DATABASE MGMT & SYS IMPLEM Term Project

Jordan Tippett & Adam Craig

Option 1

System Selection: Google Bigquery & Cloud Sql for Postgresql

Postgres:

- Row based database
- Highly customizable
- Indexes
- Scalability issues?

Google Bigquery

- Columnar database
- Not very customizable
- Supposedly very scalable

Goals / Measuring Performance

Goals

- Performance on selection of complex data types
- No index vs index
- Columnar databases vs Row databases
- Scalability

Measuring Performance

- Average of 5 runs drop fastest and slowest
- We used elapsed time instead of execution time because Bigquery didn’t have a clear way to report execution time (another example of limited options)
    - Using execution time for postgres would have resulted in an unfair comparison

Experiment 1: INDEX VS NO INDEX

Purpose: Test whether an index on a complex data type increases query performance

Expectation: Postgresql will have better performance because of the index

Index on complex.unique1 in postgresql

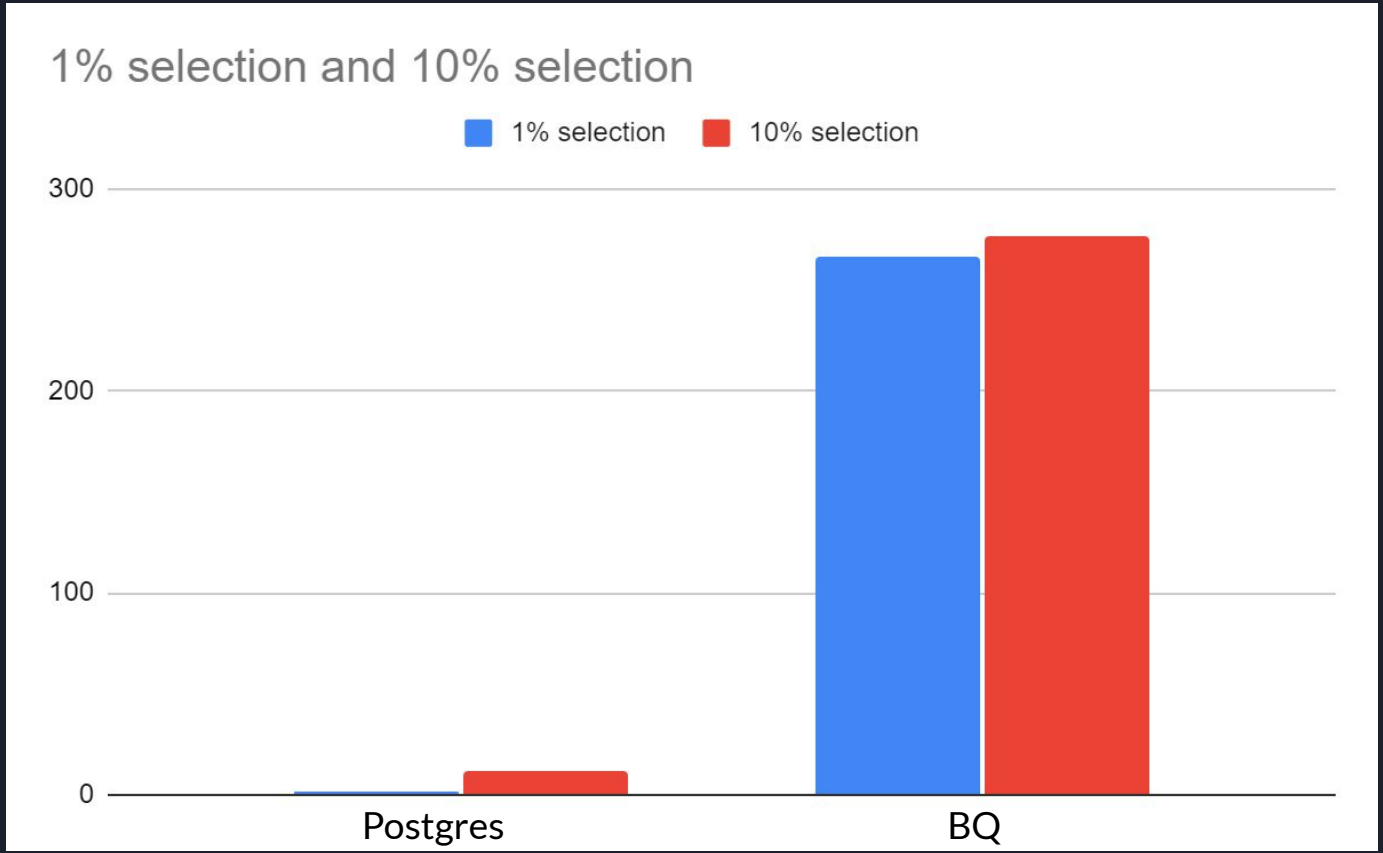
Query 1: 1 % Selectivity

```
SELECT COUNT(complex.unique1)
FROM 100Ktup
WHERE complex.unique1 BETWEEN 0 AND 999
```

