Bixi dataset exploration

Group: 20 Random

Tung Leu 40025151 ; Khalil Nijaoui 40092653
Ying Fang 26897657; Arshia Hamidi 40068250

BIXI montreal (public bicycle sharing system)

Dataset

-Dataset: BIXI Montreal (public bicycle sharing system)

-Source : kaggle: https://www.kaggle.com/aubertsigouin/biximtl

- **Topic**: Bixi- Movements history

-Size: 1.15GB (23 csv files)

-Records: 22.2 million record

Why this dataset

The dataset contains the details of the travels made via the BIXI Montreal service bike network in CSV format. Such information would be helpful to determine the location of new BIXI station, to allocate more bikes or even to construct bike paths(attractions).

Systems used

SQL: PostgreSQL

NoSQL: MongoDB

Exploring data & comparing systems execution time

	SQL queries duration	MongoDb queries duration
1 Average bixi ride (duration) per year	**4 secs 638 msec**	1 min 19 secs
2 Average bixi ride(duration) per month	**4 secs 555 msec**	1 min 16 secs
3 Longest ride duration per year	**4 secs 688 msec**	1 min 16 secs
4 Most month with no membership	**2 secs 694 msec**	43 secs 617 msec

Exploring data & comparing systems execution time

	SQL queries duration	MongoDb queries duration
5 Most busy day	**4 secs 478 msec**	55 secs 255 msec
6 Most favorite path	**3 secs 576 msec**	1 min 119 msec
7 Most busy month	**4 secs 140 msec**	41 secs 522 msec
8 Most busy start station	**9 secs 187 msec**	51 secs 097 msec

Exploring data & comparing systems execution time

	SQL queries duration	MongoDb queries duration
9 Most busy end station	**8 secs 941 msec**	41 secs 398 msec
10 Less busy station (less visited)	**9 secs 327 msec**	52 secs 656 msec

SQL Queries

1 Average bixi ride (duration) per year

```
SELECT AVG(duration_sec) AS AVG_DUR, date_trunc('year', start_date)
From od
GROUP BY date_trunc('year', start_date)
```

2 Average bixi ride(duration) per month

```
SELECT AVG(duration_sec) AS AVG_DUR, date_trunc('month', start_date)
From od
GROUP BY date_trunc('month', start_date)
```

3 Longest ride duration per year

```
SELECT date_trunc('year', start_date) as year, MAX(duration_sec)
FROM OD
GROUP BY year
select date_trunc('year', start_date) as year, duration_sec
from OD, (select date_trunc('year', start_date) as y, MAX(duration_sec)as
maxdur
from OD group by date_trunc('year', start_date)) o
where o.maxdur=OD.duration_sec
```

4 Most month with no membership

```
SELECT date_trunc('month', start_date) as month, count(*)
FROM OD
WHERE is_member = 0
GROUP BY month
ORDER by count(*)
Desc limit 1;
```

5 Most busy day

```
select date_trunc( 'day', start_date ) as day, count(*)
from OD
group by day
order by count(*)
desc limit 1;
```

6 Most favorite path

```
select count(*),start_station,end_station_code
from station join OD on station.Code = OD.end_station_code
group by start_station,end_station_code
order by count(*) desc
```

7 Most busy month

```
SELECT date_trunc('month', start_date) as month, count(*)
FROM OD
GROUP BY month
ORDER by count(*)
```

8 Most busy start station

```
select count(*),start_station_code,station.name
from station join OD on station.Code = OD.start_station_code
group by start_station_code,station.name
order by count(*) desc
```

9 Most busy end station

```
select count(*),end_station_code,station.name
from station join OD on station.Code = OD.end_station_code
group by end_station_code,station.name
order by count(*) desc
```

10 Less busy station (less visited)

```
select count(*),end_station_code,station.name
from station join OD on station.Code = OD.end_station_code
group by end_station_code,station.name
order by count(*)
```

MongoDB queries

1 Average bixi ride (duration) per year

```
db.OD.aggregate([
                {$group: {
                                _id: {year : { $substr : ["$start_date", 0, 4
]}
                                },avgride: {$avg:"$duration_sec"}
                }},{$sort:{avgride:-1}}])
```

