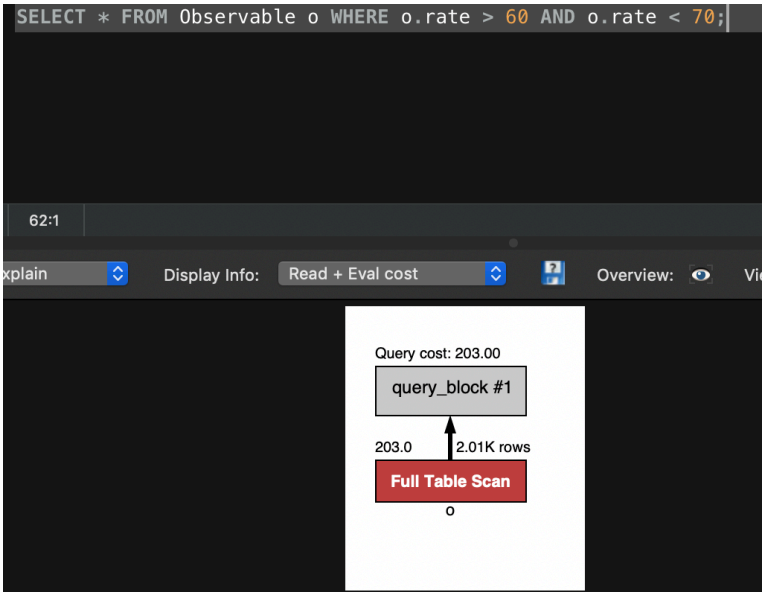
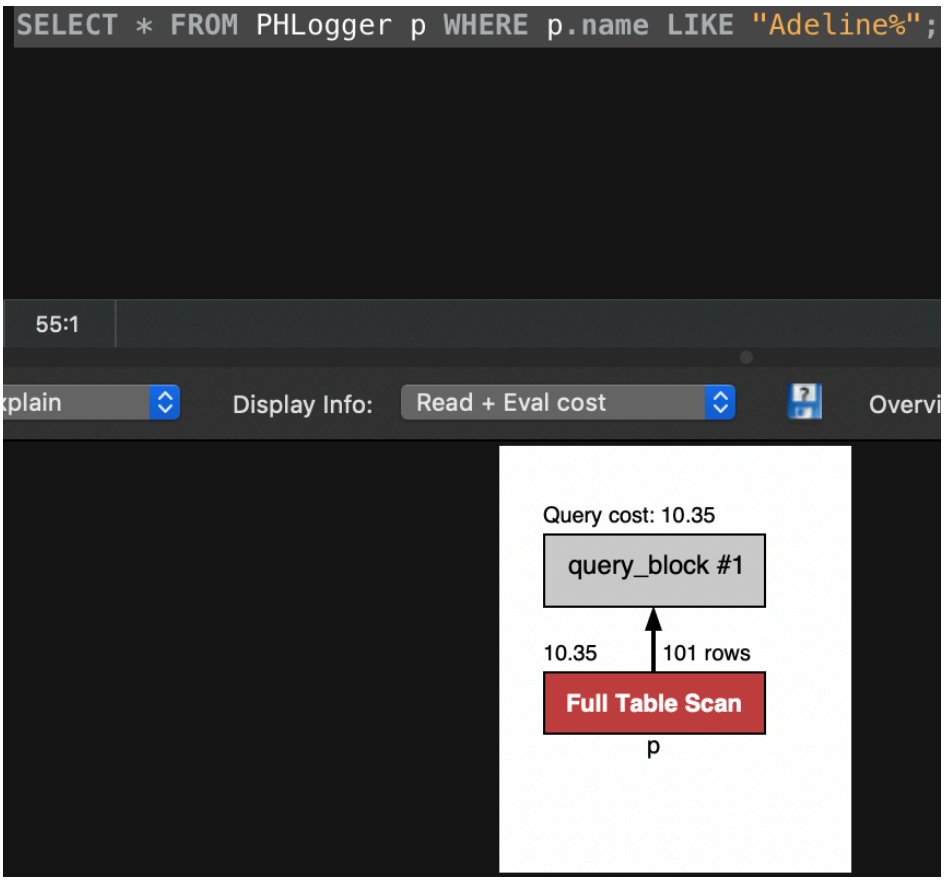
Last Name: ZHOU First Name: YIHUA Student ID: 83186710

1. [12 pts] Report on the query plan of each query. (Snapshot of the query plan, not the whole screen)