Query 2: 10 % selectivity

```
SELECT COUNT(complex.unique1)
FROM 100Ktup
WHERE complex.unique1 BETWEEN 792 AND 10791
```

100k Tuples


1% selection and 10% selection
1% selection
10% selection
300
200
100
0
Postgres
BQ

Experiment 2: COLUMNAR VS ROW DATABASE

Purpose: Test how the number of attributes selected impacts query performance

Expectation: Postgresql will perform better when selecting all attributes and Google Bigquery will perform better when selecting a single attribute

Query 1: 1 % Selectivity

```
SELECT *
FROM 100Ktup
WHERE complex.unique1 BETWEEN 0 AND 999
```

Query 2: 10 % selectivity

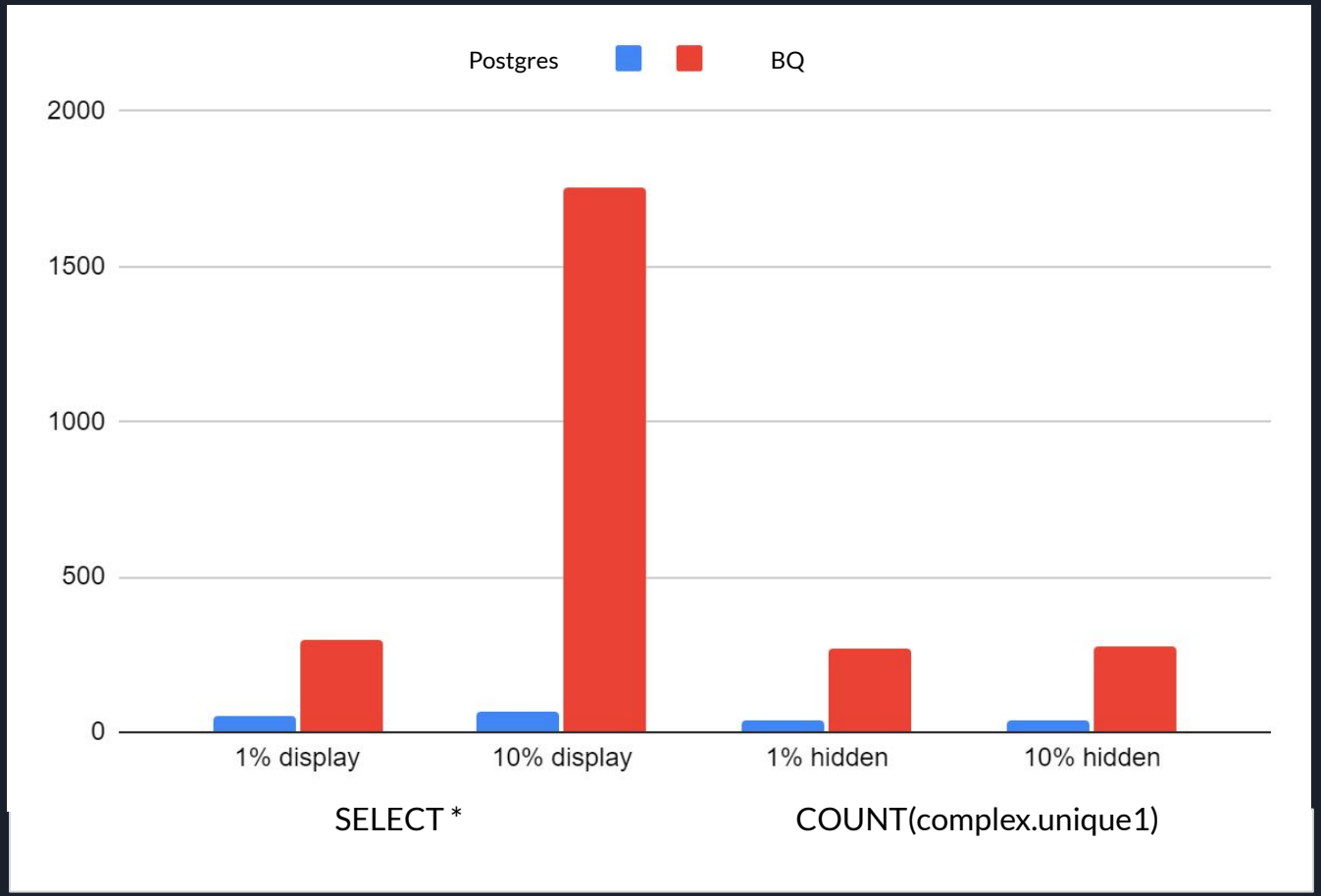
```
SELECT *
FROM 100Ktup
WHERE complex.unique1 BETWEEN 792 AND 10791
```