2 Average bixi ride(duration) per month

```
db.OD.aggregate([
                {$group: {_id: {month : { $substr : ["$start_date", 5, 2
]}},avgride: {$avg:"$duration_sec"}
                }},{$sort:{avgride:-1}}])
```

3 Longest ride duration per year

```
db.OD.aggregate([
                {$group: {_id: {year : { $substr : ["$start_date", 0, 4 ]}
                                },longest_ride: {$max:"$duration_sec"}
                }}])
```

4 Most month with no membership

```
db.OD.aggregate([{$match: {is_member: 0}},
{$group: {_id: {month : { $substr : ["$start_date", 5, 2 ]}
},count: {$sum:1}}},{$sort:{count:-1}}])
```

5 Most busy day

```
db.OD.aggregate([
                {$group: {_id: {day : { $substr : ["$start_date", 0, 10 ]}},count:
{$sum:1}}},{$sort:{count:-1}}])
```

6 Most favorite path

```
db.OD.aggregate([
                {$group: {_id: {start : "$start_station_code",
end : "$end_station_code"},count: {$sum:1}}},{$sort:{count:-
1}}],{allowDiskUse: true})
```

7 Most busy month

```
db.OD.aggregate([{$group: {_id: {month : { $substr : ["$start_date", 5, 2
]}},count: {$sum:1}}},{$sort:{count:-1}}])
```

8 Most busy start station

```
db.OD.aggregate([{$group:{_id:"$start_station_code", count:
{$sum:1},avgride: {$avg:"$duration_sec"}}},{$lookup: {from:  "station",
localField: "_id", foreignField: "code",as: "name"}
  },{$sort:{count:-1}}],{allowDiskUse: true})
```

9 Most busy end station

```
db.OD.aggregate([{$group: {_id:"$end_station_code", count:
{$sum:1},avgride: {$avg:"$duration_sec"}}},
{ $lookup: { from:  "station",localField: "_id", foreignField: "code",
   as: "name" }  },{$sort:{count:-1}}],{allowDiskUse: true})
```

10 Less busy station (less visited)

```
 db.OD.aggregate([{$group: {_id:"$start_station_code", count:
{$sum:1},avgride: {$avg:"$duration_sec"}}},
{$lookup: { from:  "station", localField: "_id",foreignField: "code", as: "name"}
  },{$sort:{count:1}}],{allowDiskUse: true})
```

Queries results

1. **Average bixi ride (duration) per year**

average duration in seconds:	year
803.99	2018
821.65	2019
837.45	2017
837.59	2016
936.33	2020

2. **Average bixi ride (duration) per year**

month	average duration in seconds:
07	870.74
06	866.26
05	864.88
08	854.01
09	801.21
04	797.49
10	727.71
11	714.82

3 Longest ride duration per year

longest duration:	year
7199	2016
7199	2017
7199	2018
7199	2019
7199	2020

4 Most month with no membership

month	number of rides without membership
07	899387
08	883219
06	734949
09	614032
05	590979
10	225917
04	140959
11	47425

5. Most busy day

Day	number of rides
2019-05-26	42878
2019-07-03	41063
2019-07-10	40652
2019-07-18	40612
2017-07-30	40441

6 Most favorite path

start station	end station	number of rides
Metro Jean-Drapeau (Chemin Macdonald)	Metro Jean-Drapeau (Chemin Macdonald)	19511
de la Commune / Place Jacques-Cartier	de la Commune / Place Jacques-Cartier	14377
Metro Laurier (Rivard / Laurier)	Marquette / Laurier	10859
de la Commune / St-Sulpice	de la Commune / St-Sulpice	10748
MÃ©tro Pie-IX (Pierre-de-Coubertin / Pie-IX)	Desjardins / Ontario	9421

7 Most busy month

month	number of rides
07	4294745
08	4159406
06	3761931
09	3581575
05	3140251
10	1844432
04	935795
11	440794

8 Most busy start station

start station	number of rides
Mackay /de Maisonneuve (Sud)	191388
Métro Mont-Royal (Rivard / du Mont-Royal)	183307
Métro Laurier (Rivard / Laurier)	180170
de Maisonneuve / Stanley	150937
du Mont-Royal / Clark	150672

9 Most busy end station

end station	number of rides
Berri / de Maisonneuve	211126
Mackay /de Maisonneuve (Sud)	190936
Métro St-Laurent (de Maisonneuve / St-Laurent)	188779
de la Commune / Place Jacques-Cartier	182687
Métro Mont-Royal (Rivard / du Mont-Royal)	172418

