DATABASE MGMT & SYS IMPLEM Term Project

Jordan Tippett & Adam Craig

Option 1

System Selection: Google Bigquery & Cloud Sql for Postgresql

Interests:

- 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

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

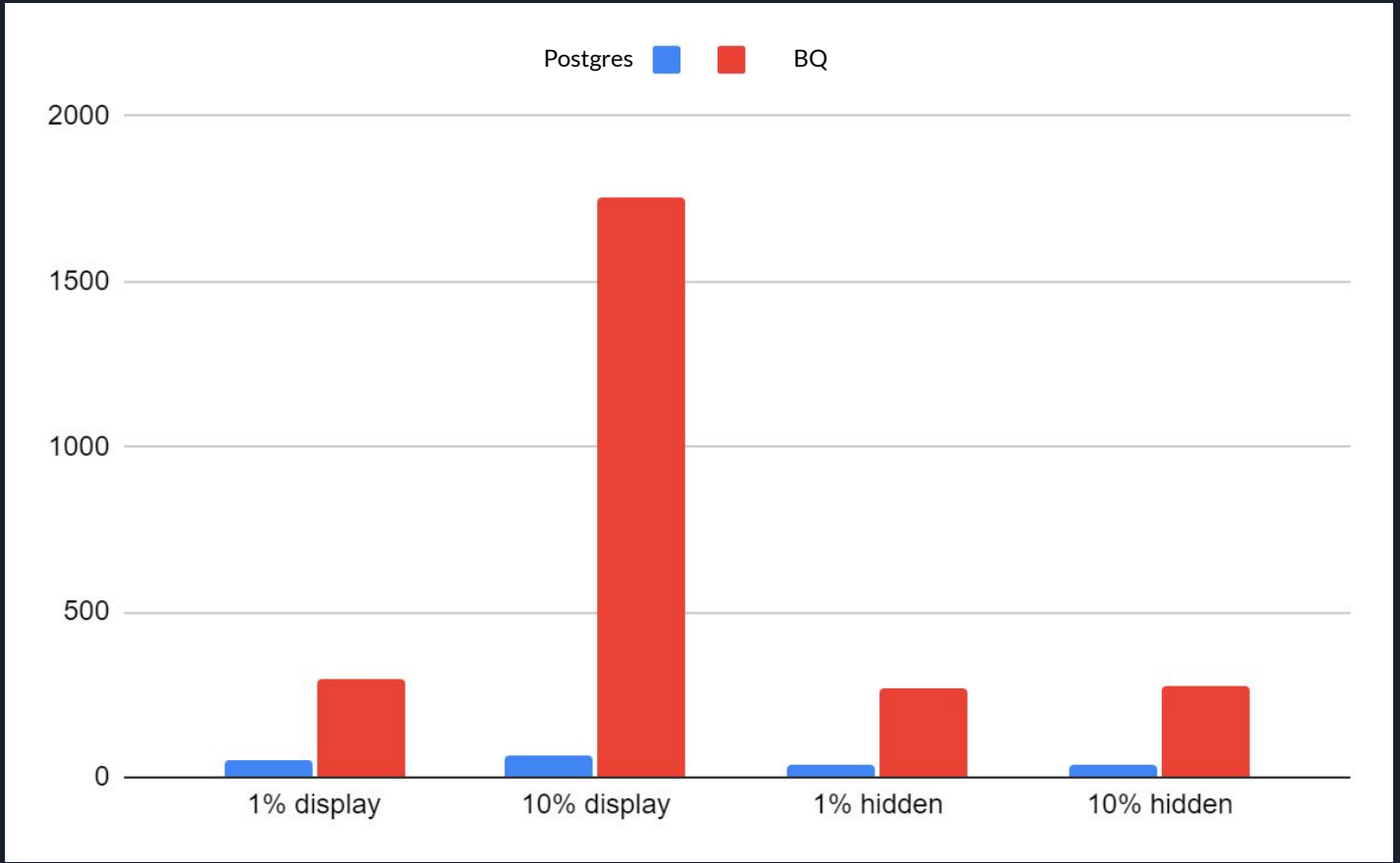
```
SELECT *
FROM 100Ktup
WHERE complex.unique1 BETWEEN 792 AND 10791
```

