN (select UID from numbers group by UID having count(\*)>1);



One UID can be used for several numbers. This UIDs are the same as in table accounts.

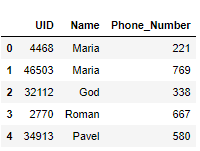
Tables accounts and numbers have the same length. Check if the UIDs column are identical.

select distinct accounts.UID = numbers.UID from

accounts inner join numbers on accounts.UID = numbers.UID;

They are the same. So we can merge first two tables.

select a.UID, a.Name, b.phone\_number from  
 (SELECT UID, Name, ROW\_NUMBER() OVER(ORDER BY (SELECT NULL)) AS rownum1  
 FROM accounts) a  
join  
 (SELECT UID, phone\_number, ROW\_NUMBER() OVER(ORDER BY (SELECT NULL)) AS rownum2  
 FROM numbers) b  
on a.rownum1 = b.rownum2;



**call\_forwarding**

call\_forwarding table obviously shows from which to which numbers calls redirect. I assume that call forwarding always forwards numbers in any cases for each call.

select distinct count(call\_forwarding.from) from call\_forwarding; - get first number

select distinct count(call\_forwarding.To) from call\_forwarding; - get second number

Output: 49 and 43

We can see that unique values from From column correspond to total number of rows in table. It's mean that we can exactly forward call without ambigious situations.  
 Now let's check if numbers in To column consist in From column.

select distinct call\_forwarding.To from call\_forwarding

where call\_forwarding.To In (select call\_forwarding.From from call\_forwarding);

Output: 580, 684, 338, 568, 428, 563, 753, 638, 221, 593, 647, 521, 888,

858, 769, 154, 182, 677, 557, 382, 335, 103, 145

This result means that there is sequence of forwarding calls. Let's find final numbers in each row of this sequence.

DROP TEMPORARY TABLE IF EXISTS nums\_To;

CREATE TEMPORARY TABLE nums\_To

select call\_forwarding.From , call\_forwarding.From as forwarded from call\_forwarding;

Drop procedure if exists while\_loop;

DELIMITER //

CREATE procedure while\_loop()

BEGIN

set @i = 0;

while(@i < 10) Do

set @i = @i+1;

update nums\_To, call\_forwarding

set nums\_To.forwarded = call\_forwarding.To

where nums\_To.forwarded = call\_forwarding.From;

end while;

END //

call while\_loop();

update call\_forwarding, nums\_To

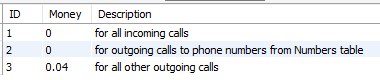
set call\_forwarding.To = nums\_To.forwarded

where call\_forwarding.From = nums\_To.From;



**rates**

Rates. It's clear



**call\_logs**

call\_logs - information about each call. This table one of the most important, because we can come to different conclusions from this date, for example, count charges of each call.

We have call forwarding. So let's change column To using call\_forwarding table.

update call\_logs, call\_forwarding

set call\_logs.To = call\_forwarding.To

where call\_logs.To = call\_forwarding.From;

In each call record we have client UID, who made call, or client UID, who get call. Find both of them for easier charges computation.

DROP TEMPORARY TABLE IF EXISTS new\_call\_logs;

CREATE TEMPORARY TABLE new\_call\_logs

select calls\_out.call\_id, calls\_out.From, calls\_out.To,calls\_out.UID as UID\_From, numbers.UID as UID\_To, calls\_out.Timestamp\_start, calls\_out.Timestamp\_end from

(select \* from call\_logs where call\_dir="out") as calls\_out left join numbers

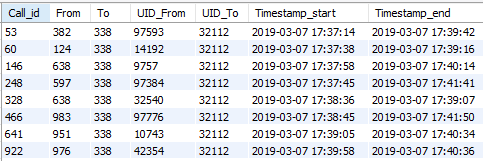
on calls\_out.To = numbers.Phone\_number

union

select calls\_in.call\_id, calls\_in.From, calls\_in.To, numbers.UID as UID\_From, calls\_in.UID as UID\_To, calls\_in.Timestamp\_start, calls\_in.Timestamp\_end from

(select \* from call\_logs where call\_dir="in") as calls\_in left join numbers

on calls\_in.From = numbers.Phone\_number;



Compute charges. According to rates, charges get from outcome calls, which don't include in Numbers table list.

DROP TEMPORARY TABLE IF EXISTS charges;

CREATE TEMPORARY TABLE charges

select new\_call\_logs.Call\_id, new\_call\_logs.From, new\_call\_logs.To, new\_call\_logs.UID\_From, new\_call\_logs.UID\_to,

((UNIX\_TIMESTAMP(Timestamp\_end) - UNIX\_TIMESTAMP(Timestamp\_start))\*0.04) as charge from new\_call\_logs;

update charges, numbers

set charges.charge = 0

where charges.UID\_To = numbers.UID;

Total charges

select sum(charge) from charges;

2836.6

Let's find the most active users. The most active users make most of calls (number). Also we can take duration of calls as measure.

