A PROJECT REPORT

ON

**“DATA PROVENANCE USING QUERY INVERSION”**

**SUBMITTED BY**

JAGTAP SHRIKANT NANDKUMAR (2013H112176P)

**GUIDED BY**

ASMA RANI



DEPARTMENT OF CSIS

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI – PILANI CAMPUS

RAJASTHAN – 333 031

**DECLARATION CERTIFICATE**

## This is to certify that the work presented in the project entitled "Data Provenance using Query Inversion" in partial fulfillment of the requirement for the award of degree of M.E in COMPUTER SCIENCE, Birla Institute of Technology & Science, Pilani is an authentic work carried out under my supervision. To the best of my knowledge, the content of this project or report does not form a basis for the award of any previous degree to anyone else.

Date :

Mrs. ASMA RANI

Guide Name & Signature

Dept.of Computer Science & Information System

Birla Institute of Technology & Science, Pilani

Rajasthan

**ABSTRACT**

The provenance of a piece of data, a so-called data item, includes information about the source data from which it is derived and the transformations that lead to its creation and current representation. In the context of relational databases, provenance has been studied both from a theoretical and algorithmic perspective. Yet, in spite of the advances made, there are very few practical systems available that support generating, querying and storing provenance information (We refer to such systems as provenance management systems or PMS). These systems support only a subset of SQL, a severe limitation in practice since most of the application domains that benefit from provenance information use complex queries. Such queries typically involve nested sub-queries, aggregation and/or user defined functions. Without support for these constructs, a provenance management system is of limited use. Furthermore, existing approaches use different data models to represent provenance and the data for which provenance is computed (normal data). This has the intrinsic disadvantage that a new query language has to be developed for querying provenance information. Naturally, such a query language is not as powerful and mature as, e.g., SQL. Different notions of provenance for database queries have been proposed and studied in the past few years. In this article, we detail three main notions of database provenance, some of their applications, and compare and contrast amongst them. Specifically, we review why, how, and where provenance, describe the relationships among these notions of provenance, and describe some of their applications in confidence computation, view maintenance and update, debugging, and annotation propagation.

**ACKNOWLEDGEMENT**

This research project would not have been possible without the kind support of many people. I take this opportunity to acknowledge them who have been great sense of support and inspiration thought the research project for its successful accomplishment.

I would express our profound thanks to **ASMA RANI** who guided me and helped me through theoretical aspects and spent her time for understanding the things related to this project and guiding me in all the difficulties that I faced during this research project.

I am also thankful to my parents, colleagues and friends for helping me whenever I needed them.

Finally, I would like to thank **CSIS Department** of **Birla Institute of Technology & Science, Pilani – Pilani Campus** for giving me this wonderful opportunity on working on this research project.

JAGTAP SHRIKANT NANDKUMAR  
2013H112176P

**TABLE OF CONTENTS**

[1 INTRODUCTION TO DATA PROVENANCE 4](#_Toc386723895)

[1.1 WHAT IS DATA PROVENANCE 4](#_Toc386723896)

[1.2 WHY ,HOW AND WHERE – PROVENANCE? 5](#_Toc386723897)

[1.2.1 WHY PROVENANCE 5](#_Toc386723898)

[1.2.2 HOW PROVENANCE 5](#_Toc386723899)

[1.2.3 WHERE PROVENANCE 5](#_Toc386723900)

[1.3 IMPLEMENTING DATA PROVENANCE 6](#_Toc386723901)

[1.4 HOW DOES DATA PROVENANC WORK 7](#_Toc386723902)

[2 IMPLEMENTATION 8](#_Toc386723903)

[2.1 USING QUERY INVERSION APPROACH 8](#_Toc386723904)

[2.1.1 DESIGN DATABASE SCHEMA 8](#_Toc386723905)

[2.1.2 MAINTAIN SOURCE QUERIES WITH TIMESTAMP 9](#_Toc386723906)

[2.1.3 ANALYZE SOURCE QUERY USING SQL PARSER 9](#_Toc386723907)

