Naman Ahuja
2019101042
Question3:
To design this data store, we use a relational database management system. An RDBMS will be helpful as the metrics for our APIs are well-defined and I do not foresee a clear-cut reason for adding extra columns to the tables. In trade for this rigidity of table columns, we will get very fast responses to our queries, which will be helpful as our database will grow by an average of a million entries a day.

This vertical scalability of our database will allow for easy on-the-fly analytics along with making expressive queries using the SQL language.

Seeing the nature of the example queries, I suspect we will be querying our databases based mostly on the time parameter of when the API was called. Using this insight, we can further speed up our queries by selecting a database that indexes the entries based on time, such as TimeDB.

The data store can be broadly divided into 3 different tables.

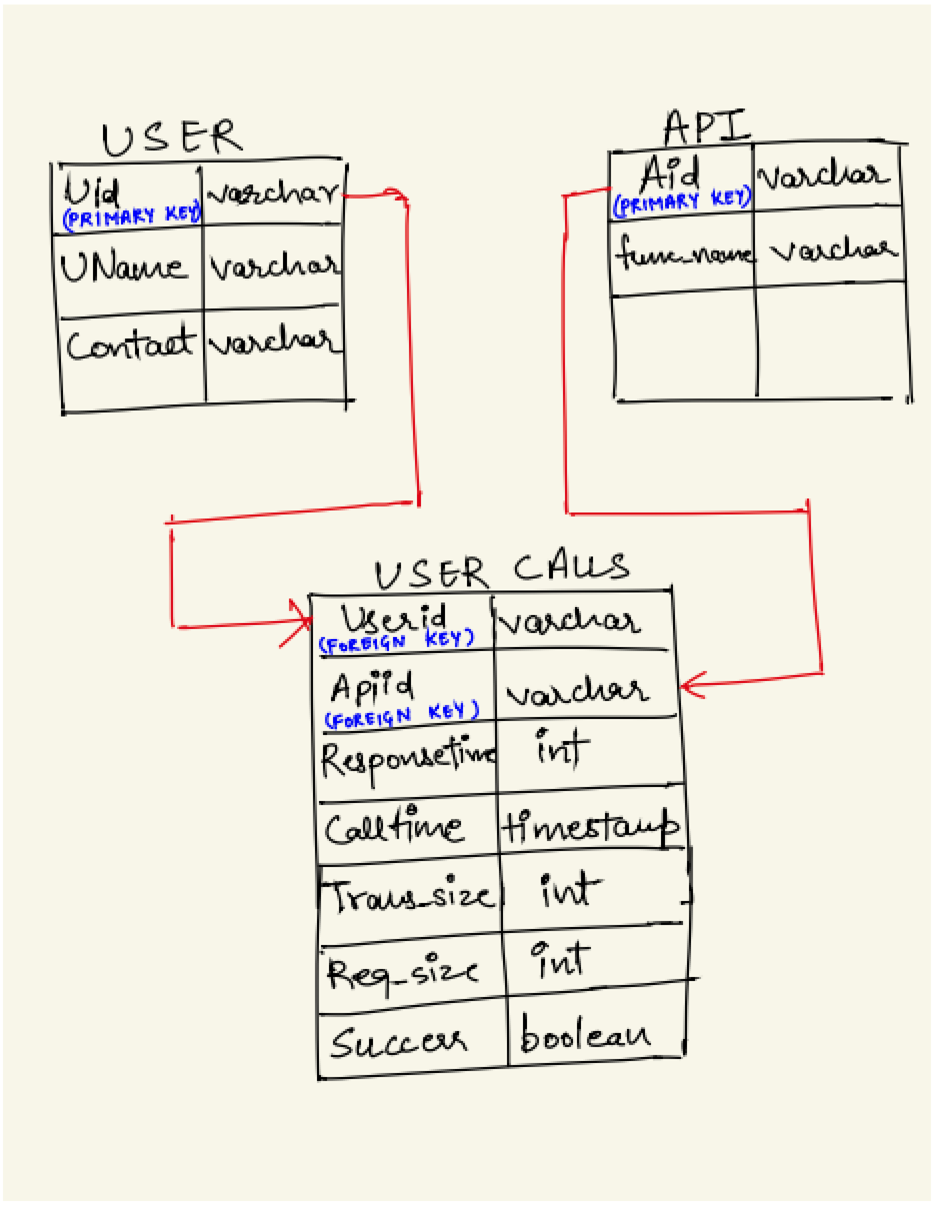
- User_Table: This table stores all the relevant information about a user. Some columns that we will be needing in this system are
  - uid (Primary Key): The id can be used to uniquely identify all users and hence cannot have a NULL value. Data Type: varchar(255)
  - name: name of the user. Datatype: varchar(255)

- API: This table contains the set of API's that the users can query. It contains all the relevant information about what the API is used for. The columns of this table are as follows:
  - aid(Primary Key): Used to uniquely identify every API
  - func_name(secondary key): the name of the API

- USER_calls: This table stores all the relevant information about the activity in a particular day. Whenever an API call is made by any user, the information of the API and the user along with the metadata of the API call made is stored in this table. This table has the following columns(for any 1 API call made by any user):
  - Userid (Foreign Key from user_data): user id of the user who made the API call
  - Apiid(Foreign key from column_table): id of the API the user called.
  - Call_time: the time at which the request was made
  - response_time : the response time for the api call.
  - trans_size(in Kb): Size of the data the user requested in Kilobytes
  - req_size (in Kb): Size of the data the API transmits in Kilobytes.

- success: whether the API call was successfully made or not.

The Schema of the Datastore is as follows:



Queries:

For SQL the sample queries provided in the questions can be solved as follows:

Get the API with the maximum average response time across the users

```
Query : SELECT AVG(response_time) AS A FROM USER_calls GROUP BY api_id
            ORDER BY A DESC LIMIT 1;
```

- Get the API with the maximum average response time for each user.

```
Query: SELECT AVG(response_time) AS A FROM USER_CALLS GROUPBY userid, apiid
          ORDER BY A DESC LIMIT 1;
```

- Get the error percentage of each API in buckets of 1 hour for 24 hours.

```
Query: SELECT
```