EXPLAINING THE WHY-NOT QUESTIONS IN TOP-K QUERY EXECUTION

Roopa Lakshmi G, Madhu Shalini R, Nethra J, Kousika.N
UG Scholar, Assistant Professor
Dept of Computer Science and Engineering,
Sri Krishna College of Engineering and Technology
kousika@skcet.ac.in

Abstract

In recent days, almost in all fields the data is digitalized. The enormous amount of data generates another set of data in turn. A more specialized database systems are required to manage this set of data, thus enabling efficient fetching of data whenever needed .Due to the fact that the existing database systems are increasingly more difficult to use, improving the quality and the usability of database systems has gained tremendous momentum over the last few years. In particular, the feature of explaining why some expected tuples are missing in the result of a query has received more attention .To approach this problem, we use the query-refinement method. That is, when the original top-k SQL query and a set of missing tuples are given as inputs, our algorithms return to the user a refined query that includes both the missing tuples and the original query results and the penalty will be calculated for getting the expected or the missing tuple. Also the non-numerical values are internally converted into numerical values to get the tuples based on non-numerical entities in the database table.

Keywords- top-k query, missing tuples, SQL

1 INTRODUCTION

After decades of effort working on database performance, recently the database research community has paid more attention to the issue of database usability, i.e., how to make database systems and database applications more user friendly. Among all the studies that focus on improving database usability (e.g., keyword search, form-based search, query recommendation and query auto-completion), the feature of explaining why some expected tuples are missing in the result of a query, or the so-called “why-not?” feature, is gaining momentum.

A why-not question is being posed when a user wants to know why his/her expected tuples do not show up in the query result. Currently, end users cannot directly examine the dataset to determine “why-not?” because the query interface (e.g., web forms) restricts the types of query that they can express. When end users query the data through a database application and ask “why-not?” but do not find any means to get an explanation through the query interface, that would easily cause them to throw up their hands and walk away from the tool forever—the worst result that nobody, especially the database application developers who have spent months to build the database applications, want to see. Unfortunately, supporting the feature of explaining missing answers requires deep knowledge of various database query evaluation algorithms, which is beyond the capabilities of most database application developers. In view of this, recently, the database community has started to research techniques to answer why-not questions on various query types.
In this paper, we study the problem of answering why-not top-k query in SQL .To

address the problem of answering why-not questions on top-k SQL queries, we employ the query refinement approach. Generally, the original SQL query and a set of missing tuples are given as inputs. This approach will return to the user a refined query whose result includes the missing tuples as well as the original query results.

2 RELATED WORKS

A common scenario faced by SQL programmers involves asking why one or more tuples are missing from the results of a query. One might wonder why, for instance, the result of a query is empty or why a query did not return certain tuples. In the case when queries are used to define multiple views, one may ask why, for instance, an employee information is missing from both the employee register and the payroll views. Often, the first reaction of programmers is to review the query itself since the explanation could be that a filter is too restrictive, or an inner-join should be an outer-join. However, if the expressions in the queries appear to be correct, then the next step is exploring the data sources to figure if data that maybe combined to yield the desired tuples are indeed there. The work was proposed to answer the Why not question in the execution of SPJ queries. This answers the why not question by telling that which query operator eliminated the desired tuple. It does not give the query to get that tuple. A work was proposed to tell why the expected tuples are missing in the SPJ query execution. It gives the user reason why tuple is missing and it also tells how to modify the query in order to get the expected tuple. Later the work was expanded to tell why the expected tuples are missing in the SPJA query execution. It gives the reason why the tuples are missing and also it tells how to modify the data to get the expected tuples. We can also conquer the why not questions and it was proposed to tell why the expected tuples are missing in the SPJA query. It adopts the query refinement approach to answer the question. It tells how to modify the query in order to get the expected tuple like few editing operations in the original query. Missing answers to top-K queries gives the solution to why not questions in the execution of non-SQL queries.ie. Normal Queries.

3 PROPOSED WORK

In proposed system user can give a SQL query as input. The system includes Selection, Projection, Join, Union and Aggregation (SPJUA) query operations. After execution of query, if the user expected answer is not in query result, then user can ask a why-not question for expected result. In a Why-Not Top-K Question algorithm the weighting value is calculated for an expected result. The Weighting value is used for the highest preference of user expected tuples which limits its practicability. In a Why-Not Top-K Dominating Question algorithm there is no need to specify the weighting value. These algorithms are able to return high quality explanations efficiently. Then Refined Query is generated for user expected answer. The Conquer method generates the refined query by modifying the predicates value which is provided by user in the query. The non-numerical values are internally converted into numerical values

3.1 Original Query Processing

The user given query is initially executed to get the results. From the result the user can find the missing or expected tuple.

