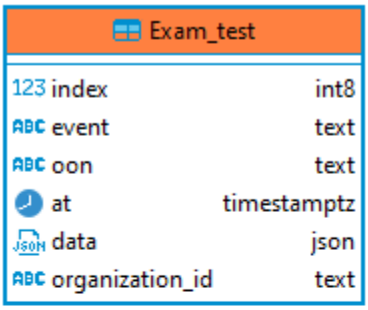
Boston Makmur Gemilang
Data Engineer Test

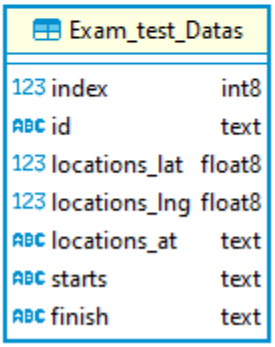
Summary: After we have successfully migrated our Json files to PostgreSQL server, we will split and connect between each table to creates relationship.

Details:
Database : PostgreSQL
Server : Localhost

Explanations:
The exam test & Exam_test_Datas are the result of the json files we have migrated recently from python.



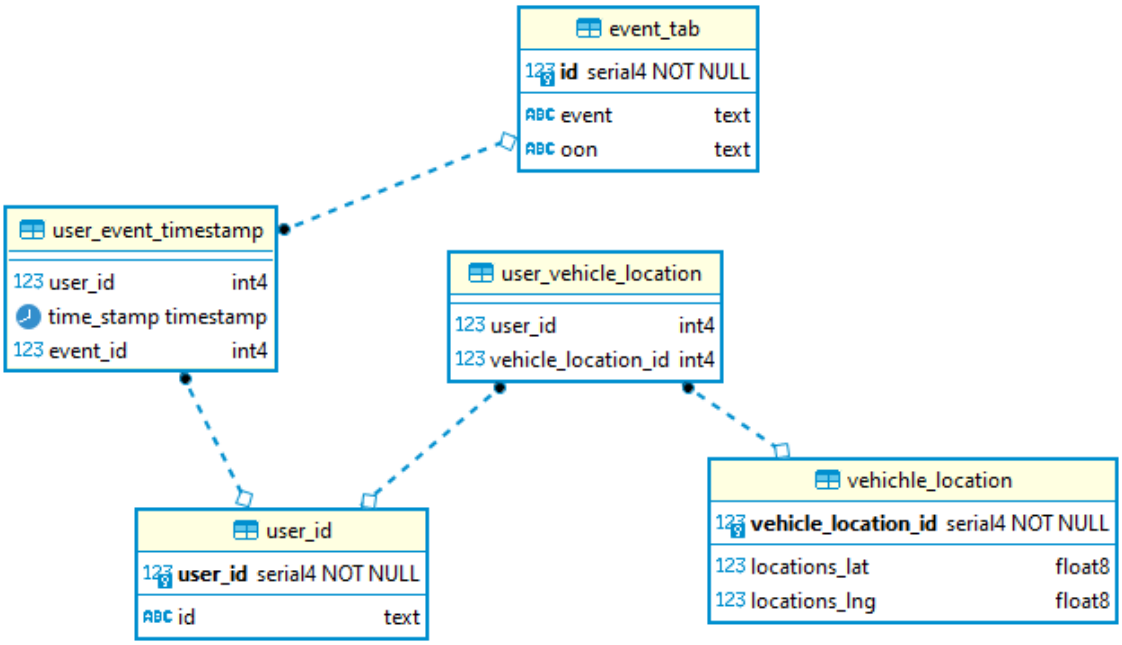
Exam_test table:
- index : Automatically generated by python
- event: Indicates the type of event
- oon: Indicates on which entity event occurred
- at:timestamp of the entity
- data: This is the list of nested JSON (we parsed it into - another tables on Exam_test_Datas)
- organization_id : Identifies the organization event belongs to



Exam_test_Datas table:
- index : Automatically generated by python
- id : The user id of the vehicle
-locations_lat : The latitude of the vehicle
-locations_lng : The longitude of the vehicle
-locations_at : The timestamp
-starts: Timestamp
-finish: Timestamp

Problems: as we can see, both of the tables contains a lot of information, and it has to be normalized and breakdown to avoid the anomalies or redundancy in the upcoming times.