[2.1.4 REWRITE SOURCE QUERY 9](#_Toc386723908)

[2.1.5 EXECUTE THE REWRITTEN QUERY TO GET SOURCE TUPLES 13](#_Toc386723909)

[2.2 SQL PARSER 14](#_Toc386723910)

[2.3 DEMONSTRATION 15](#_Toc386723911)

[3 CONCLUSION 19](#_Toc386723912)

[4 REFERENCES 20](#_Toc386723913)

# INTRODUCTION TO DATA PROVENANCE

## WHAT IS DATA PROVENANCE

With the increasing amount of available storage space and the acceleration of information flow induced by the internet, a growing interest in information about the creation process and sources of data has developed. Scientists in the fields of biology, chemistry or physics use data from so-called curated databases. Most of the data stored in a curated database is a result of manual transformations and derivations. Researchers that use data from a curated database are interested in information about the sources and transformations that were applied to this data. This information can be used to assess the data quality, examine or reconsider derivation processes or re-run a specific experiment. Data warehouses are used to integrate data from different sources and with different data representations, and to analyze the integrated data. Such analyses could benefit from information about the original data sources and transformations used to create the data in the data warehouse.

*Provenance* information describes the origins and the history of data in its life cycle. Such information (also called *lineage*) is important to many data management tasks. Historically, databases and other electronic information sources were trusted because they were under centralized control: it was assumed that trustworthy and knowledgeable people were responsible for the integrity of data in databases or repositories. As argued by Lynch [49], this assumption is no longer valid for online data. Today, data is often made available on the Internet with

No centralized control over its integrity: data is constantly being created, copied, moved around, and combined indiscriminately. Because information sources (or different parts of a single large source) may vary widely in terms of quality, it is essential to provide provenance and other context information which can help end users judge whether query results are trustworthy.

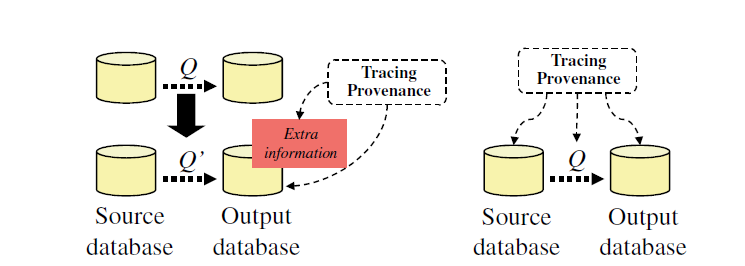


Fig1. DATA PROVENANCE

## WHY, HOW AND WHERE – PROVENANCE?

### WHY PROVENANCE

*Why-provenance* that captures the different witnesses. It restricts attention to a smaller number of witnesses. The witness basis can be viewed as a compact representation of the set of all witnesses.

### HOW PROVENANCE

One can derive the why-provenance of an output tuple from its *how-provenance* polynomial. Why-provenance describes the source tuples that witness the existence of an output tuple in the result of the query. However, it leaves out some information about *how* an output tuple is derived according to the query.

### WHERE PROVENANCE

*Where-provenance* describes the relationship between source and output *locations*. In the relational setting, a location is simply a column of a tuple in a relation, which precisely refers to a “cell” in a relation.

## IMPLEMENTING DATA PROVENANCE

”Use SQL to compute and query the provenance of SQL queries”. What do we mean by that? We envision a system that represents the results of a query alongside with its provenance in a single relation using a representation of this information that is directly interpretable by the user. Even more, provenance should not only be represented relationally, also the generation of this kind of information should be implemented as SQL queries. These goals were not chosen randomly, but are based on an analysis of the causes for the shortcomings of existing systems. As mentioned before these systems tend to use a different data model for provenance and normal information (The data stored in the database or generated by queries). This has the intrinsic disadvantage that it is not possible to query provenance and normal data using the same query language. Thus, the association between information and its provenance cannot be explored declaratively. Furthermore, a new query language has to be developed for provenance which naturally is not as sophisticated as a language like SQL that has been in use for several decades and constantly evolved over time. These problems can be avoided by representing provenance and normal data in the same data model and explicitly modeling the associations between normal and provenance data. The motivation for our goal to compute provenance as SQL queries is based on a similar argument. Using the same computational method for querying data and computing provenance allows for a tighter integration of provenance functionality into SQL. For instance, this allows to express a query over the result of a provenance computation as a single SQL statement by using the provenance computation as a sub-query.