3.2 Analysing the result

The original query result is analysed in order to decide the missing tuple or an expected tuple. The tuple can be any tuple in the table except the one in original query result.

3.3. Editing the original query

The original query is modified repeatedly in order to get the expected tuple in the result.The editing operations are done in the SPJA constructs or in the k-value or in the w-vector..

3.4. Penalty Calculation

Once the query is edited with some changes, it is executed and the penalty for the respected query is calculated. Penalty gives the amount of effort spent in modifying the original query in order the get the expected tuple into the result set.

3.5. Returning the Refined Query

Once all the editing operations are made and the query is executed, the query with expected tuple is taken and the penalty for the respective query is calculated. The algorithm will find the query with least penalty (i.e., query with least cost of editing) and gives it as an output along with the penalty.

4 IMPLEMENTATION

The algorithm gives the user 4 editing operations say

1. Modifying the SPJA constructs
2. Modifying the K value
3. Modifying the Weight value
4. Modifying all

The user specifies the type of modification to be done on the query.

4.1 Modifying the SPJA constructs

The SPJA constructs can be modified either by changing the constant value in the where clause or by adding a selection predicate. The algorithm checks if the expected tuple is present in the query result . If not, the constant value in the where clause is modified until the query result has the expected tuple.The expected tuple may or may not occur since it is based on other conditions like k-value and w-vector.

4.2 Modifying the k-value

In this case, the k- value is modified until the expected tuple is included in the query result. k-value determines the number of tuples that has to be included in the result. k-value can take the maximum value of total number of tuples in the dataset.

4.3 Modifying the weight value

The weight value determines the preferences for each case i.e, SPJA constructs, k-value and w-vector. The weight value ranges from 0.1-0.9. The weight value is modified until the expected the expected tuple appears in the query result.

Choice	Preferences (SPJ, k-value, weightings)
Modify SPJA	Spj=0.1,k=0.45,w=0.45
Modify k -value	Spj=0.45,k=0.1,w=0.45
Modify w-value	Spj=0.45,k=0.45,w=0.1
Modify all	Spj=0.33,k=0.33,w=0.33

4.4 Modifying all

In this case, all the values (Spj constructs, k-value and weight value) are modified to get the expected tuple. The original query is modified until the expected tuple occurs in the result. The probability of getting the expected tuple is 1 in this case. This case is considered to be better than other cases because we are sure about getting the expected tuple in the resultant query.

Finally, the cost for getting the expected tuple and penalty for changing the original query is calculated and the refined query is returned. From the refined query, the user can identify which factor has eliminated the expected tuple and also why not the expected tuple did not appear in the original query result.

$$\boxed{\text{Penalty}= \lambda spj * \Delta spj + \lambda k * \Delta k + \lambda w * \Delta w}$$

where

λspj= the current value of spj
Δspj= difference between the current value and previous value of spj
λk = the current value of k
Δk= difference between the current value and previous value of k-value
λw = the current value of weight
Δw= difference between the current value and previous value of weight

5 ALGORITHM

Input:

Original query and the expected tuple

```
1: Obtain $QS^0$ and j value.
2: if $QS^0$ does not exist then
3: return “cannot answer the why-not question”;
4: end if
5: switch(choice)
6: case 1:
7:     SPJA constructs are changed.
8:     if Q_result equal to j then
9:             calculate penalty;
10:            if penalty <=min_penalty then
```

```
11:                 min_penalty = penalty;
12:         end if
13:  end if
14:         Return query with min_penalty
15:case 2:
16:  k value alone is changed
17:  if Q_result equal to j then
18:         calculate penalty;
19:         if penalty <=max_penalty then
20:                 max_penalty = penalty;
21:         end if
22:  end if
23:         Return query with min_penalty
24:case 3:
25:  weight values are changed
26:  if Q_result equal to j then
27:         calculate penalty;
28:         if penalty <=max_penalty then
29:                 max_penalty = penalty;
30:         end if
31:  end if
32:         Return query with min_penalty
33:case 4:
34:  all are changed randomly
35:  if Q_result equal to j then
36:         calculate penalty;
37:         if penalty <=max_penalty then
38:                 max_penalty = penalty;
39:         end if
40:  end if
41:  Return query with min penalty
```