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
```


Postgres
BQ
2000
1500
1000
500
0
1% display
10% display
1% hidden
10% hidden

Experiment 4: SCALABILITY

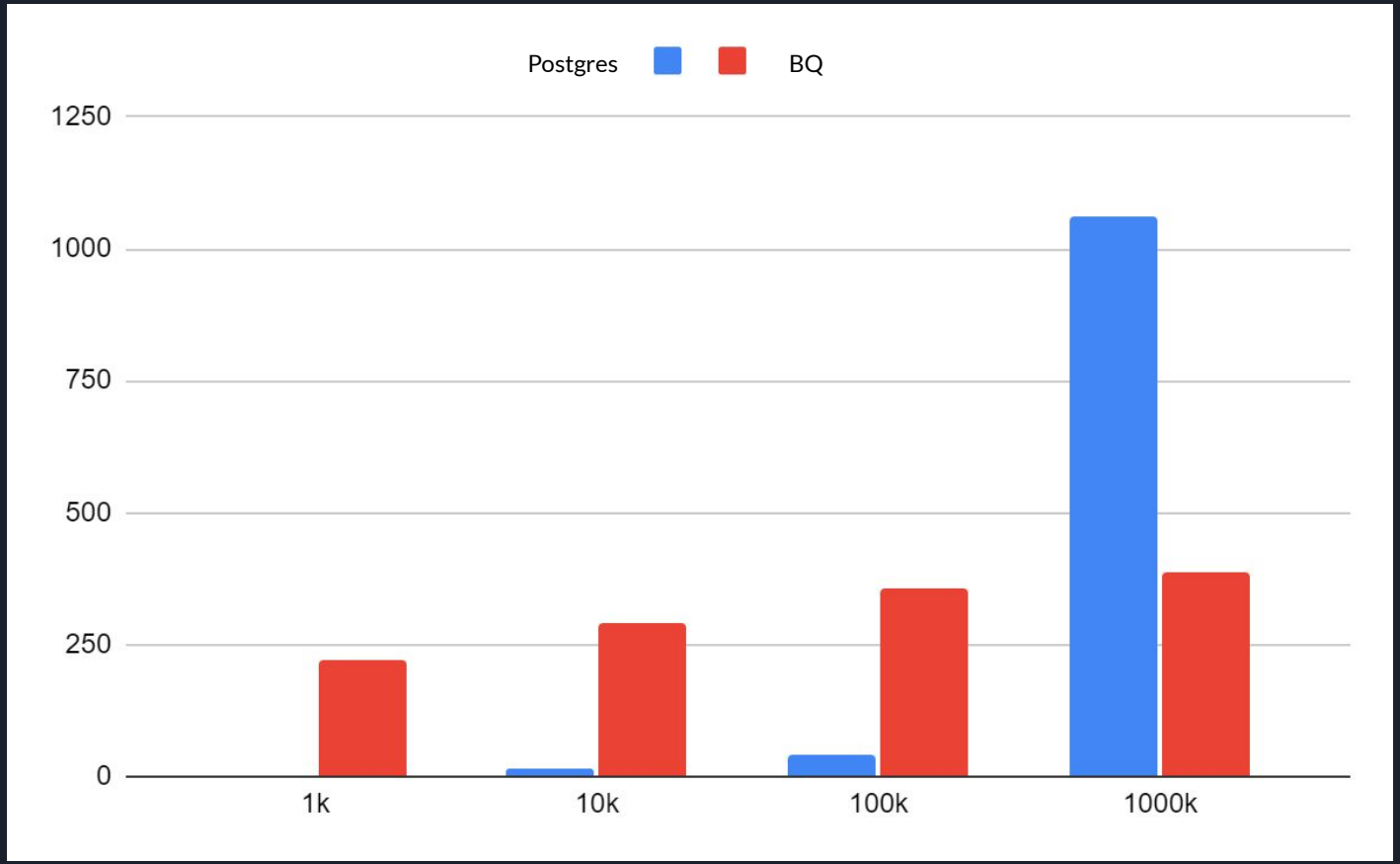
Purpose: Test scalability of Postgresql and Google Bigquery

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