Clients, who have most of income calls.

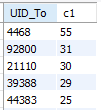
DROP TEMPORARY TABLE IF EXISTS in\_UID;

CREATE TEMPORARY TABLE in\_UID

select UID\_To, count(UID\_To) as c1 from new\_call\_logs

group by UID\_To;

select \* from in\_UID order by c1 desc Limit 5;



Clients, who make most of outcome calls.

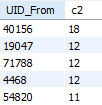
DROP TEMPORARY TABLE IF EXISTS out\_UID;

CREATE TEMPORARY TABLE out\_UID

select UID\_From, count(UID\_From) as c2 from new\_call\_logs

group by UID\_From;

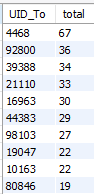
select \* from out\_UID order by c2 desc Limit 5;



Let's combine this results and get top-10 of the most active clients.

select in\_UID.UID\_To, (in\_UID.c1 + out\_UID.c2) as total from in\_UID , out\_UID

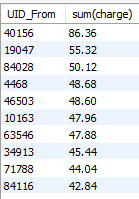
where in\_UID.UID\_To = out\_UID.UID\_From order by total desc limit 10 ;



Top-10 clients with highest charges.

select UID\_From, sum(charge) from charges where UID\_From is not null

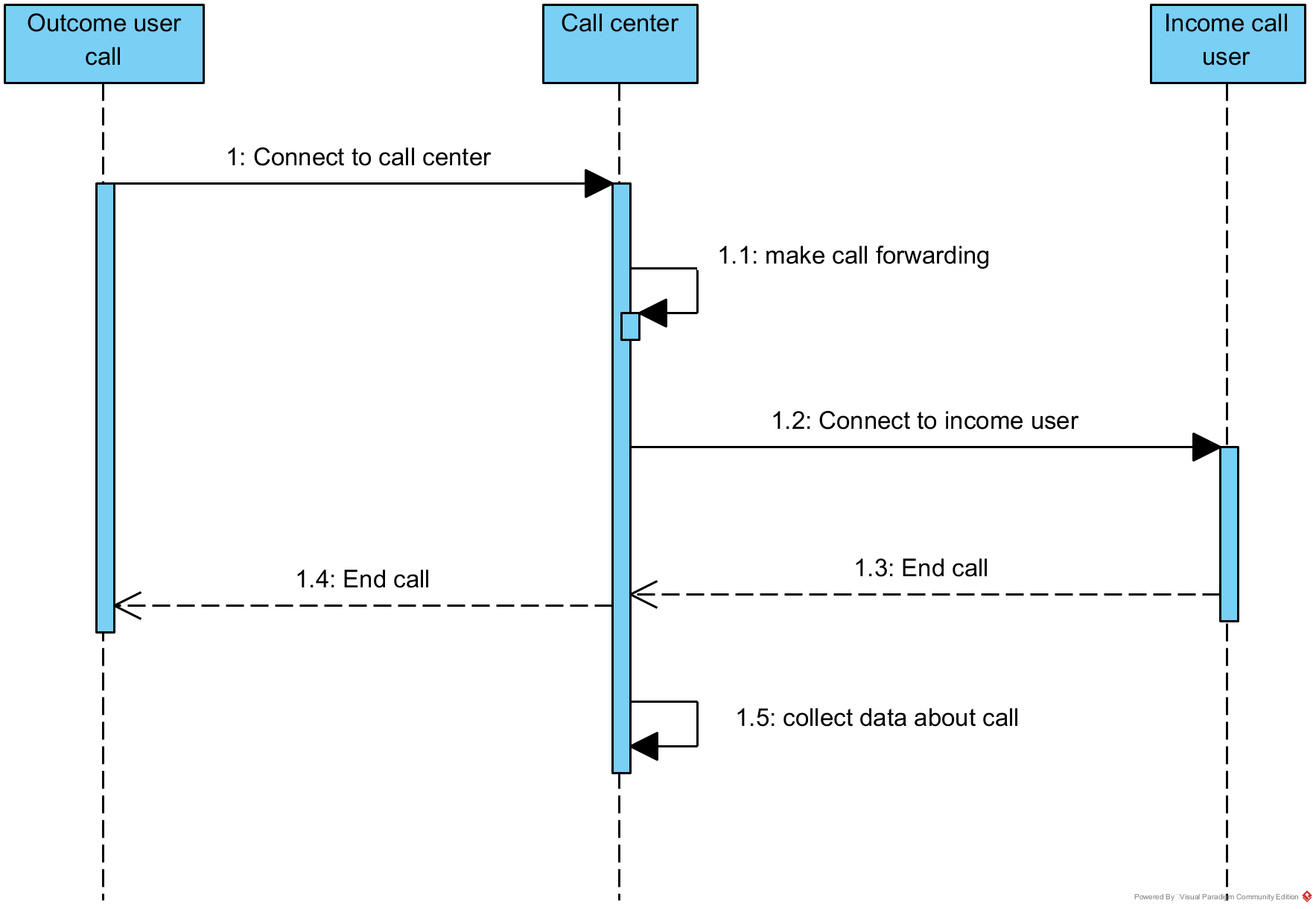
group by UID\_From order by sum(charge) desc limit 10;