Solution: creates a separates table to store each unique attributes as a primary key



Explanations : we have divided our table into a more approachable design, now that each unique attributes has primary key.

User_id table
- User_id = Unique Key/ Primary Key
- Id = The id of the user

vehicle_location table
- vehicle_location_id = Unique Key/Primary Key
- locations_lat = The latitude of the vehicle
- locations_lng = The longitude of the vehicle

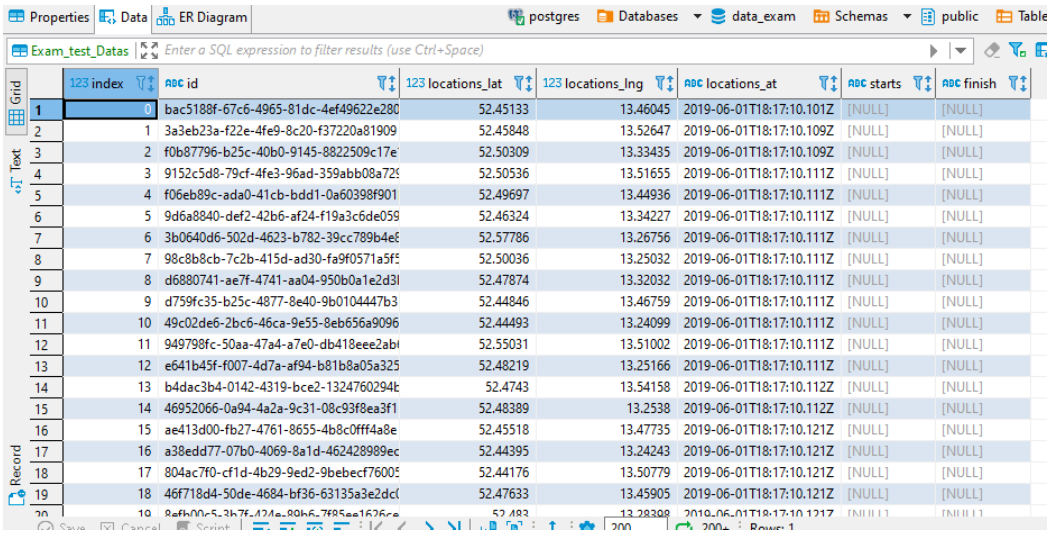
user_vehicle_location table: serves as an intermediate table for both user and vehicle tables
- user_id = Foreign key of user_id
- vehicle_location_id = Foreign key of vehicle_id

Event_tab table
- id = Unique Key/ Primary key
- event: Indicates the type of event
- oon: Indicates on which entity event occurred