Lessons Learned

- 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
- Perhaps a better comparison would be Cloud SQL vs BQ
- By using the cloud, you relinquish a lot of control

THANK YOU!

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

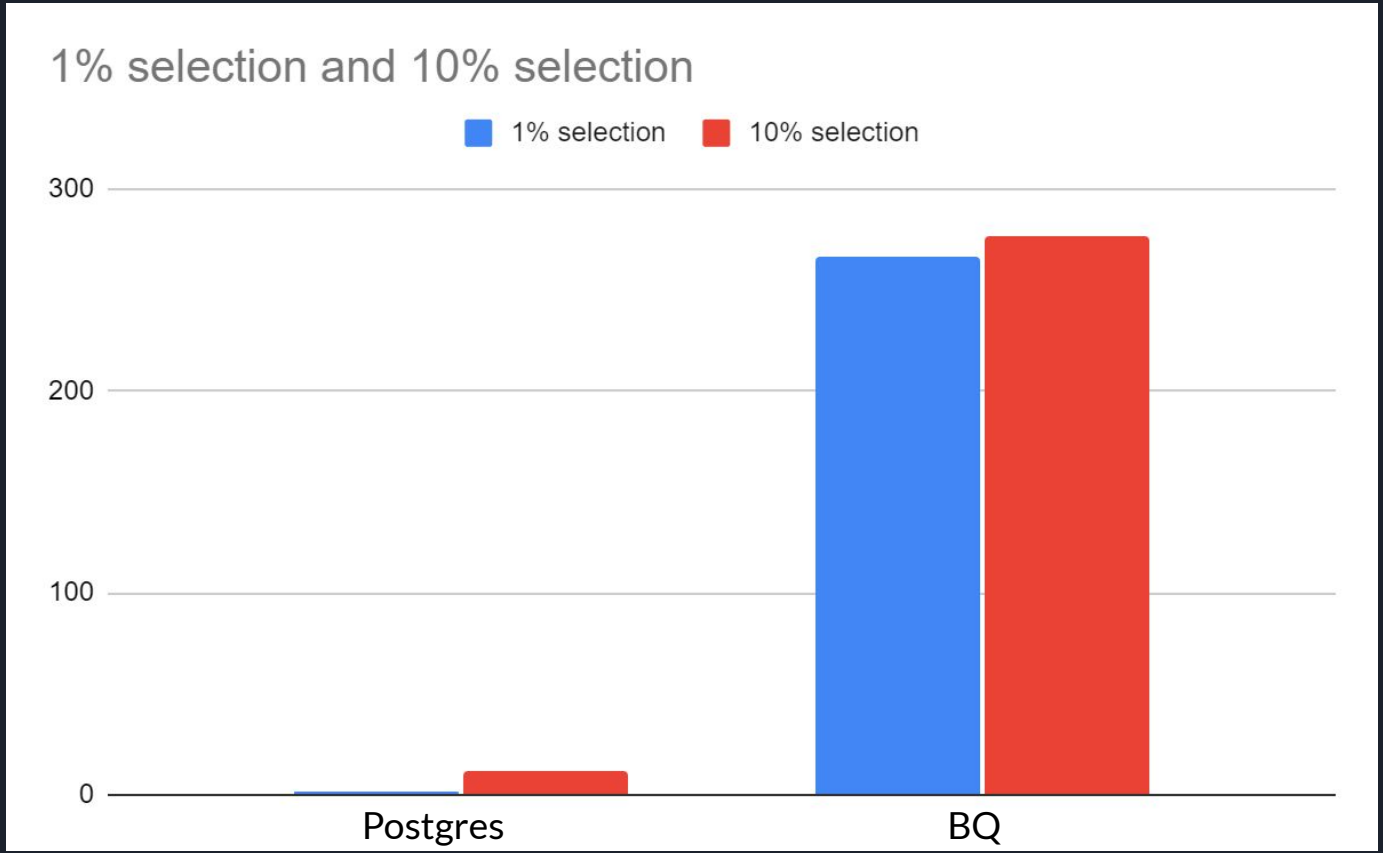