## HOW DOES DATA PROVENANC WORK

To implement Data Provenance :

* Store fired query in Query Table with the help of TimeStamp.
* Create output table and link Query table and Output table.
* Select Output table for Data Provenance.
* Get the corresponding source tuple (Query String ,TimeStamp).
* Parse the source query string with the help of Sql Parser.
* Query Metadata as the output of Sql Parser.
* Use Query Metadata and according to query type rewrite the source query.
* Use TimeStamp constraint in the rewritten query to get correct input tuples.

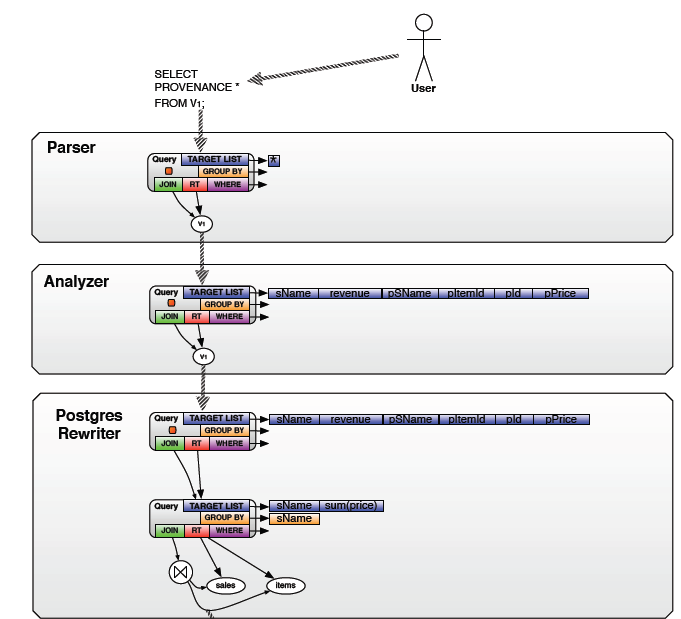


Fig2.Process of Data Provenance

# IMPLEMENTATION

## USING QUERY INVERSION APPROACH

### DESIGN DATABASE SCHEMA

Source Tables { Customer Table,Billpayment Table}

Output Table { output38}

Intermediate Table{q table} - Used for Provenance

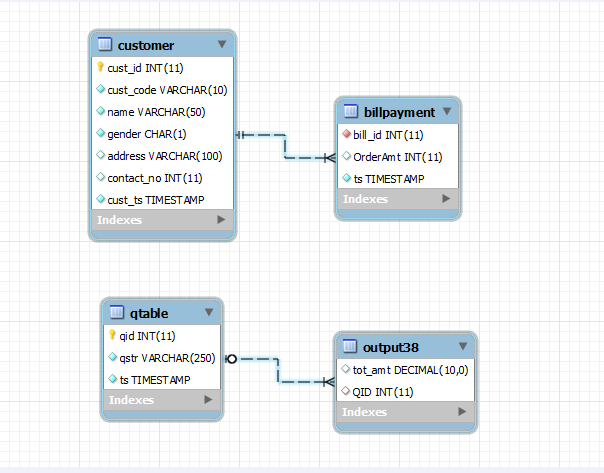


Fig3. ER Diagram

### MAINTAIN SOURCE QUERIES WITH TIMESTAMP

Details for Qtable

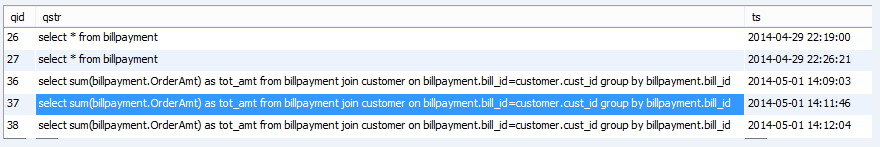


Fig4.Qtable

### ANALYZE SOURCE QUERY USING SQL PARSER

* Pass the input source query to the sql parser.
* Using the parser find the sql statements present in the query string.
* For each sql statement divide the query into different parts like
  + Select Part, From Part, Join Part, OnCondition Part, Group by part etc.
* Each part contains its associated list of table details and column details.
* Finally we obtain the query metadata which will be the input for query rewriting model.

