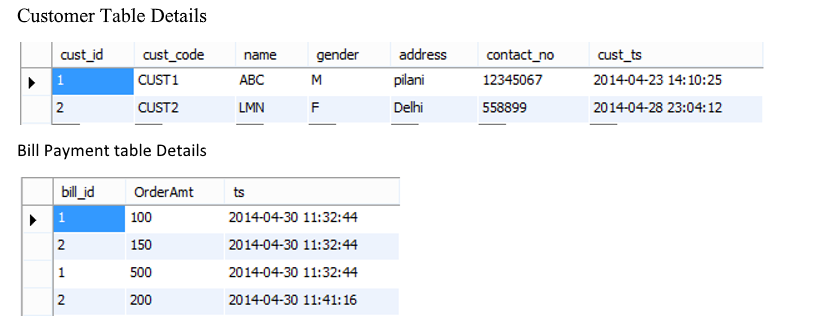
A：Data Provenance: Using Query Inversion.

github: https://github.com/shrikantnjagtap/Data-Provenance-using-Query-Inversion

1. **what ？**

*“Use SQL to compute and query the provenance of SQL queries.”*

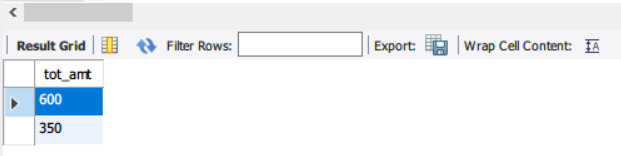
该项目做的主要工作是针对一次SQL查询，能够产生一个对应的SQL语句用来查询其起源数据。SQL查询的起源数据，指的是原始记录，由这些原始记录组合最终产生了SQL查询的结果。例如，我们有如下两个表：customer 和 billpayment表。表结构和内容如下：



我们输入如下的SQL语句，用来查询 customer1 和 customer2 订购的总数量：

select sum(billpayment.OrderAmt) as tot\_amt from billpayment join customer on billpayment.bill\_id=customer.cust\_id group by billpayment.bill\_id;

输出结果如下：



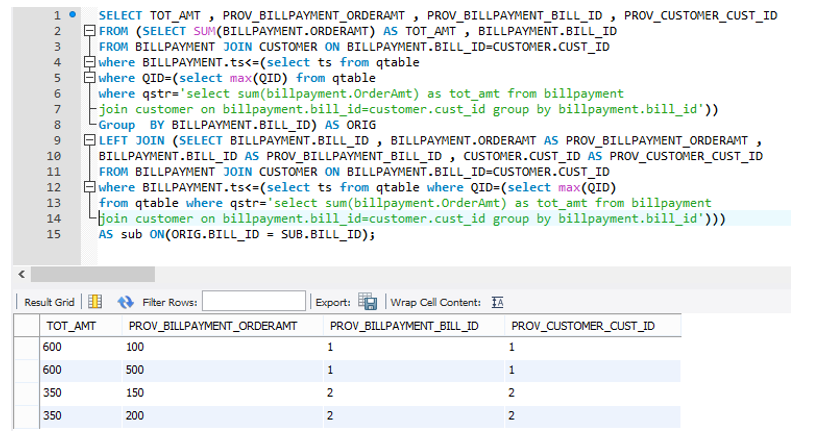
此处600和350是怎么来的，是从哪些记录产生的，这些记录就是上述SQL查询的起源数据。