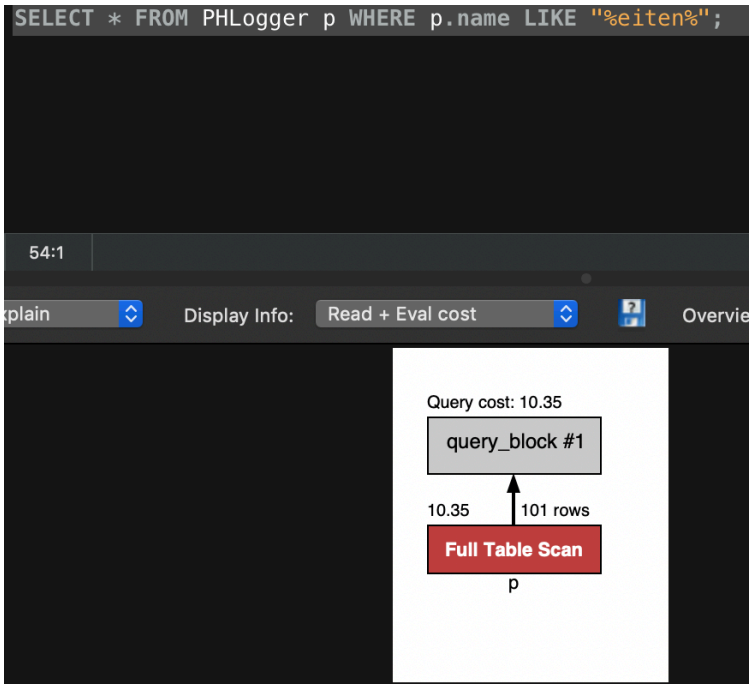
a) [3 pts] SELECT * FROM Observable o WHERE o.rate > 60 AND o.rate < 70;



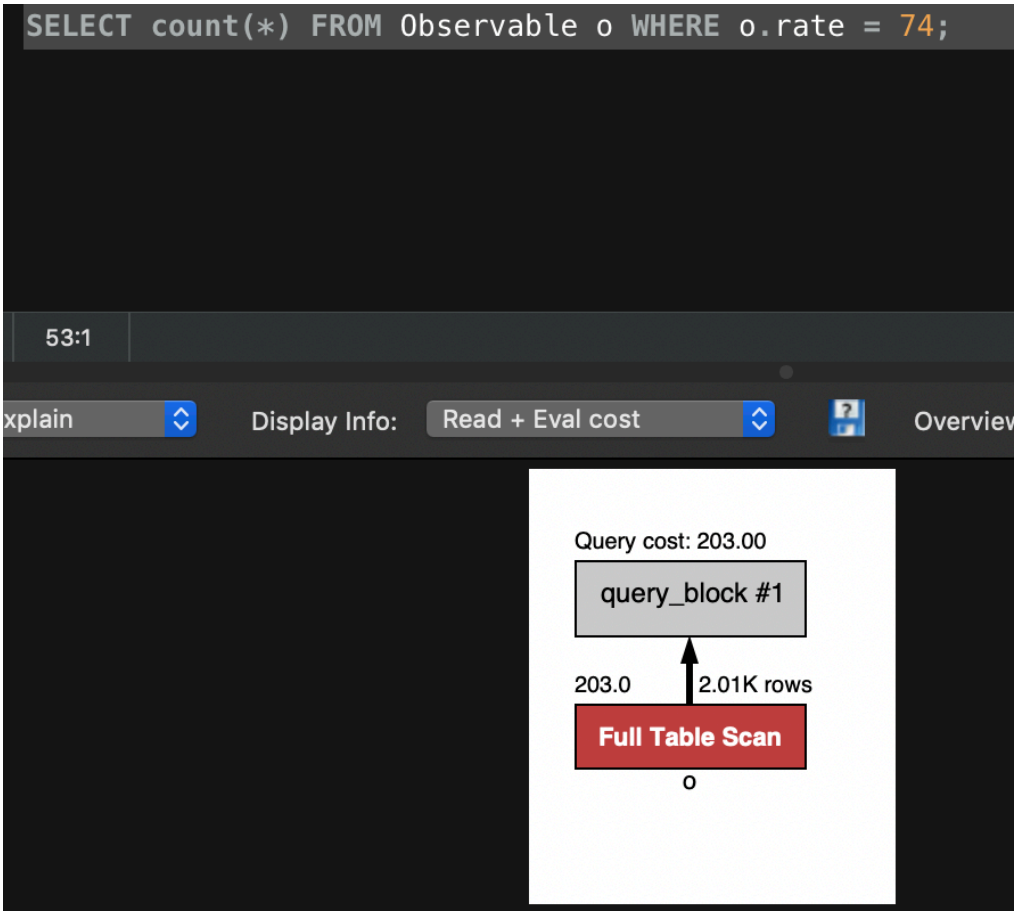
b) [3 pts] SELECT * FROM PHLogger p WHERE p.name LIKE "Adeline%";



