πname (σcategory =’food’ v σcategory = healthcare’((r(Products)\*r(Produced\_by))\*(r(Manufacturer)))

2) Retrieve the names of all manufacturers who always produce products of category ‘drink’.

πname (σcategory =’drink’((r(Products)\*r(Produced\_by))\*(r(Manufacturer))) - πname (σcategory <>’drink’((r(Products)\*r(Produced\_by))\*(r(Manufacturer)))

3) Retrieve the descriptions of all products that are produced by two or more manufacturers.

πdescription(πPid(σMid <> σMid\_other (δMid->δMid\_other(πPid,Mid(r(Produced\_by))\* (πPid,Mid(r(Produced\_by))\*r(Products)) Get some help from tutor

4) For all products of category ‘food’ list their descriptions and the names of their manufacturers.

πdescription,name(σcategory =’food’ ((r(Products)\*r(Produced\_by))\*(r(Manufacturer)))

b) [8 marks] Translate the following two queries into plain English and into SQL:

1) πPhone (r(Products) \* (σAmount>50 (r(Produced\_By))) \* r(Manufacturer))

Retrieve phone numbers of the manufacturers who provided more than 50 products.

SELECT phone FROM Products NATURAL JOIN Produced\_By NATURAL JOIN Manufacturer WHERE Amount >50;

2) πMId (σAmount>50 (r(Produced\_By))) ∩ πMId (r(Produced\_By) \* (σDescription=’Muffin’ (r(Products))))

Retrieve Mid of manufacturer which produce more than 50 muffins.

SELECT MId FROM Products NATURAL JOIN Produced\_By WHERE Amount > 50 AND Description = ‘Muffin’;

Q2:

a) [20 marks] Heuristic query optimization Suppose we are given a query in SQL

SELECT StudentId, Name, Grade FROM Student NATURAL JOIN Enrolled NATURAL JOIN Course WHERE CourName = ‘Database Systems Engineering’ AND Term < 2019 AND NoOfPts > 300;

1) [5 marks] Transfer the above given query into Relational Algebra.

πStudentId,Name,Grade(σCourName =’Database Systems Engineering’ ^Term <2019 ^NoOfPts >300((r(Student)\*r(Enrolled))\*r(Course)))

2) [5 marks] Draw a query tree corresponding to your answer for subquestion 1).

图片包含 动物

描述已自动生成

3) [10 marks] Transfer the query tree from 2) into an optimized query tree using the query optimization heuristics.

Apply for the rule 6 move select operation down the tree.

图片包含 文字

描述已自动生成

图片包含 文字, 地图

描述已自动生成Apply for rule 9 Switching join based on frequency, Students who have more than 300 is bigger than courses which name is database systems engineering. Thus, we can swap node student and node course.

Apply for rule 7 to the tree we can get the final tree below:

图片包含 文字, 地图

描述已自动生成

1. 图片包含 文字

   描述已自动生成 [10 marks] For the given query below draw a query tree and calculate the cost of executing the query.

π StudentId, CourName ( r(Enrolled) e.CourseId = c.CourseId r(Course) )

1. [20 marks] For the given query below draw a query tree and calculate the cost of executing query.

πStudentId, Name (σterm =2019 ˄ CourseId =‘SWEN304’ (r(Student) \* r(Enrolled) ) )

图片包含 文字

描述已自动生成

Q3:

1. [10 marks] Improve the cost estimate of the following query:

select count(\*) from customer where no\_borrowed = 6;

图片包含 屏幕截图

描述已自动生成

It is the original query which has cost 114.42. We always think worst situation for improve the performance. Although it is simple query but it has a high cost. The reason of it may because this query scanned whole customer table. For optimizing the cost of the query, there are multiple ways to achieve our goals. We can use ‘order by’ to sort the number of borrowed books. The output is down below:

图片包含 屏幕截图

描述已自动生成

If we only consider worst case, the cost even increased after ordering number of books. It is still use sequential scan to scan the whole customer table. Because, we have a lot of data in this table which is around 5000. Thus, some other scan techniques can be implemented to improve the efficiency. There are two scans may occur, index scan and bitmap scan. However, I should create an index for no\_borrowed.

图片包含 屏幕截图

描述已自动生成

After I created the index for this table on no\_borrowed. The cost decreased dramatically. From 114.66 to 5.55. Because, this query is asking for one value. The system use Index only scan to find out the our target. Thus, the performance increased a lot.

The increase is around 114.66-5.55/114/66 = 95.15%.

However, bitmap scan will use index to identify the portion and use sequential scan to find concrete query. In this scenario, the query subsets is small. Index scan is good enough to deal with this situation.