### REWRITE SOURCE QUERY

**Rewrite Rules Translation**

A translation between SQL and relational algebra is needed to be able to implement the rewrite rules. Given such a translation, an SQL query can be rewritten by

1. translating it into relational algebra, applying the algebraic rewrites, and translating it back to SQL

OR

1. Using the mapping between relational algebra and SQL to translate the rewrite rules to

SQL

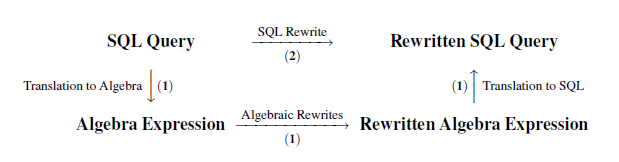


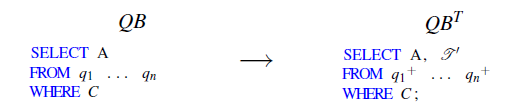
Fig5 Translation Diagram

The translation of the rewrite rules to SQL is more complex, but it has the advantage that no translation between algebra and SQL has to be applied at run-time. Therefore, we have chosen to implement the second approach. We first introduce an canonical translation between SQL and relational algebra. Afterwards, types of queries are classified regarding their provenance behaviour. Finally, we present how each query type is rewritten.

**Rewrite of SPJ Query Blocks**

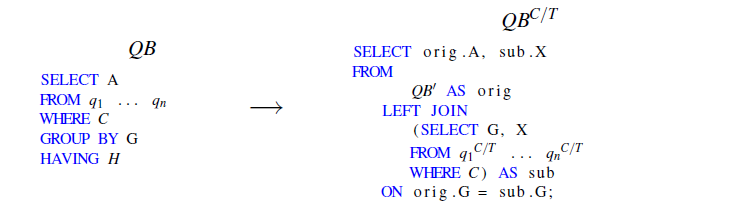
The general approach we follow for translating the algebraic rewrite rules to SQL is to first transform a query block QB into an algebra expression q, apply the rewrite rules for one of the provenance metaoperators to generate, e.g., q+, and finally transform q+ back into SQL, thus, generating a rewritten SQL statement QB+. In the implementation of the rewrite rules the direct translation from QB to QB+ is applied.

**SIMPLE SELECT FORMAT**



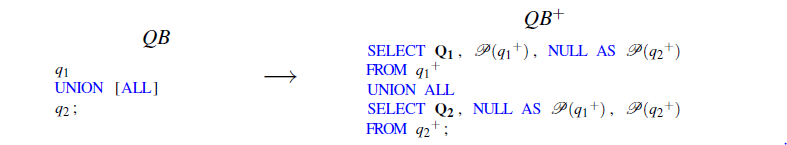
The modified provenance attribute list in this projection is denoted as 