10 Less busy station (less visited)

station	number of rides	average ride
MTL-ECO5.1-01	3	238
Centre des loisirs (Tassé / Grenet)	13	1859
Messier / St-Joseph	14	667
Ateliers municipaux de St-Laurent (Cavendish / Poirier)	18	1989
Place Rodolphe-Rousseau (Gohier / Édouard-Laurin)	30	999

SQL indexing

the planner is concerned with minimising the total cost of the query. With databases, the cost of I/O typically dominates. For that reason, "count(*) without any predicate" queries will only use an index-only scan if the index is significantly smaller than its table. This typically only happens when the table's row width is much wider than some indexes.

rider who are members and started in station: 6100: No indexing

select * from OD

where OD.is_member=1 and start_station_code =6100 limit 10

2 secs 373 msec

rider who are members and started in station: 6100: index on is_member and start_station_code

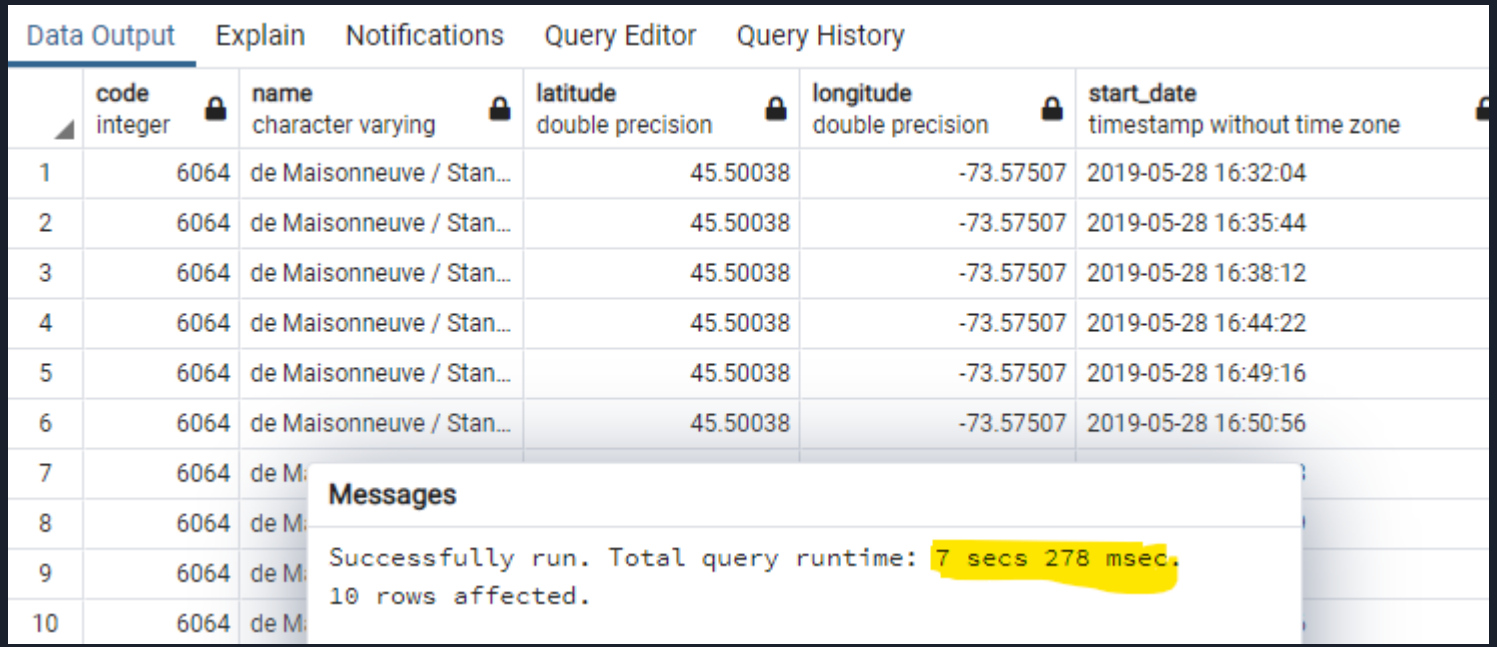
select * from OD

where OD.is_member=1 and start_station_code =6100 limit 10

194 msec

finding information about rides that started in de Maisonneuve / Stanley station.