5.1 INPUT TABLES

Table 1

ID	NAME
P1	Alice
P2	Bob
P3	Chandler
P4	Daniel
P5	Eagle
P6	Fabio
P7	Gary
P8	Henry

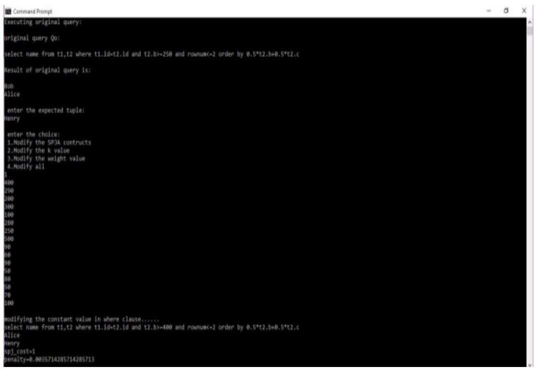
Table 2

ID	A	B	C
P1	90	400	80
P2	60	290	60
P3	90	200	100
P4	50	300	70
P5	80	100	210
P6	50	250	70
P7	70	280	50
P8	100	500	100

Table 3

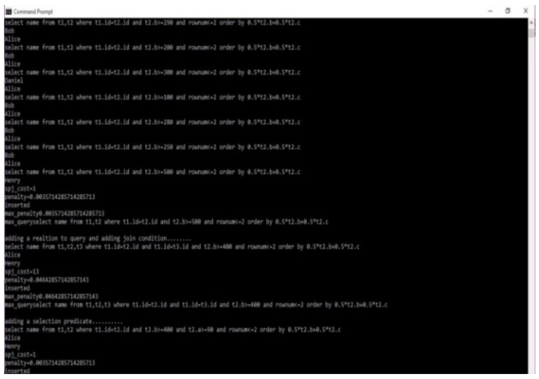
ID	D	E	F
P1	60	200	70
P2	100	250	90
P3	70	280	80
P4	90	300	90
P7	80	300	100
P8	60	200	60

5.2 OUTPUT



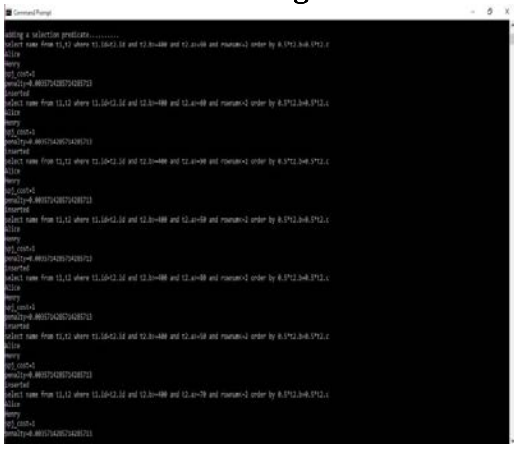
```
Executing original query:

original query Qo:

select name from t1,t2 where t1.id=t2.id and t2.b>=250 and rownum<=2 order by 0.5*t2.b+0.5*t2.c

Result of original query is:

Bob
Alice

 enter the expected tuple:
Henry

 enter the choice:
 1.Modify the SPJA contructs
 2.Modify the k value
 3.Modify the weight value
 4.Modify all
1
400
290
200
300
100
280
250
500
90
60
90
50
80
50
70
100

modifying the constant value in where clause......
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
```