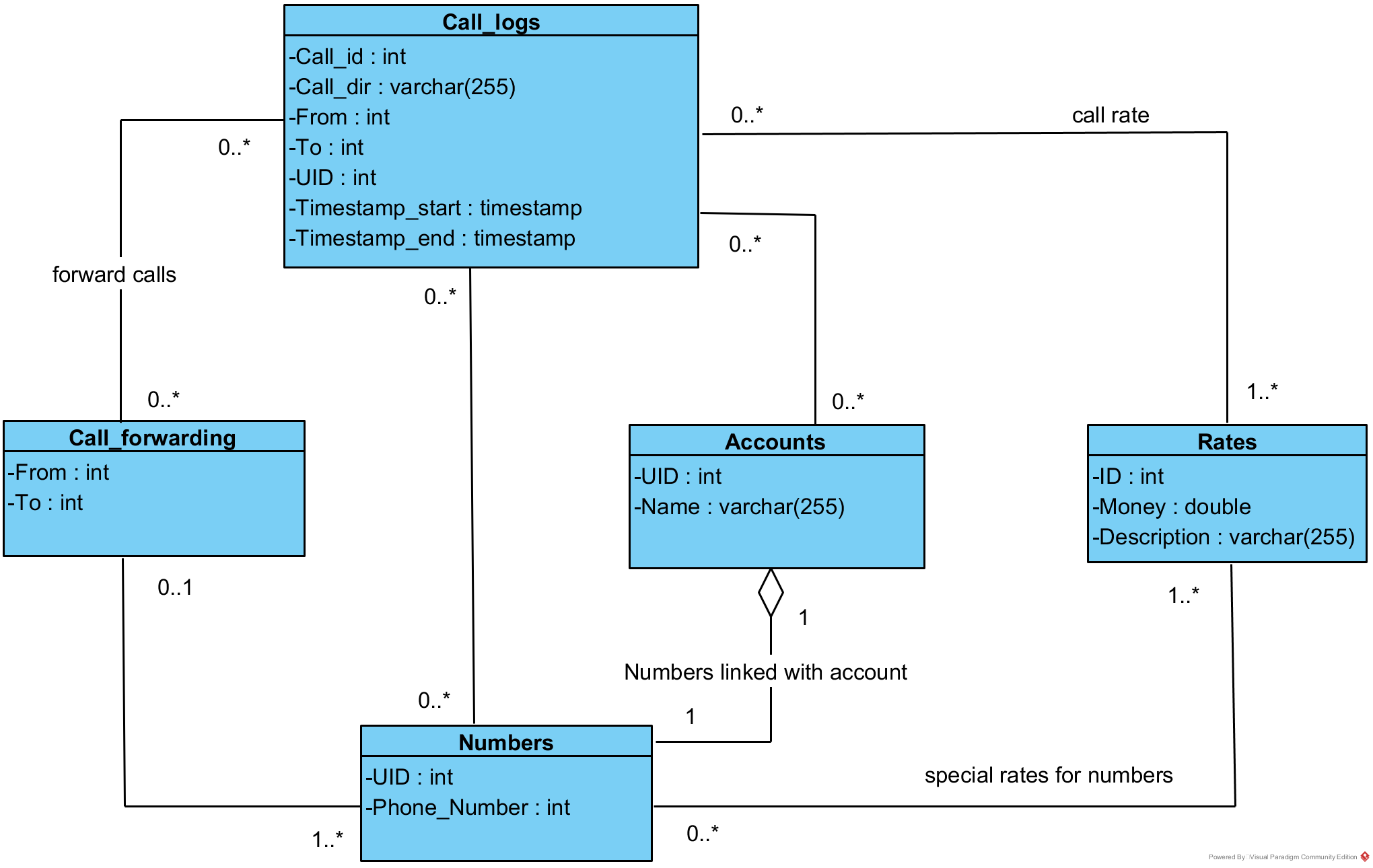
**User stories**

1) As client, I want to make calls and get calls to/from other clients to communicate with them.  
2) As client, I want to forward calls to my number to another number to centralize my calls.  
3) As client, I want have convenient rate to reduce my charges.  
4) As System Analyst, I want to get information about calls to analyze it to make a work of call center better.  
5) As administrator of call center, I want system analysts, engineers and developers work properly to get fault tolerant system.

**Sequence diagram**

****

**Class diagram**

****