c) [3 pts] SELECT * FROM PHLogger p WHERE p.name LIKE "%eiten%";



d) [3 pts] SELECT count(*) FROM Observable o WHERE o.rate = 74;



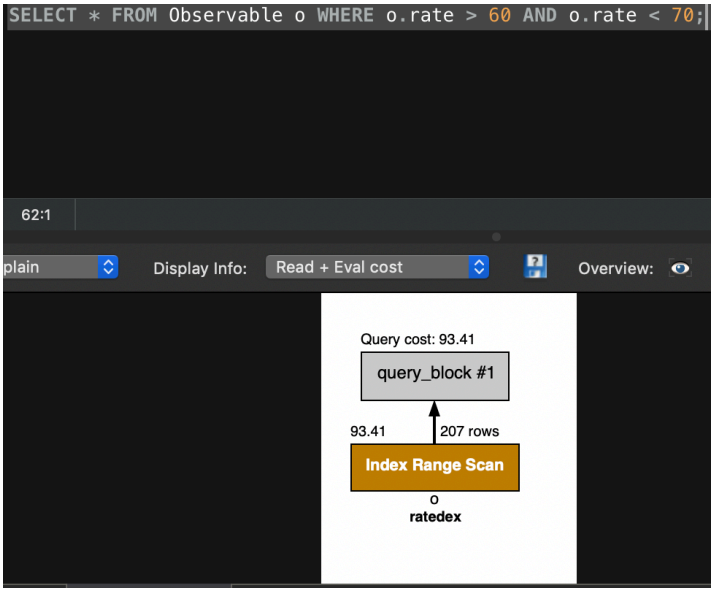
2. [10 pts] Now create indexes (which are B+ trees, under the hood of MySQL) on the PHLogger.name attribute and Observable.rate. (e.g., create two indexes, one per table.) Paste your CREATE INDEX statements below.

```
CREATE INDEX namedex ON PHLogger(name) using btree;

CREATE INDEX ratedex ON Observable(rate) using btree;
```

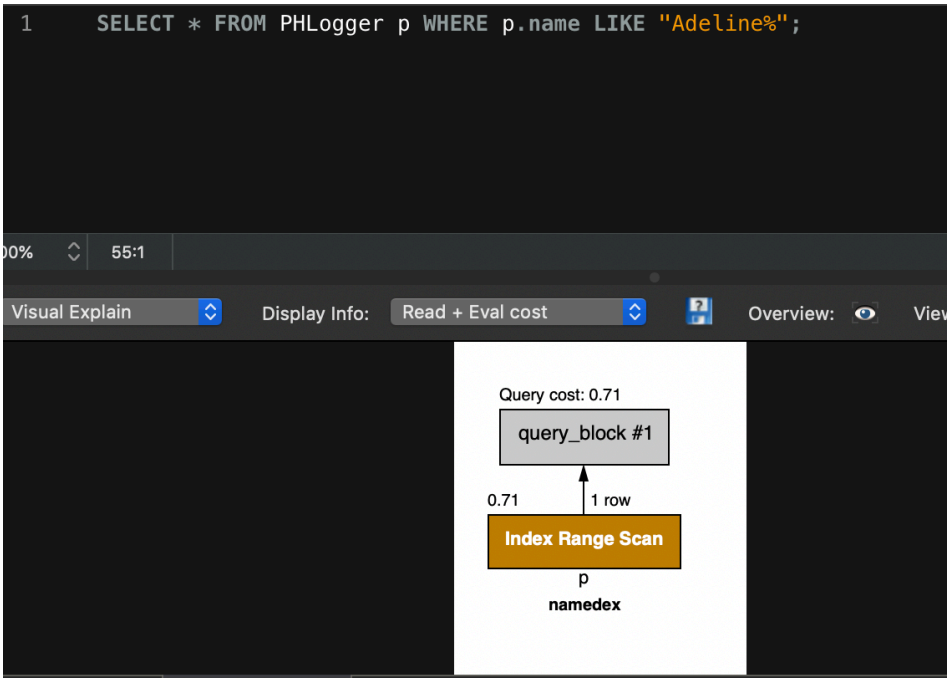
3. [12 pts] Re-explain the queries in Q1 and indicate whether the indexes you created in Q2 are used, and **if so whether it is an index-only plan**. Report on the query plan after each query, as before.