Fig 1



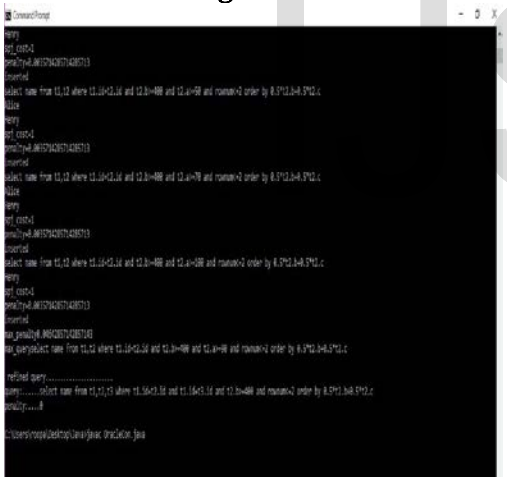
```
select name from t1,t2 where t1.id=t2.id and t2.b>=290 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Bob
Alice
select name from t1,t2 where t1.id=t2.id and t2.b>=200 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Bob
Alice
select name from t1,t2 where t1.id=t2.id and t2.b>=300 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Daniel
Alice
select name from t1,t2 where t1.id=t2.id and t2.b>=100 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Bob
Alice
select name from t1,t2 where t1.id=t2.id and t2.b>=280 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Bob
Alice
select name from t1,t2 where t1.id=t2.id and t2.b>=250 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Bob
Alice
select name from t1,t2 where t1.id=t2.id and t2.b>=500 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
max_penalty0.0035714285714285713
max_queryselect name from t1,t2 where t1.id=t2.id and t2.b>=500 and rownum<=2 order by 0.5*t2.b+0.5*t2.c

adding a realtion to query and adding join condition........
select name from t1,t2,t3 where t1.id=t2.id and t1.id=t3.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=13
penalty=0.04642857142857143
inserted
max_penalty0.04642857142857143
max_queryselect name from t1,t2,t3 where t1.id=t2.id and t1.id=t3.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c

adding a selection predicate..........
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=90 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
```

Fig 2



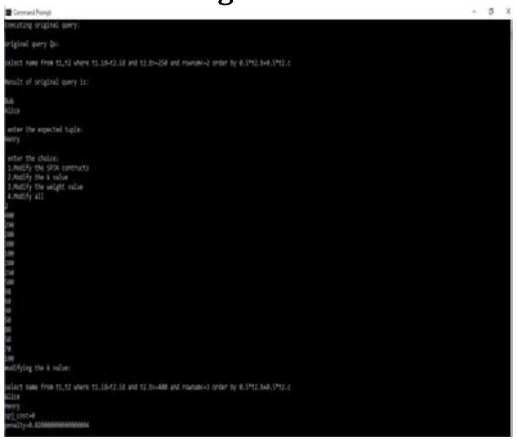
```
adding a selection predicate..........
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=90 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=60 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=90 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=50 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=80 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=50 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=70 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
```

Fig 3



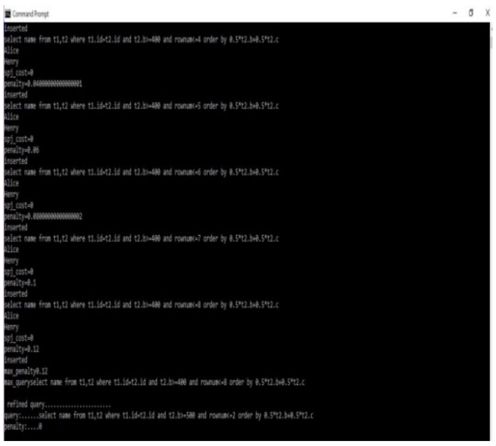
```
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=50 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=70 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=100 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
Henry
spj_cost=1
penalty=0.0035714285714285713
inserted
max_penalty0.04642857142857143
max_queryselect name from t1,t2 where t1.id=t2.id and t2.b>=400 and t2.a>=90 and rownum<=2 order by 0.5*t2.b+0.5*t2.c

 refined query.......................
query:......select name from t1,t2,t3 where t1.id=t2.id and t1.id=t3.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
penalty:....0

C:\Users\roopa\Desktop\Java>javac OracleCon.java
```

Fig 4



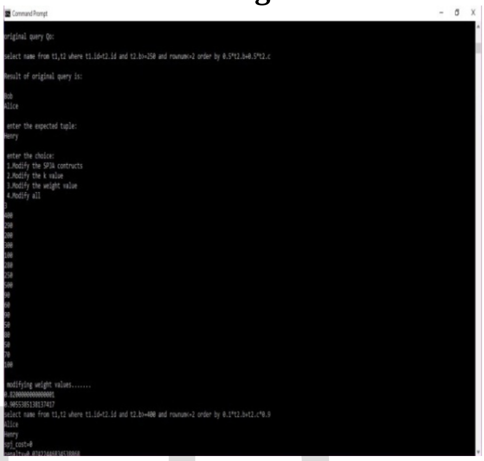
```
Executing original query:

original query Qo:

select name from t1,t2 where t1.id=t2.id and t2.b>=250 and rownum<=2 order by 0.5*t2.b+0.5*t2.c

Result of original query is:

Bob
Alice

 enter the expected tuple:
Henry

 enter the choice:
 1.Modify the SPJA contructs
 2.Modify the k value
 3.Modify the weight value
 4.Modify all
2
400
290
200
300
100
280
250
500
90
60
90
50
80
50
70
100
modifying the k value:

select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=3 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=0
penalty=0.020000000000000004
```