**SELECT-GROUP BY- HAVING FORMAT**

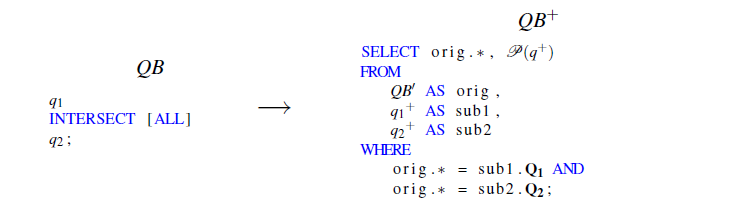




**UNION FORMAT**



**INTERSECTION FORMAT**



**Rewrite of SPJ-sub Query Blocks**

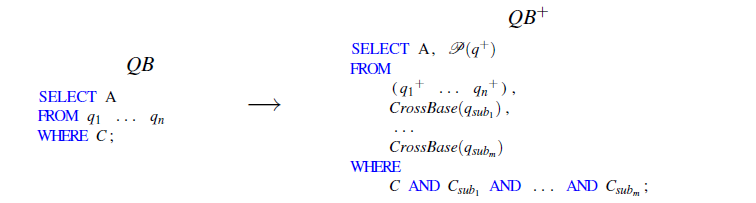
The algebraic representation q of an SPJ-sub query block QB is generated like for an SPJ query block except that each sublink in QB is translated into an algebraic sublink expression. The result of this translation is an algebra expression of the following format:



For a sublink query qsub over base relations R1 to Rn the CrossBase is expressed in relational algebra as:



The rewritten algebra expression q+ is translated into a query block for the rewritten SPJ part and one query block for each CrossBase using the SQL versions of Csubi and the CrossBase(qsubi ):



### EXECUTE THE REWRITTEN QUERY TO GET SOURCE TUPLES

Find the query on the target database and observe the source tuples to validate the provenance.

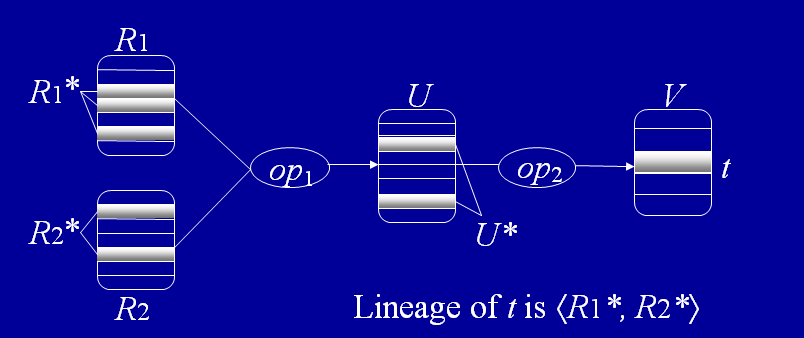


Fig.6.a Lineage Diagram

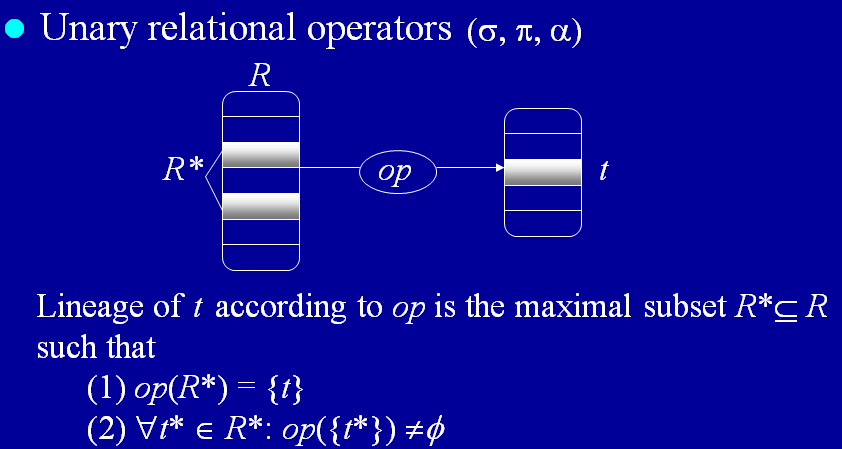


Fig.6.b Lineage Diagram

## SQL PARSER

It is used to analyze and get all the SQL statement details which are used for Query Rewriting.