Without indexes



Data Output | Explain | Notifications | Query Editor | Query History

	code integer	name character varying	latitude double precision	longitude double precision	start_date timestamp without time zone
1	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:32:04
2	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:35:44
3	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:38:12
4	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:44:22
5	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:49:16
6	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:50:56
7	6064	de M			
8	6064	de M			
9	6064	de M			
10	6064	de M			

Messages

```
Successfully run. Total query runtime: 7 secs 278 msec.
10 rows affected.
```

finding information about rides that started in de Maisonneuve / Stanley station.

With indexes (on code and name)

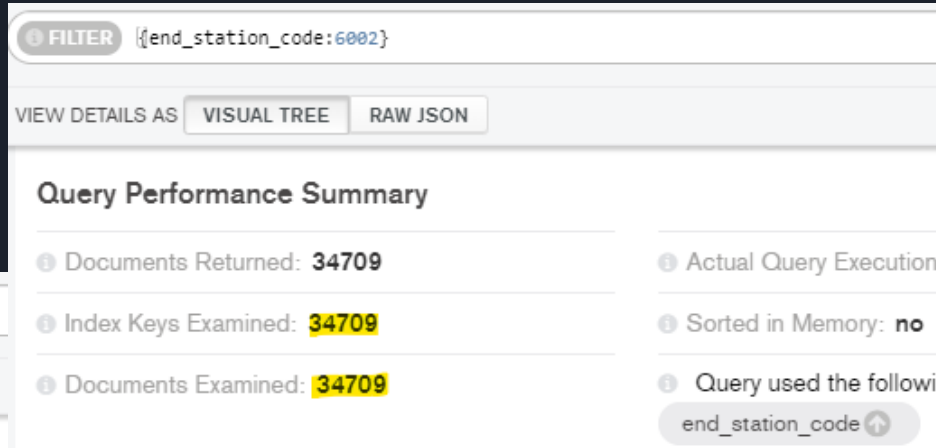
	code integer	name character varying	latitude double precision	longitude double precision	start_date timestamp without time zone	start_station_code integer
1	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:32:04	6064
2	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:35:44	6064
3	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:38:12	6064
4	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:44:22	6064
5	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:49:16	6064
6	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:50:56	6064
7	6064	de Maisonneuve / Stan...	45.50038	-73.57507	2019-05-28 16:51:23	6064
8	6064	de Maisonneuve / Stan...				6064
9	6064	de Maisonneuve / Stan...				6064
10	6064	de Maisonneuve / Stan...				6064

Messages

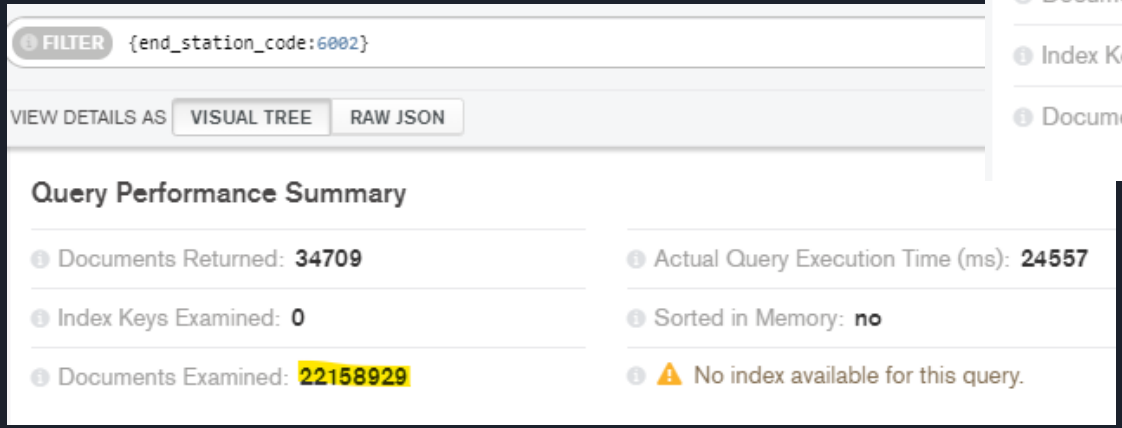
```
Successfully run. Total query runtime: 361 msec.
10 rows affected.
```