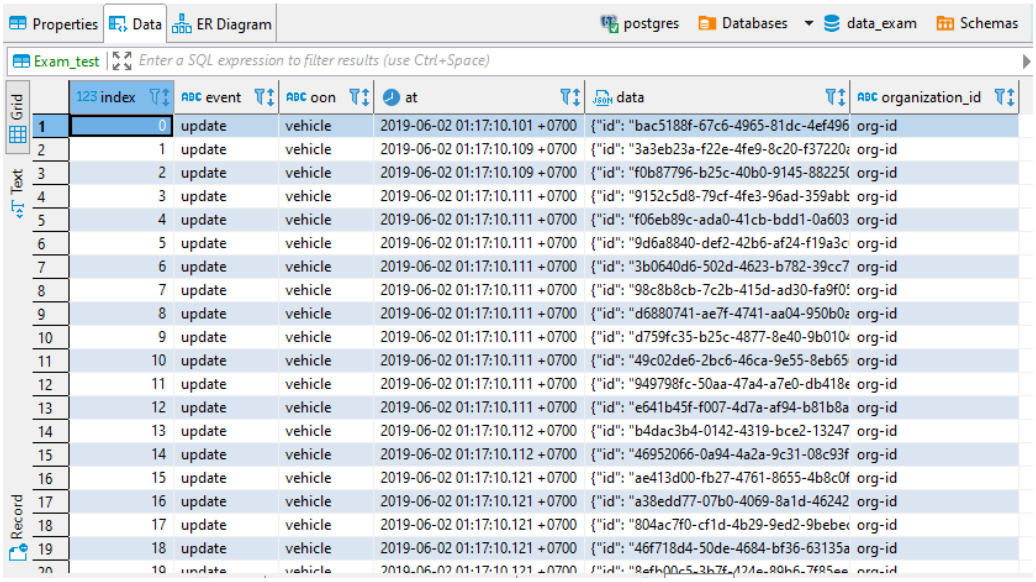
user_event_timestamp : user event timestamp tables uses both user_id & timestamp to indicates precisely when one's event take times
- user_id = Foreign key of user_id
- timestamp = timestamp
- event_id = Foreign key of event_id

Before Normalization:

Exam_test_Datas table

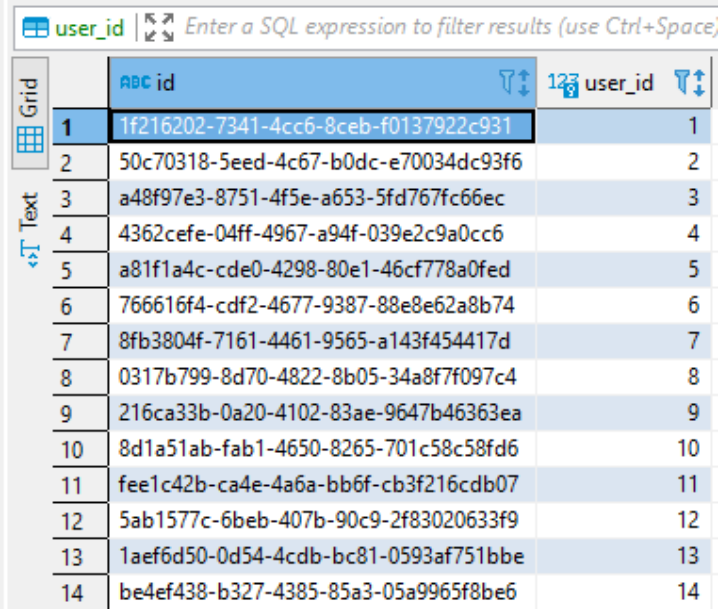


Exam_test table:



After Normalization:

User_id table

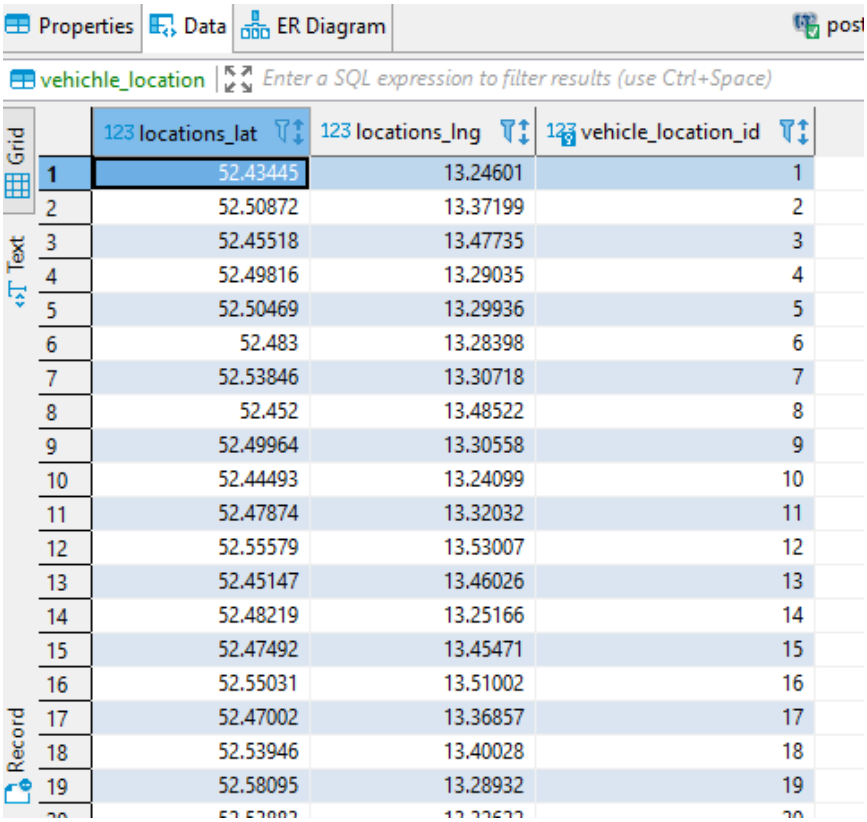


user_id | Enter a SQL expression to filter results (use Ctrl+Space)

Grid / Text

	id	user_id
1	1f216202-7341-4cc6-8ceb-f0137922c931	1
2	50c70318-5eed-4c67-b0dc-e70034dc93f6	2
3	a48f97e3-8751-4f5e-a653-5fd767fc66ec	3
4	4362cefe-04ff-4967-a94f-039e2c9a0cc6	4
5	a81f1a4c-cde0-4298-80e1-46cf778a0fed	5
6	766616f4-cdf2-4677-9387-88e8e62a8b74	6
7	8fb3804f-7161-4461-9565-a143f454417d	7
8	0317b799-8d70-4822-8b05-34a8f7f097c4	8
9	216ca33b-0a20-4102-83ae-9647b46363ea	9
10	8d1a51ab-fab1-4650-8265-701c58c58fd6	10
11	fee1c42b-ca4e-4a6a-bb6f-cb3f216cdb07	11
12	5ab1577c-6beb-407b-90c9-2f83020633f9	12
13	1aef6d50-0d54-4cdb-bc81-0593af751bbe	13
14	be4ef438-b327-4385-85a3-05a9965f8be6	14

Vehicle location table



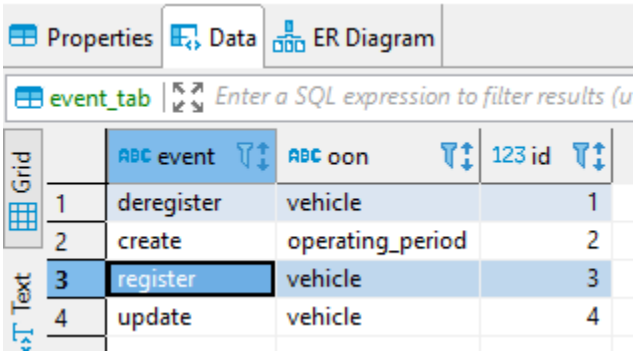
Properties | Data | ER Diagram | post

vehichle_location | Enter a SQL expression to filter results (use Ctrl+Space)