Query 1: 1 % Selectivity

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

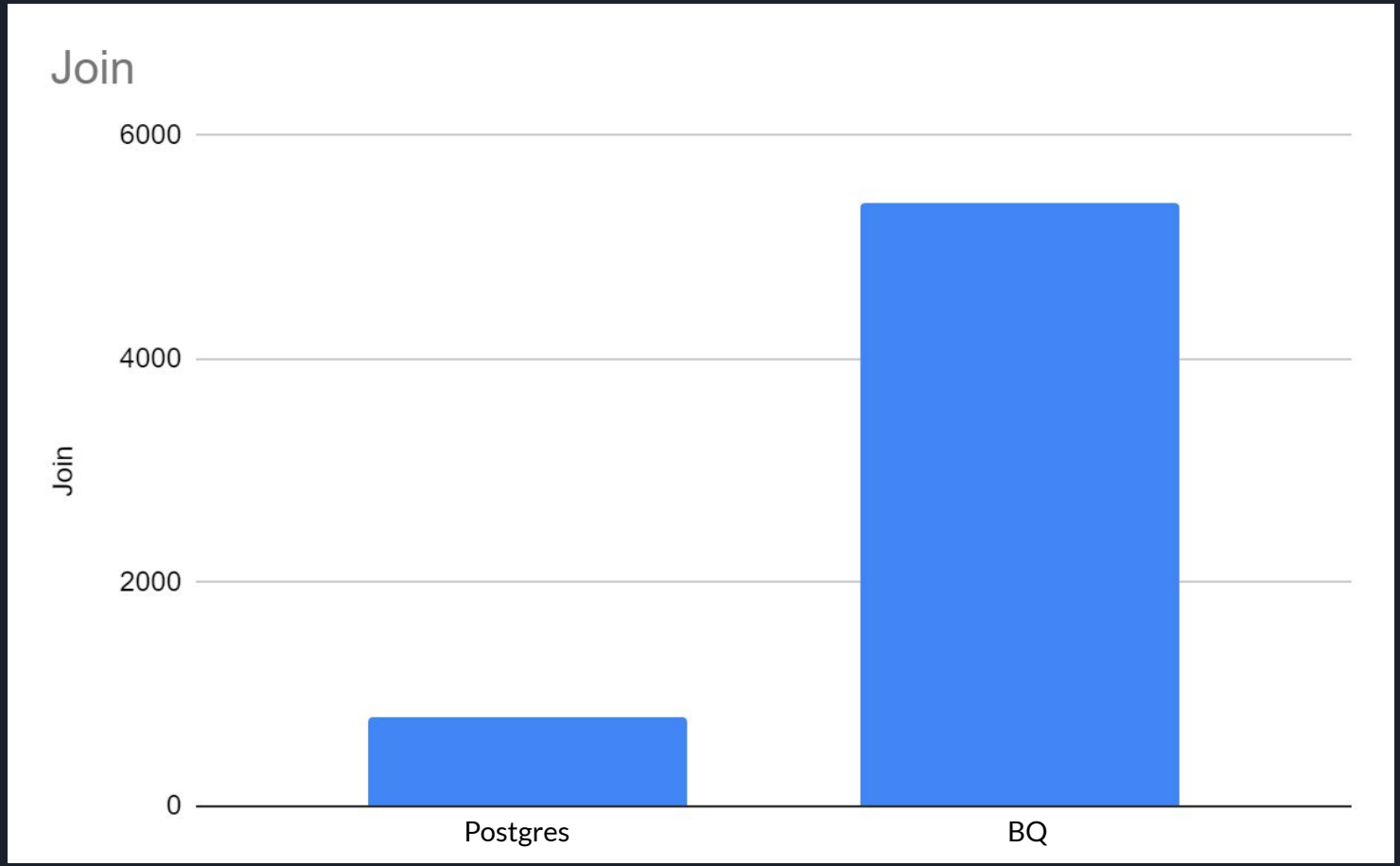
Purpose: Test query performance for an equijoin on complex data type with and without index

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
Join
Postgres
BQ