Report on Week 3

Lei Liu, 9669373

Task 1

```
--1. simple selection (σ) query over a single relation
select * from Bar where name like 'T%' ;

--2. a simple projection (π) query over a single relation
select name from bar;

--3. a σ query over the Cartesian product1 (×) of two relations
select * from Bar, Serves;

--4. if possible, reuse the previous query and ensure duplicate (δ) removal
select distinct * from Bar, Serves;

--5. if possible, reuse the previous query and instead of Cartesian product, used the natural join (⋈)
select distinct * from Bar b, Serves s where b.name = s.bar;

--6. a union (∪ ) query
select * from bar
UNION
select bar, price from serves;

--7. an intersection (∩) query
select name from bar
INTERSECT
select bar from serves;

--8. an aggregation (γ) query
select drinker, count(times_a_week) as total_times from frequents group by drinker;
```

Task 2

Firstly, I suppose, each query will be more costly with the increase of the size of its input, as more data are scanned, selected and output. For simple query 1 and 2, the increase of time cost could be linear since the input increase linearly. While, there could be an exponential increase of the time cost for query 3, 4 and 5. This is because it manipulate more than one table, so that the time and space cost would be the time of tables scale. Similarly, the query 6, 7 and 8 would also cost a lot of time and space as the increase of the scale of database, because complex operation are applied.

Additional, a simple selection query (query 1) would cost more time than a simple projection on the same input where each tuple are required an evaluation of predicate.

Besides, the distinct operation filters duplicate tuples from a large intermediate relation produced by Cartesian join of two relations, so that require more executing time than a Cartesian join which could cause memory overflow on huge database. A natural join should be executed faster than both these two queries, as join and selection can be done at the same time. This produce smaller intermediate relation, thus saving time on I/O costs.

Comparing query 6 and 7 with 8, UNION and INTERSECT maybe cost less time and space, because they are a combination of two sub queries that are simple selection and projection. Whereas, an aggregation query are much more complex. Although it only scan one table, more complex operation like group by and count are executed, which require more intermediate operation.

Task 3

Experiment result on queries from task 1:

Table 1: Average running time results of different queries executed at different scale database

Scale	Selection	Projection	Cartesian Product	Distinct Cartesian Production	Natural Join	Union	Intersection	Aggregation
bbd	0.000602	0.000367	0.00022175	0.0009025	0.00050125	0.0005245	0.00018	0.0002705
bbd_1	0.000684	0.000519	0.00017975	0.000072	0.00016075	0.0003095	0.0002345	0.00063125
Bbd_10	0.000511	1.66E-05	0.001414	0.0016585	0.000241	0.00046975	0.000701	0.014959
Bbd_100	8.84E-05	0.000524	0.0025635	0.007758	0.0003025	0.00037075	0.0003115	0.13105625
Bbd_1000	0.000498	0.000291	0.016227	0.0663535	0.00111325	0.0007925	0.000597	1.635359
Bbd_10000	0.000274	0.000653	0.15047225	0.6749615	0.00445975	0.00206525	0.00163575	20.75804775

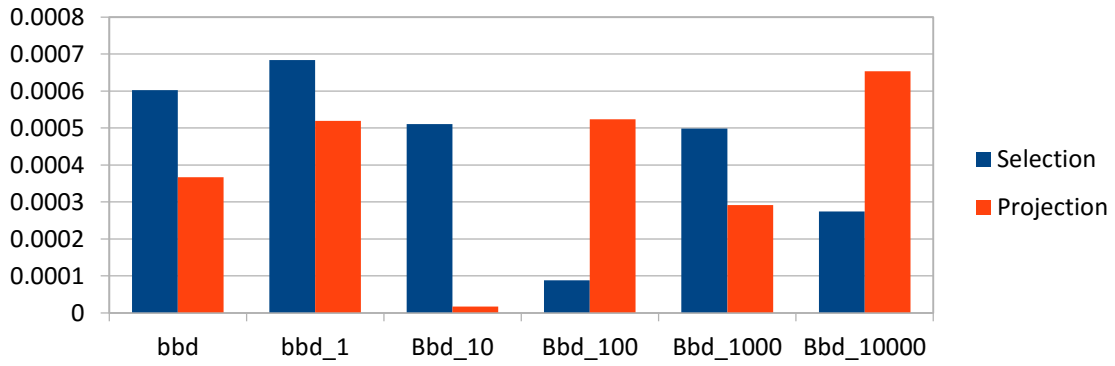

Figure 1: Compare of Running Time between Selection and Projection Queries

From figure 1 and table 1, we can see that both simple selection query and simple projection query have slight change on executing time. However, it is noticeable that selection query cost less time on large scale database. The reason could be the optimization from Sqlite3 such as index or operation on simply the predicate. While, with the rise of scale, projection query cost more time as expected since more I/O costs are requied.