Query 3: 1 % Selectivity

```
SELECT COUNT(complex.unique1)
FROM 100Ktup
WHERE complex.unique1 BETWEEN 0 AND 999
```

Query 4: 10 % selectivity

```
SELECT  COUNT(complex.unique1)
FROM 100Ktup
WHERE complex.unique1 BETWEEN 792 AND
10791
```



SELECT *

COUNT(complex.unique1)

Experiment 3: Join on index vs no index (complex data type)

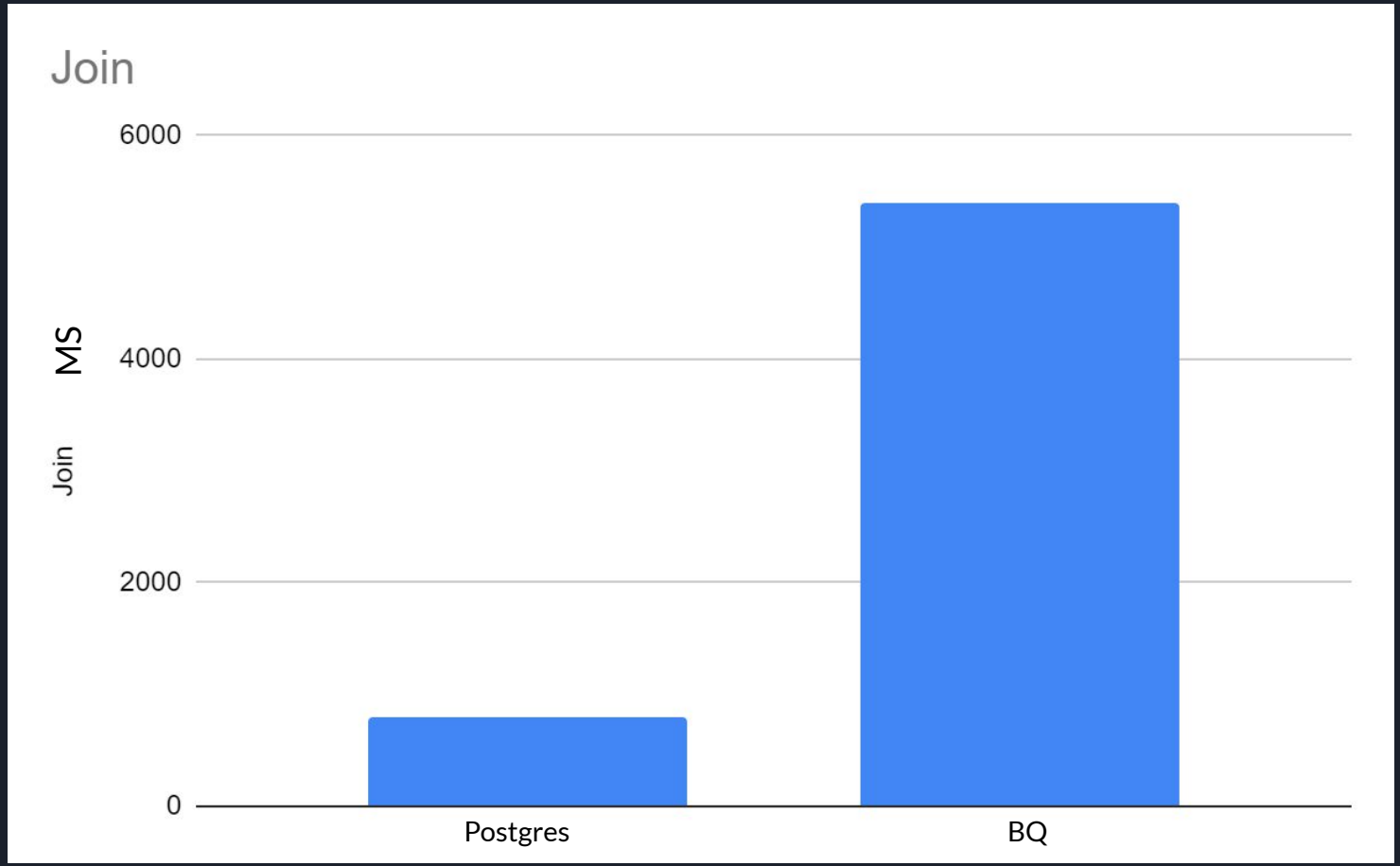
Purpose: Test query performance for an equijoin on complex data type with and without index

Expectation: Postgresql will have better performance because of the index

Index on complex.unique1 in postgresql

Query 1: Index VS no index on complex.unique1

```
SELECT *
FROM 100ktup, 1mtup
WHERE 100ktup.complex.unique1 = 1mtup.complex.unique1
```