Grid / Text / Record

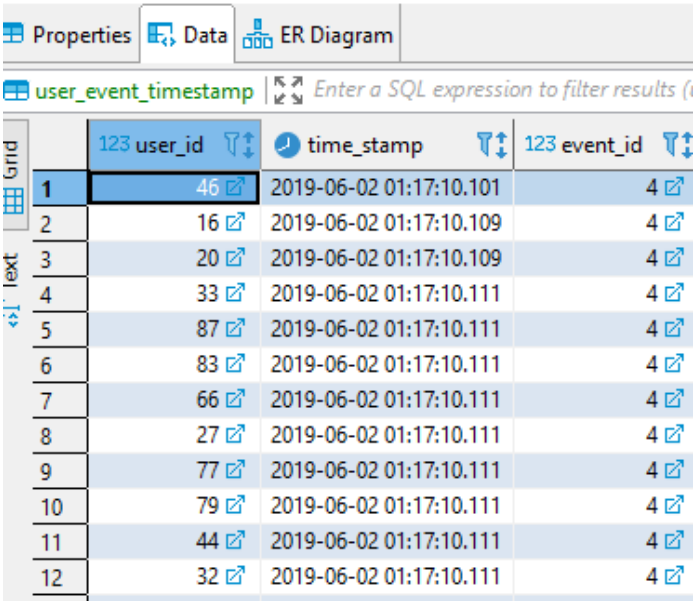
	locations_lat	locations_lng	vehicle_location_id
1	52.43445	13.24601	1
2	52.50872	13.37199	2
3	52.45518	13.47735	3
4	52.49816	13.29035	4
5	52.50469	13.29936	5
6	52.483	13.28398	6
7	52.53846	13.30718	7
8	52.452	13.48522	8
9	52.49964	13.30558	9
10	52.44493	13.24099	10
11	52.47874	13.32032	11
12	52.55579	13.53007	12
13	52.45147	13.46026	13
14	52.48219	13.25166	14
15	52.47492	13.45471	15
16	52.55031	13.51002	16
17	52.47002	13.36857	17
18	52.53946	13.40028	18
19	52.58095	13.28932	19

Event_tab table



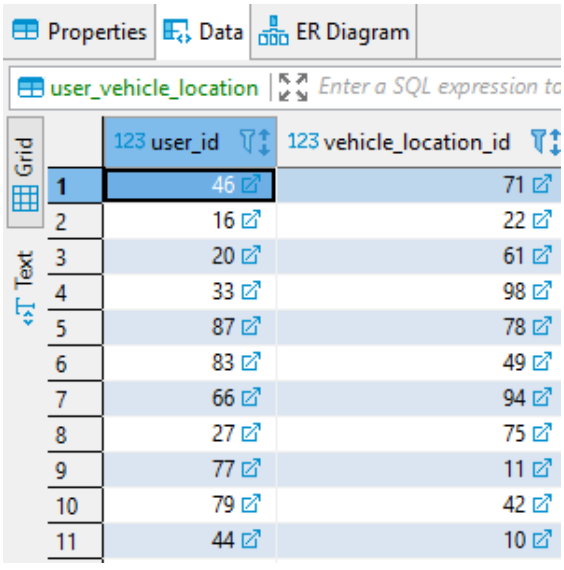
	event	oon	id
1	deregister	vehicle	1
2	create	operating_period	2
3	register	vehicle	3
4	update	vehicle	4

User_event_timestamp table



	user_id	time_stamp	event_id
1	46	2019-06-02 01:17:10.101	4
2	16	2019-06-02 01:17:10.109	4
3	20	2019-06-02 01:17:10.109	4
4	33	2019-06-02 01:17:10.111	4
5	87	2019-06-02 01:17:10.111	4
6	83	2019-06-02 01:17:10.111	4
7	66	2019-06-02 01:17:10.111	4
8	27	2019-06-02 01:17:10.111	4
9	77	2019-06-02 01:17:10.111	4
10	79	2019-06-02 01:17:10.111	4
11	44	2019-06-02 01:17:10.111	4
12	32	2019-06-02 01:17:10.111	4

User_vehicle_location table



	user_id	vehicle_location_id
1	46	71
2	16	22
3	20	61
4	33	98
5	87	78
6	83	49
7	66	94
8	27	75
9	77	11
10	79	42
11	44	10

Thank you for your attentions, let’s move on to the visualization part.