Class Diagram :

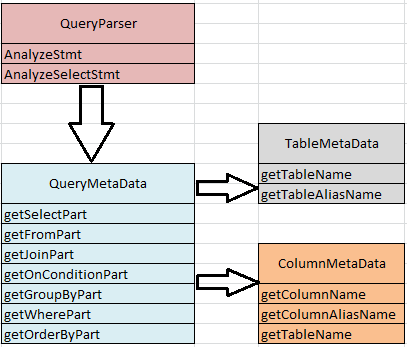


Fig7.SQL Parser Class Diagram

ALGORITHM :

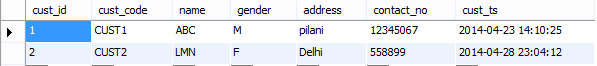
1. Get the input query to parse.
2. Get the list of SQL statements from input query.
3. Analyze statement to get the type of query.
4. Call AnalyzeSelectStmt if Query type is SELECT.
5. Get the TABLE LIST from the sql statement and store it in QueryMetadata.
6. Get the COLUMN LIST from the Select part of the sql statement.
7. Get the Tables used for JOIN.
8. Get the Columns used for OnCondition of JOIN.
9. Get the Columns from the Where Clause of the sql statement.
10. Get the Columns from the Group by Clause.
11. Return the QueryMetadata Object details to the Provenance Model.

## DEMONSTRATION

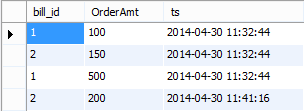
PREPROCESSING STEP

Create customer table and billpayment table for data provenance analysis.

Customer Table Details

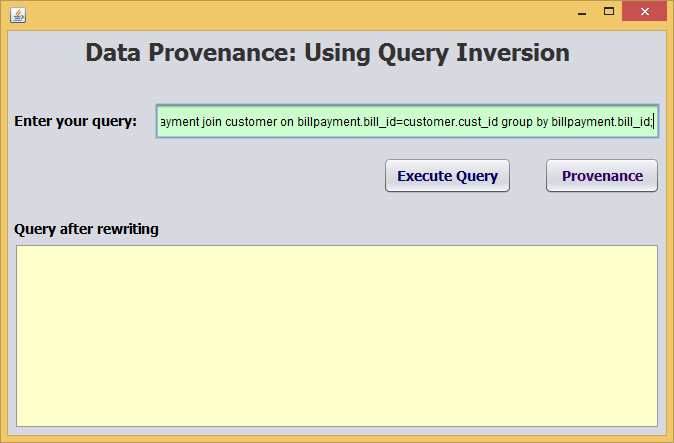


Bill Payment table Details



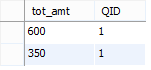
STEP 1:

User enters the query.



STEP 2

Output of the source Query



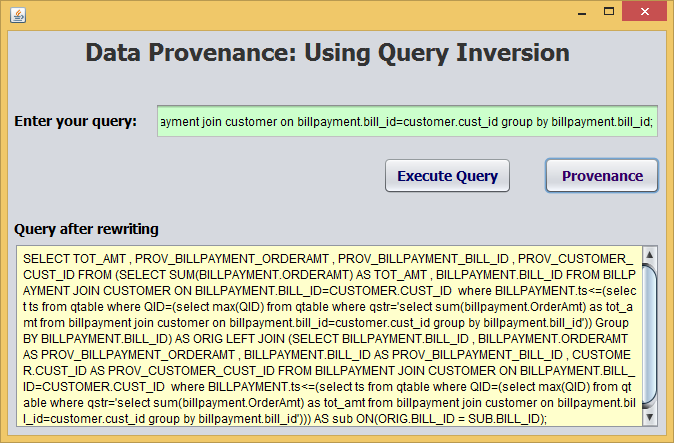
STEP 3

Store details of user input query with timestamp. Link QID of the inserted row with the Output table.