a) [3 pts] SELECT * FROM Observable o WHERE o.rate > 60 AND o.rate < 70;



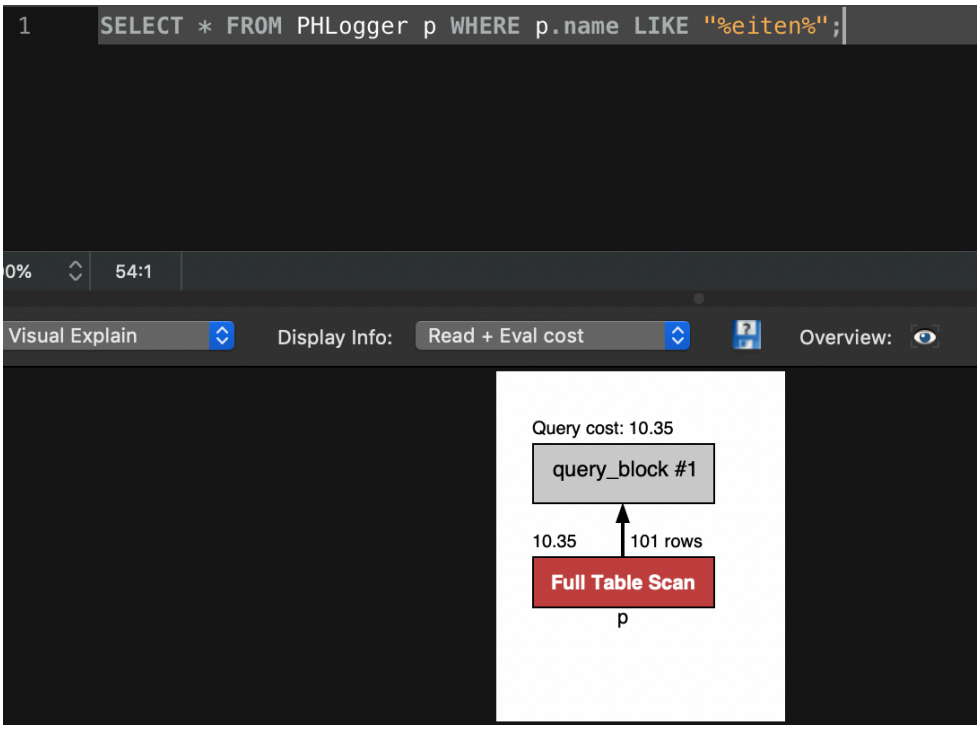
The ratedex is used and not an index-only plan.

b) [3 pts] SELECT * FROM PHLogger p WHERE p.name LIKE "Adeline%";



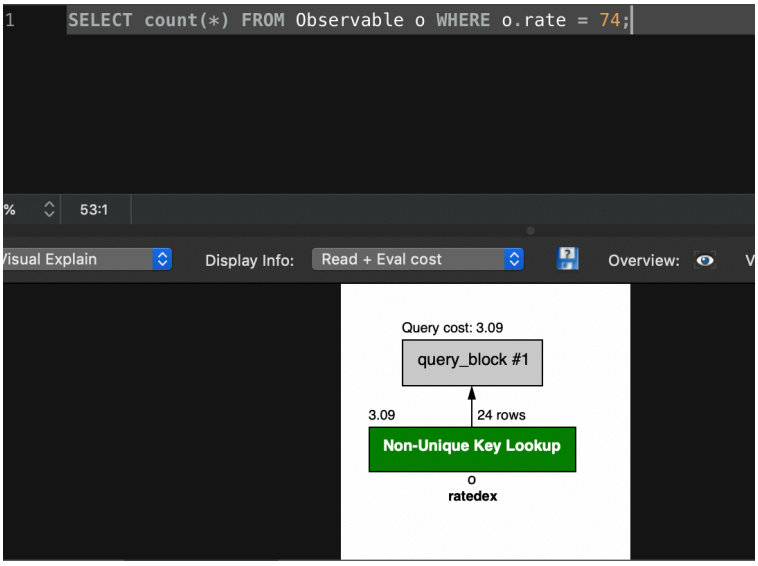
The namedex is used and not an index-only plan

c) [3 pts] SELECT * FROM PHLogger p WHERE p.name LIKE "%eiten%";



No index is used.

d) [3 pts] SELECT count(*) FROM Observable o WHERE o.rate = 74;



The ratedex is used and it is an index-only plan

4. [21 pts] Examine the above queries with and without the use of an index. Please briefly answer the following questions.

a) [7 pts] For range queries (e.g., query a), explain whether an index is useful and why (assume the number of result records in the selected range is extremely small compared to the number of records in the file).