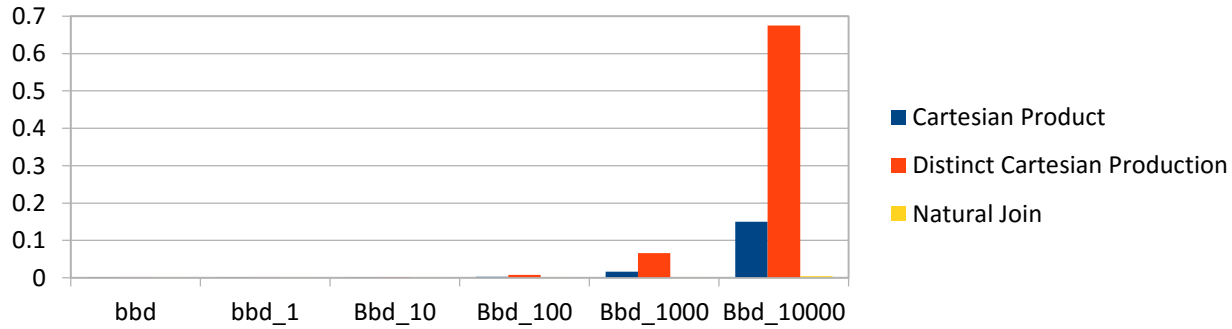

Figure 2: Compare of Running Time between Cartesian Product, Distinct Cartesian Production and Natural Join Queries

As expected above, the time cost of Cartesian product and eliminating duplicate from Cartesian

product increased exponentially on larger scale of database, but much more significantly on distinct Cartesian product query, showing in figure 2. This is because a whole table scan is required for each queried table and used to produce a bigger table (the scale is the product of every queried table's scale that require numerous I/O costs) and then filtered duplicate which also need lots of I/O operation. The natural join cost more time on large scale as well, but much more slightly even on a huge scale data. This is because the table join and filter is done in the same step which has less I/O costs.

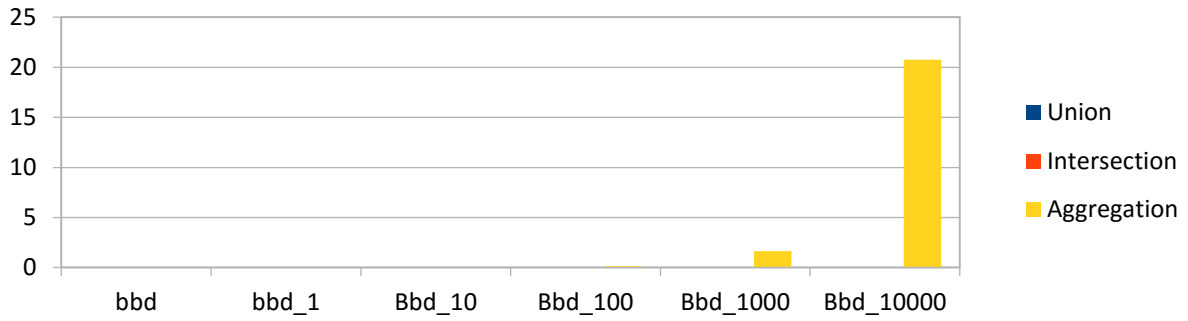

Figure 3: Compare of Running Time between Cartesian Product, Distinct Cartesian Production and Natural Join Queries

From this figure, we can found that UNION query and INTERSECT query had almost no change at their executing costs, while the cost of aggregation query increased exponentially. All queried will produce intermediate relation, but differ from their logical functions. It indicates that aggregation function are much more costly and complex than the others.

In conclusion, from this experiment, natural join, simple selection and projection have less challenge on Sqlite3. However, Cartesian join, distinct operation are complex and more costly because of the product of large intermediate relation, and aggregation is costly because of the complex of logical operation. These will throw a great challenge on executing queries.