该项目做的工作就是能够以一次SQL查询的语句作为输入，输出能够查询这次SQL查询结果的起源数据的SQL，我们执行输出的SQL就能拿到起源数据。

以刚刚的查询各个customer的总数量的SQL语句作为输入，输出的对应SQL语句如下：

SELECT TOT\_AMT , PROV\_BILLPAYMENT\_ORDERAMT , PROV\_BILLPAYMENT\_BILL\_ID , PROV\_CUSTOMER\_CUST\_ID FROM (SELECT SUM(BILLPAYMENT.ORDERAMT) AS TOT\_AMT , BILLPAYMENT.BILL\_ID FROM BILLPAYMENT JOIN CUSTOMER ON BILLPAYMENT.BILL\_ID=CUSTOMER.CUST\_ID where BILLPAYMENT.ts<=(select ts from qtable where QID=(select max(QID) from qtable where qstr='select sum(billpayment.OrderAmt) as tot\_amt from billpayment join customer on billpayment.bill\_id=customer.cust\_id group by billpayment.bill\_id')) Group BY BILLPAYMENT.BILL\_ID) AS ORIG LEFT JOIN (SELECT BILLPAYMENT.BILL\_ID , BILLPAYMENT.ORDERAMT AS PROV\_BILLPAYMENT\_ORDERAMT , BILLPAYMENT.BILL\_ID AS PROV\_BILLPAYMENT\_BILL\_ID , CUSTOMER.CUST\_ID AS PROV\_CUSTOMER\_CUST\_ID FROM BILLPAYMENT JOIN CUSTOMER ON BILLPAYMENT.BILL\_ID=CUSTOMER.CUST\_ID where BILLPAYMENT.ts<=(select ts from qtable where QID=(select max(QID) from qtable where qstr='select sum(billpayment.OrderAmt) as tot\_amt from billpayment join customer on billpayment.bill\_id=customer.cust\_id group by billpayment.bill\_id'))) AS sub ON(ORIG.BILL\_ID = SUB.BILL\_ID);

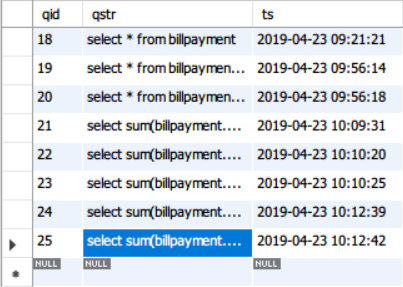
执行结果如下：（即600是由OrdAmt等于100和500两条记录产生而来，350由150和200两条记录而来。）



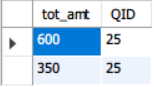
**2. How ?**

实现步骤：

1. 将每次查询的SQL字符串存在Qtable中，并且写入时间戳。



2. 针对每次查询结果产生一个output表存储结果，并用QID和Qtable关联起来。



3. 查询某次查询结果的起源数据时，首先根据QID在Qtable中拿到对应的SQL查询语句。

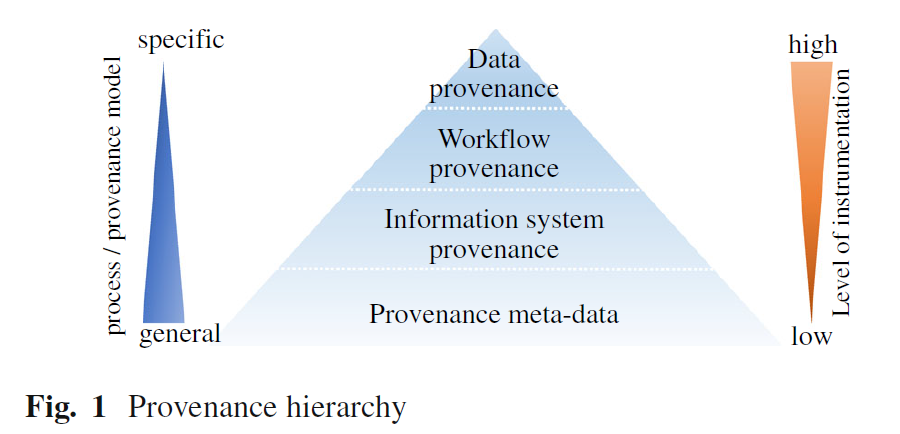
4. 利用开源工具（gsqlparser）对SQL语句进行分析。可以拿到：Select Part，From Part，Join Part，OnCondition Part， GroupBy part等等，每一个part都有对应的表信息或列信息。这样的信息组成了QueryMetaData。

5. 将QueryMetaData作为输入，根据 SQL 重写的一个规则进行起源数据SQL查询语句的生成（重写的规则涉及到数据库领域的Query Optimization，没有细看）

**3. Summary**

整个工作由于直接使用了数据库记录作为起源数据的数据结构，因而查询语言和数据结构都不用去重新实现，十分地方便。

但是工作集中在 Provenance 中的一个数据库应用层面。按照 ”A survey on provenance: What for? What form? What from?“（VLDBJ’2017）的分法该工作的Provenance是在一个特定的Information System层次。整个应用局限在结构化的数据库，对于NoSQL就不适用了。



B: FramworkProvW3C-NoSQL

github:https://github.com/tassioferenzini/FrameworkFProvW3C-NoSQL?tdsourcetag=s\_pctim\_aiomsg

整个项目做的工作主要是对W3C官方定义的Data Model（https://www.w3.org/TR/prov-dm/）进行了一个到JAVA代码的映射，包括所有的RDF关系，RDF实体等。所有的关系实现了get ，set，toString的函数方法。可以考虑在后续的工作中作为基础类直接使用。