With index

For the range query (60 < a.rate < 70), it will only search under the range using ratedex, and less time comparing to full table screen

Without index

Full table scan may be less time if having bigger range than (60 < a.rate < 70)

b) [7 pts] For each LIKE query (b and c), explain whether an index is useful and why (<= 2 sentences per query).

B: the namedex will be better, because it has less time than full table screen

C it needs full table screen because B+ tree doesn't supply this option

c) [7 pts] For equality queries (e.g., query d), explain whether an index is useful and why (again assuming that the number of selected result records is small compared to the number of records in the file).

Using index because system can find its records without entering any other records. More sufficient than using full table scan.

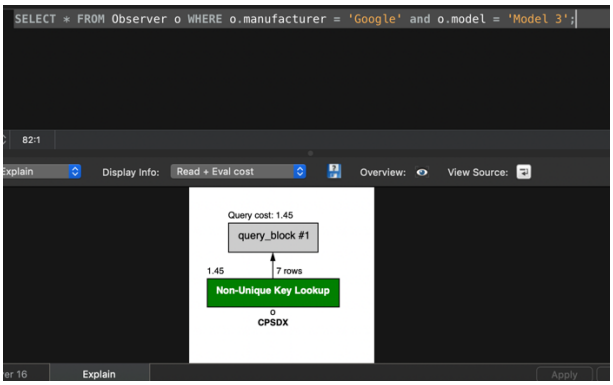
5. [21 pts] It's time to go one step further and explore the notion of a "composite Index", which is an index that covers several fields together.

a) [5 pts] Create a composite index on the attributes manufacturer and model (in that order!) of the Observer table. Paste your CREATE INDEX statement below.

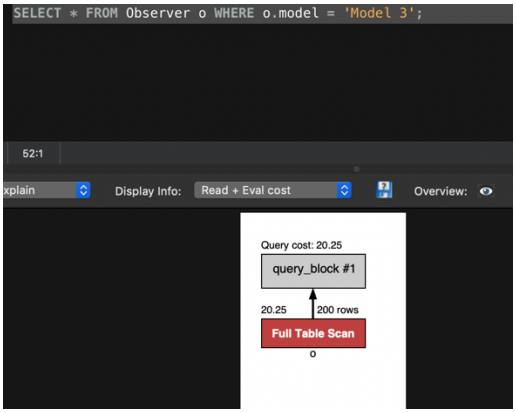
CREATE INDEX CPSDX ON Observer(manufacturer,model);

b) [6 pts] 'Explain' the queries 1 and 2 below. Report on the query plan of each query, as before.

1) [3 pts] SELECT * FROM Observer o WHERE o.manufacturer = 'Google' and o.model = 'Model 3';



2) [3 pts] SELECT * FROM Observer o WHERE o.model = 'Model 3';



c. [10 pts] Report for each query whether the composite index is used or not and why (<= 2 sentences per query).

A: composite index makes equality search so every page is equal to unique value, index is better

B: values in the composite indexing are in order, in order to make sure value is in specific column, it needs full table screen.

6. [24 pts] For each of the following queries indicate whether the use of an index would be helpful or not. If so, specify which tables and attributes an index should be created on and the best choice between a clustered or unclustered index. For the sake of this question, you don't have to worry about how your choice of the index would affect other queries running in the system -- consider each query in isolation.

a) [8 pts] SELECT * FROM Observer o WHERE o.phlid = 1;

A clustered index is better, foreign key In Overserve table is not unique, so clustered will force than together into same place.

b) [8 pts] SELECT o.kind, count(*) AS cnt FROM Observer o GROUP BY o.kind;

A unclustered index is better, since clustered here does not help anything. So unclustered index would be best try.

c) [8 pts] SELECT * FROM PHLG_obs;

since it select everything from table, full table scan is best choice