Explanation: Why index scan have a better performance for this query? Compared with sequential scan, the advantage of the index scan is very efficient when we only select small amount of data. In our query, we only selected ‘no\_borrowed’ = 6. The results of this datasets is 63 rows.Thus, index scan is the suitable scan method for our question. However, Sequential scan is good at handling with selection of large rows.

1. [7 marks] Improve the efficiency of the following query:

select \* from customer where customerid = 4567;

图片包含 屏幕截图

描述已自动生成

The original query has cost up to 114.25. It is also use sequential scan. Thus, to change sequential scan to index id, I create another index for customer Id.

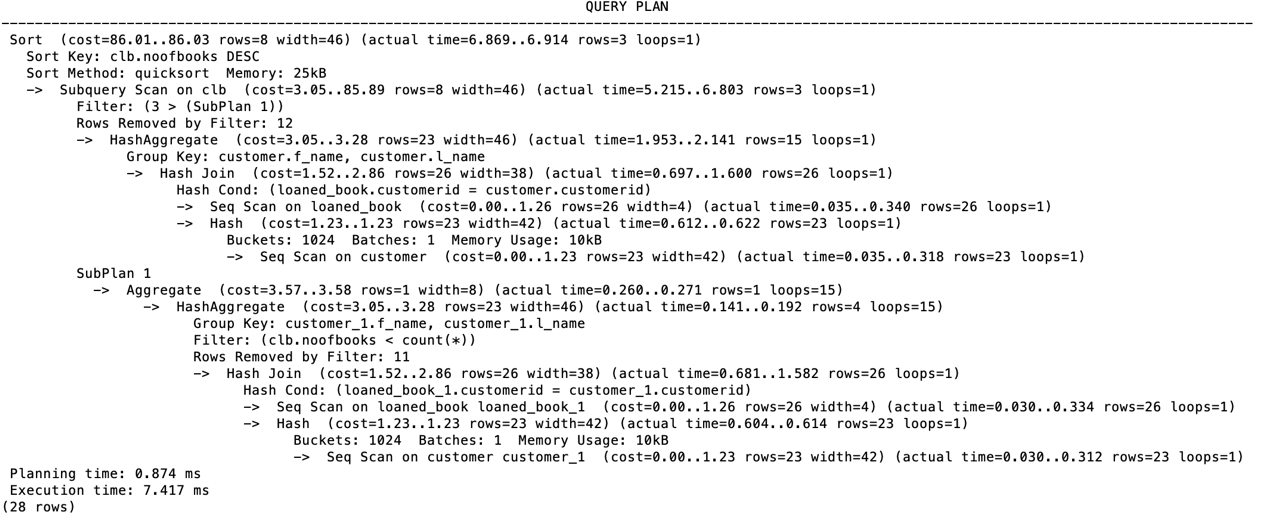
图片包含 屏幕截图

描述已自动生成

After giving index for customer Id, the performance increased from 114.25 to 8.30. The percentage increased 114.25-8.30/114.25 = 92.73% It almost increased 93%.

Explanation: It is same reason with last question. Index scan is more suitable for selecting small dataset. The target is a very specific row when customer Id equal to 4567, there is only one row being selected. Then index scan is more efficient.

Q3:



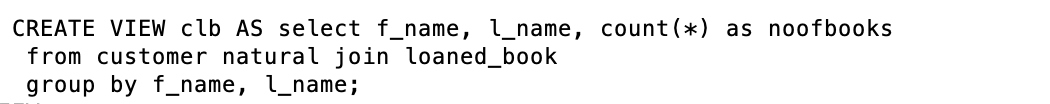
The original query have a poor performance. Thus, for increase the performance, few ways to improve. As we know that Natural Join is very expensive statement. Firstly, I swap all ‘NATURAL JOIN’ to ‘INNER JOIN’

However, the cost doesn’t change at all. Thus, I have to try some other method to reduce cost. We can investigate the purposes of this query.

图片包含 物体, 天线

描述已自动生成The result of this query is to find the people who have borrowed more than 3 books.

As we can see, there are 3 results.

For generating the same result, we can CREATE VIEW of clb.

图片包含 文字, 收据

描述已自动生成This view will generate the customer’s name and how many books they borrowed.

Thus, we can use this view to achieve our goals.

图片包含 物体

描述已自动生成

This is generate same result. Thus, we can find out the cost of this query.

图片包含 屏幕截图

描述已自动生成

Eventually, this query have same result but with very small cost.

86.03-4.15/86.03 = 95.17% improvement.

Explanation: For this question, from my perspective, the cost drop because I only use (count(\*)) once when I create the view. The original statement used twice count(\*) function, it increased the cost. Thus, once I get rid of it, the efficiency raised significantly.