C:\Users\KIRAN\Desktop\RP\Snaps\qtable status.PNG

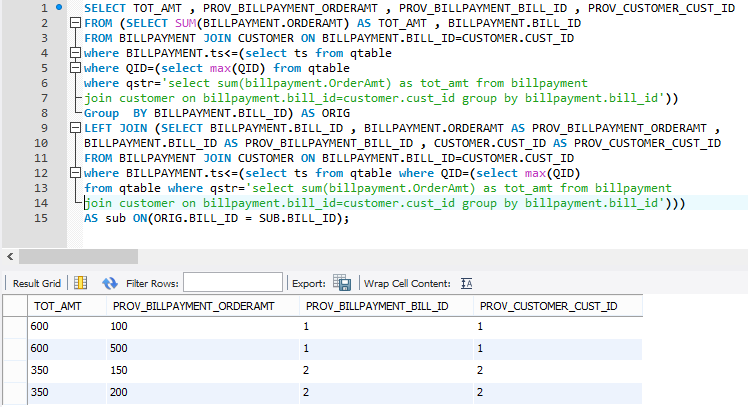
STEP 4

After pressing the provenance button, rewritten query is displayed.



STEP 5

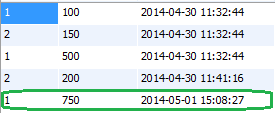
Output of the provenance query generated in previous step.



STEP 6

For validating the provenance query new tuple is added in the billpayment for CustID = ‘1’

CustID OrderAmt TS

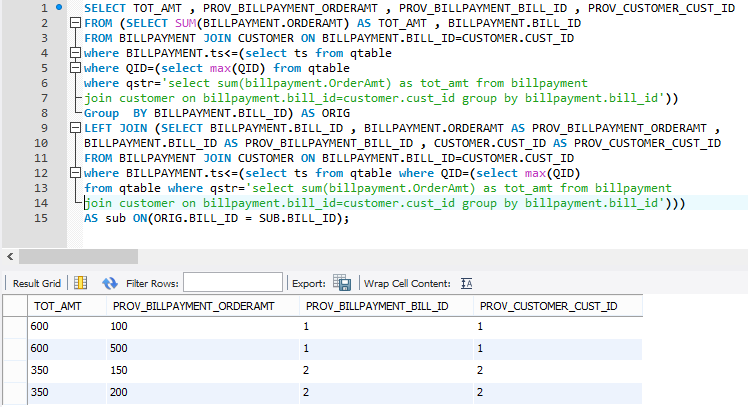


STEP 7

Repeat the step 5

RESULT :

Same as the previous provenance result.



# CONCLUSION

In this Research Practice we have implemented Data Provenance using a new query inversion approach. Here, we are storing only the user queries, unlike annotation approach in which we need to keep all the provenance details along with the query .Unlike annotation approach for each query we don’t need to do provenance processing every time. We do it only on user demand. SQL Parser is used to analyze the user query. Parser speeds up the query rewriting process to get the correct result for queries like ASPJ ,Simple where ,Union and Nested Select. This data provenance approach can be used in real time scenarios where response time for user query is important.

# REFERENCES

1. “Provenance in Databases: Why, How, and Where” by James Cheney, Laura Chiticariu and Wang-Chiew Tan
2. “On the expressiveness of implicit provenance in query and update languages,”by P. Buneman, J. Cheney, and S. Vansummeren,
3. “Perm: Efficient Provenance Support for Relational Databases” by Boric Glavic.
4. “Annotation – the new medium of communication” by Peter Buneman and Mark Steedman.
5. “Using SQL for Efficient Generation and Querying of Provenance Information” by

Boris Glavic,Renee J Miller

1. “Lineage Tracing In Data Warehouses” by Yingwei Cui.
2. “A Survey of Data Provenance Techniques” by Yogesh L. Simmhan, Beth Plale, Dennis Gannon
3. “Trio: A System for Integrated Management of Data, Accuracy, and Lineage” by Jennifer Widom
4. “A Comparation of Data Provenance Systems Based on Processing” by DAI Chao-fan, WANG Tao.
5. ” Provenance in modifiable datasets” by Jing Zhang