Fig 5



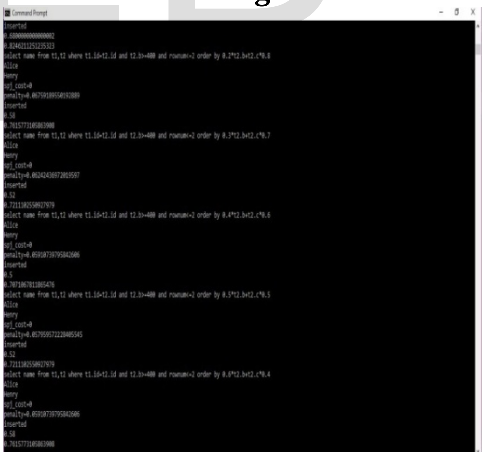
```
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=4 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=0
penalty=0.040000000000000001
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=5 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=0
penalty=0.06
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=6 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=0
penalty=0.080000000000000002
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=7 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=0
penalty=0.1
inserted
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=8 order by 0.5*t2.b+0.5*t2.c
Alice
Henry
spj_cost=0
penalty=0.12
inserted
max_penalty0.12
max_queryselect name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=8 order by 0.5*t2.b+0.5*t2.c

 refined query.......................
query:......select name from t1,t2 where t1.id=t2.id and t2.b>=500 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
penalty:....0
```

Fig 6



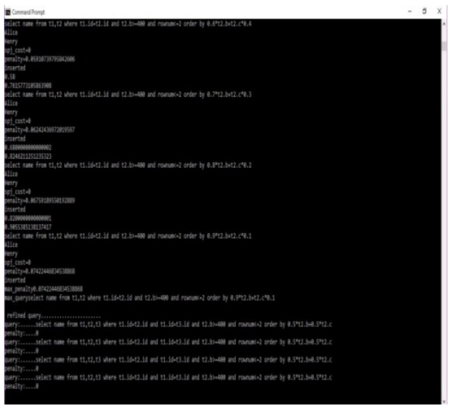
```
original query Qo:

select name from t1,t2 where t1.id=t2.id and t2.b>=250 and rownum<=2 order by 0.5*t2.b+0.5*t2.c

Result of original query is:

Bob
Alice

 enter the expected tuple:
Henry

 enter the choice:
 1.Modify the SPJA contructs
 2.Modify the k value
 3.Modify the weight value
 4.Modify all
3
400
290
200
300
100
280
250
500
90
60
90
50
80
50
70
100

 modifying weight values.......
0.8200000000000001
0.9055385138137417
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.1*t2.b+t2.c*0.9
Alice
Henry
spj_cost=0
penalty=0.07472468834538868
```

Fig 7



```
inserted
0.6800000000000002
0.8246211251235323
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.2*t2.b+t2.c*0.8
Alice
Henry
spj_cost=0
penalty=0.06759189550192889
inserted
0.58
0.7615773105863908
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.3*t2.b+t2.c*0.7
Alice
Henry
spj_cost=0
penalty=0.06242438972019597
inserted
0.52
0.7211102550927979
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.4*t2.b+t2.c*0.6
Alice
Henry
spj_cost=0
penalty=0.05910739795842606
inserted
0.5
0.7071067811865476
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+t2.c*0.5
Alice
Henry
spj_cost=0
penalty=0.05795957222805545
inserted
0.52
0.7211102550927979
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.6*t2.b+t2.c*0.4
Alice
Henry
spj_cost=0
penalty=0.05910739795842606
inserted
0.58
0.7615773105863908
```

Fig 8



```
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.6*t2.b+t2.c*0.4
Alice
Henry
spj_cost=0
penalty=0.05918739795842606
inserted
0.58
0.7615773105863908
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.7*t2.b+t2.c*0.3
Alice
Henry
spj_cost=0
penalty=0.06242438972019597
inserted
0.6800000000000002
0.8246211251235323
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.8*t2.b+t2.c*0.2
Alice
Henry
spj_cost=0
penalty=0.06759189550192889
inserted
0.8200000000000001
0.9055385138137417
select name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.9*t2.b+t2.c*0.1
Alice
Henry
spj_cost=0
penalty=0.07422446834538868
inserted
max_penalty0.07422446834538868
max_queryselect name from t1,t2 where t1.id=t2.id and t2.b>=400 and rownum<=2 order by 0.9*t2.b+t2.c*0.1

 refined query.......................
query:.......select name from t1,t2,t3 where t1.id=t2.id and t1.id=t3.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
penalty:....0
query:.......select name from t1,t2,t3 where t1.id=t2.id and t1.id=t3.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
penalty:....0
query:.......select name from t1,t2,t3 where t1.id=t2.id and t1.id=t3.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
penalty:....0
query:.......select name from t1,t2,t3 where t1.id=t2.id and t1.id=t3.id and t2.b>=400 and rownum<=2 order by 0.5*t2.b+0.5*t2.c
penalty:....0
```