MongoDB indexing

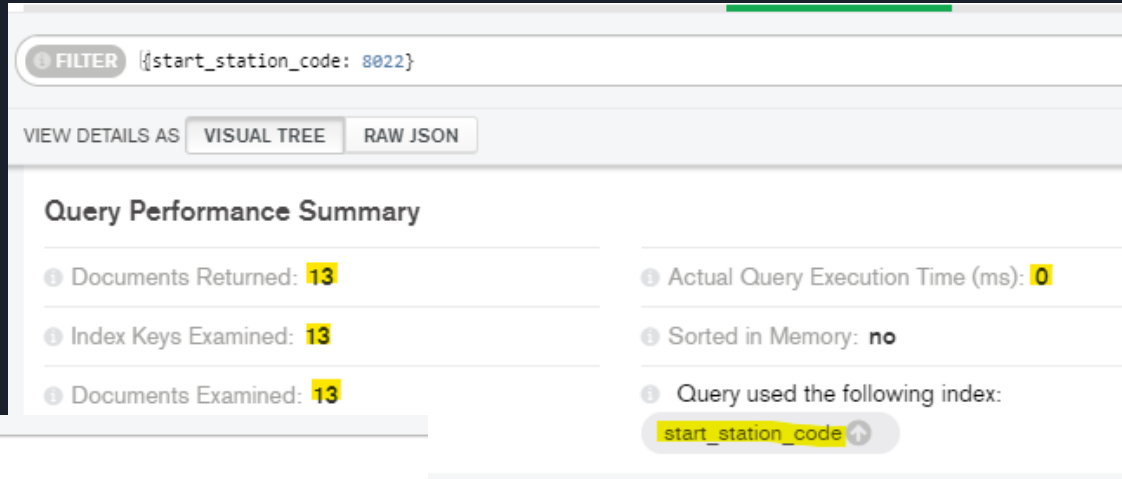
with indexing:



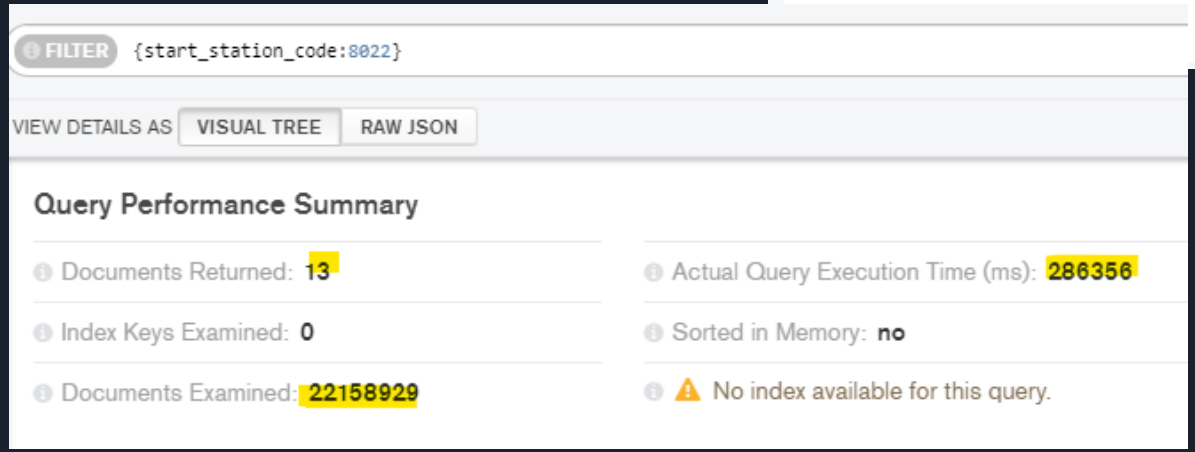
without indexing:



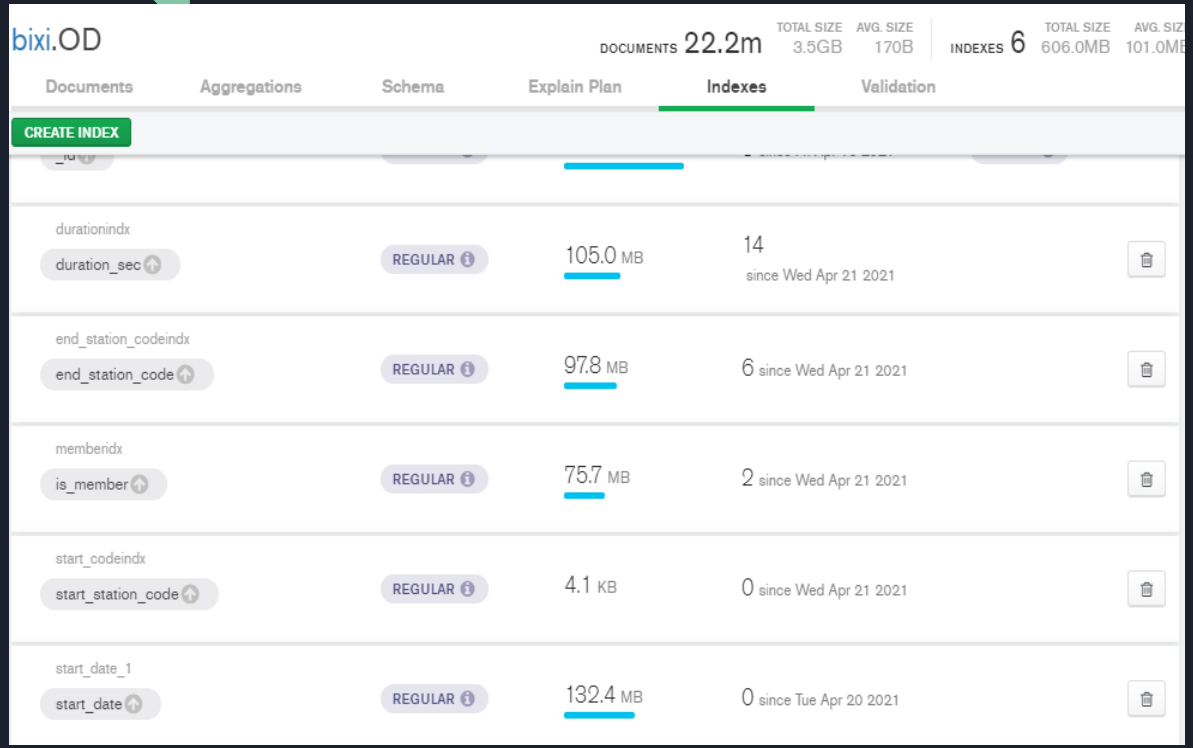
Finding information about a rarely used station(indexing)



Finding information about a rarely used station(No indexing)



Indexing price



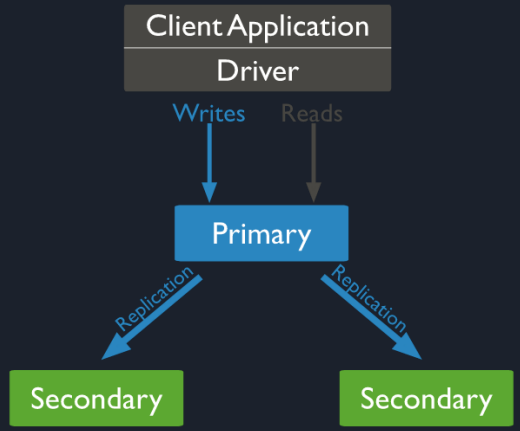
- costs for index creation
- storage
- maintenance

Heatmap of the bixi network

Consistency and Availability in MongoDB

Consistency by default since MongoDB is a single master system and all reads go to primary by default

By using the replication technique, MongoDB ensures high data availability. MongoDB creates a group of replicas instances that maintain the same data set. Once the primary node goes down, the secondaries will hold an election to choose one to become the primary to become available again



The balance

However by using replications, there are rollbacks during replica set failover. The rollback happens when there is a write operation in the primary that stepping down and the write is not yet synchronized with the replicas. And because of that, consistency is sacrificed for sake of availability

Reference: https://docs.mongodb.com/manual/core/replica-set-rollbacks/

https://www.kaggle.com/aubertsigouin/bixi-network-analysis

https://bixi.com/en/page-27

https://docs.mongodb.com/manual/replication/

Questions

Thank you