Join
6000
4000
2000
0
MS
Join
Postgres
BQ

Experiment 4: SCALABILITY

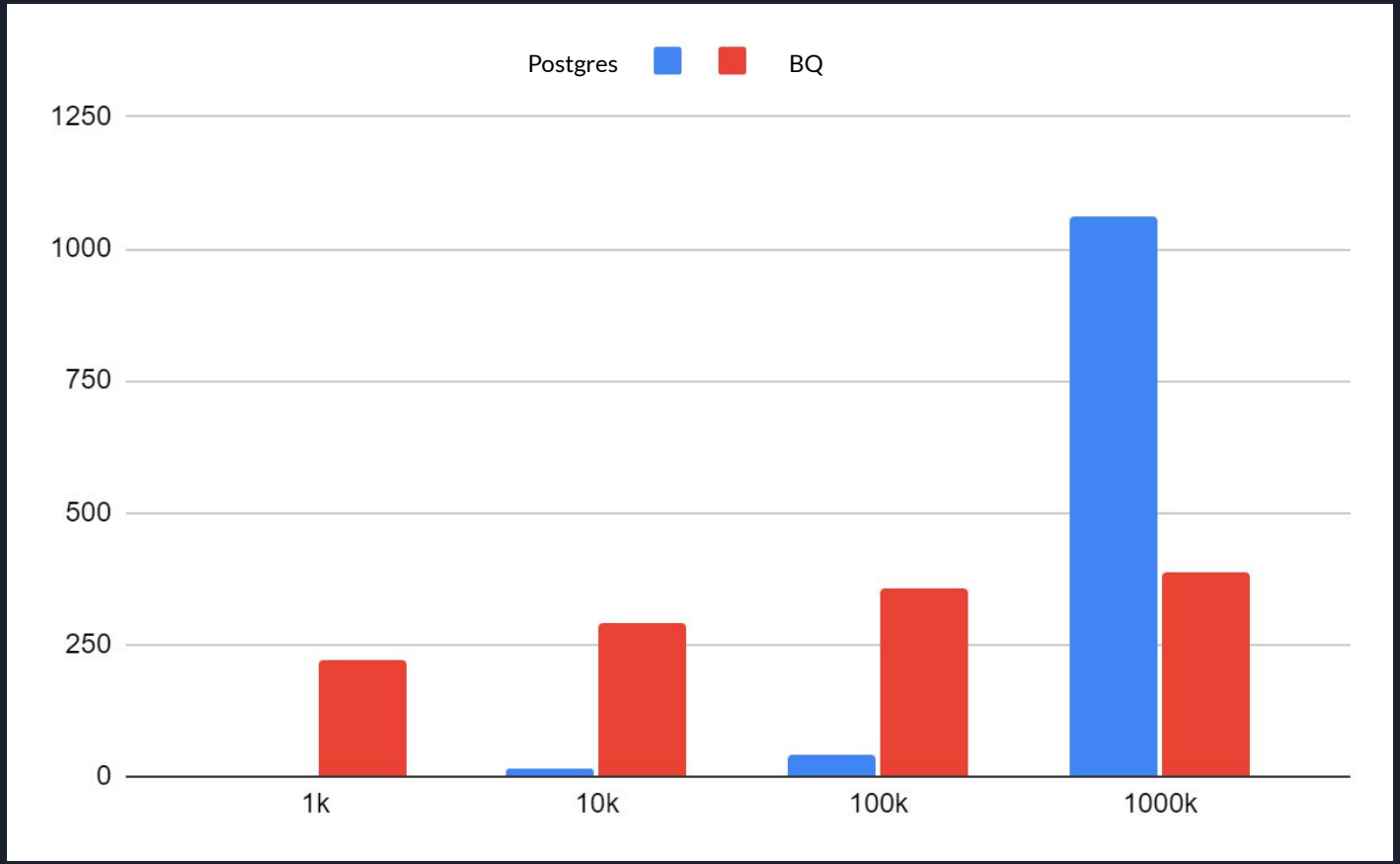
Purpose: Test scalability of Postgresql and Google Bigquery

Expectation: Google Bigquery will have better performance as the database scales

Number of Tuples: 1k, 10k, 100k, 1000k

Query:

```
SELECT COUNT(complex.unique1)
FROM 100Ktup
WHERE complex.unique1 BETWEEN 792 AND 10791
```


Postgres
BQ
1250
1000
750
500
250
0
1k
10k
100k
1000k

Conclusion

- Postgres had better performance in nearly all of the experiments except for scalability
  - It would be interesting to see how the other experiments would have resulted with larger tuple sizes > 1 million
- Having an index really increases the speed of query
- Big query performed very poorly when selecting all of the attributes in a row. If you were to use Bigquery in the real world it would be important to select only the attributes that you need
- An index has a substantial effect with increasing the query performance
- Google Bigquery was able to scale much better than postgres without modifying much of the default settings for postgres

Lessons Learned

- The permanence of google Bigquery was sometimes erratic. On our join experiment, google Bigquery averaged around 5 seconds but in one case it took 30 seconds to complete.
- Postgres indices and working memory management will extend its scalability
- Bigquery however is ultimately more scalable, without any labor overhead maintaining the database
- For most projects, small databases will be significantly faster with postgres (30X+)
- Better tests with the custom data types and BQ would involve something like a GIN + trigram index
- By using the cloud, you relinquish a lot of control

THANK YOU!