Fig 9

6 CONCLUSION

Thus we have studied the problem of answering why-not questions on top-k SQL queries. Our target is to give an explanation to a user who is wondering why his/her expected answers are missing in the query result. We return to the user a refined query that can include the missing expected answers back to the result. The non-numerical values are internally converted into numerical values to search in the database.

7.REFERENCES

1. S.AGARWAL, S.CHAUDHURI,and G.DAS,"dbxplorer: A system for keyword-based search over rational databases", in PROC. 18th Int.Conf.Data eng.,2002,pp.5-16.
2. H.WU,G.LI,C.LI,AND L.ZHOU,"SEAFORM; search -as -you -type –in forms", in Proc.VLDB Endowment, 2010,vol. 3, no. 2,pp. 1597-1568.
3. J.Akbarnejad,G.chatzopoulou,M.EIRINAKI,S.KOSHY,S.MITTAL,D.ON,N.POLYZOTIS,AND J.S.V. VARMAN,"SQL QueRIE RECOMMENDATIONS", in proc . VLDB endowment, 2010,vol.3, no. 2,pp. 1597-1600.
4. M.B.N.Khoussainova, Y.C.kwon and DD.SUCIU,"snipsuggest: context-aware autocompletion for SQL," IN PROC. Vldb ENDOWMENT,2010,VOL.4,NO.1,PP.22-33
   A.CHAPMAN AND H.JAGADISH, "WHY NOT?" IN PROC . ACM SIGMOD int.conf.manage.data,2009,pp. 523-534.
5. J.Huang ,T.chen,A-H.DOAN,and j.f.naughton,"on the provenance of non-answers to queries over extracted data," in proc VLDDB, 2008,PP.736-747.
6. M.HERCHEL AND M.A.HERNANDEZ," EXPLAINING MISSING ANSWERS TO SPJUA QUERIES", IN PROC VLBD , 2010, PP.185-196
7. Q.T.TRAN AND C-Y.CHAN, "HOW TO CONQUER WHY NOT QUESTIONS", PROC.VLDB,2010,PP.15-26
8. Z.HE AND E.LO,"ANSWERING WHY NOT QUESTIONS ON TOP K QUERIES," IN PROC .IEEE 28TH INT.CONF.DATA ENG., 2012 PP 750-761
9. I.MD.SAiful,Z. RUI , AND L.CHENGFEI, "ON ANSWERING WHY NOT QUESTIONS IN REVERSE SKYLINE QUERIS",IN PROC.IEEE 28TH INT.CONF DATA ENG., 2013,PP.973-984.
10. Z.HE AND E.O, "ANSWERING WHY NOT QUESTIONS ON TOP K QUERY",IEEE TRANS KNOWL. DATA ENG., VOL.26,NO.6,PP.1300-1315,JUN.2014.
11. A.METRO,"SEAVE: A MECHANISM FOR VERIFYING USER PRE SUPPOSITIONS IN QUERY SYSTEMS", ACM TRANS. INF.SYST.,VOL 4,NO 4,PP 312-330,1986.
12. A.METRO,"QUERY GENERALIZATION: A METHOD FOR INTEPRETING NULL ANSWERS", IN PROC. 1ST INT INT EXPERT DATABASE WORKSHOP,1984,PP 597-616
13. F.ZAO, K-L T.G.DAS, AND A.K.H.TUNG," CALL TO ORDER ; A HIERARCHICAL BROWSING APPROACH TO ELICATING USERS "PREFERENCES?", IN PROC . ACM SIGMOD INT.CONF.DATA ENG.,2010,PP.365-376.

14. A.VLACHOU, C.DOULKERIDIS, Y.KOTIDIS, AND K.NORVAG,"REVERSE TOP K QUERIES ", IN PROC IEEE 28TH INT. CONF. DATA ENG ., 2010,PP 365-376