Removing Data Staleness in Data Warehouse Using Trigger Based Approach

Bikramjit Pal¹, Mallika De²

¹Assistant Professor and Head, Department of Computer Application, JIS College of Engineering, Kalyani, Nadia, West Bengal, INDIA
²Department of Engineering and Technological Studies, University of Kalyani, Kalyani, Nadia, West Bengal, INDIA

ABSTRACT

For storing a huge amount of data, data warehouse is in high demand as it supports large memory. But due to growth of storing data and its complexity, it is necessary to keep data warehouse fresh in regular intervals. Staleness factor is one of the key to keep fresh data in data warehouse. In this paper we tried to find out the staleness factor by tracking the number of times a database has been accessed in a given time interval.

Keywords: Data warehouse, complexity, staleness factor, database

I. INTRODUCTION

A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. By subject-oriented we mean, a data warehouse can be used to analyze a particular subject area, for example, "sales" can be a particular subject; by integrated we mean, a data warehouse integrates data from multiple data sources, for example, source A and source B may have different ways of identifying a product, but in a data warehouse, there will be only a single way of identifying a product; by time-variant we mean historical data is kept in a data warehouse, for example, one can retrieve data from 3 months, 6 months, 12 months, or even older data from a data warehouse; by non-volatile we mean once data is in the data warehouse.

Stale Data: In computer processing, if a processor changes the value of an operand and then, at a subsequent time, fetches the operand and obtains the old rather than the new value of the operand, then it is said to have seen stale data. On a uniprocessor, stale data cannot be tolerated. It would mean that the processor violated fundamental expectations about its own behavior. On shared memory multiprocessors, however, it is considered acceptable for machines to generate stale data on operands shared between processes. For such operands, the expectation is that programs will take precautions (atomic instructions or critical section routine) to prevent stale data from being seen.

Staleness Factor: Staleness factor may be considered as a range of defining a data to be stale or to be fresh. For example if staleness factors is denoted by 'S' and for a certain data S = 0 then it considered the data is fresh and if S>0 then the data is stale and need to be remove from data warehouse.

II. LITERATURE SURVEY

There are numerous works on data caching and synchronization, including those that consider system-driven data freshness issues. We got lots of information regarding staleness data in data warehouse from articles of Wikipedia. Various papers have been studied which are given in reference. particularly the algorithm to minimize stale data in warehouse. In the paper Scheduling to Minimize Staleness and Stretch In Real-Time Data Warehouses by Mohammad Hossein Bateni, Lukasz Golab AT&T Labs–Research and Mohammad Taghi Hajiaghayi AT&T Labs–Research proposed an algorithm to minimize staleness as their first objective. Their second objective is to limit the maximum “stretch”, which we define (roughly) as the ratio between the duration of time an update waits till it is finished being processed, and the length of the update. In contrast to earlier work proving the nonexistence of constant-competitive algorithms for related scheduling problems, they prove that any online no preemptive algorithm, no processor of which is ever voluntarily idle, incurs staleness at most a constant factor larger than an obvious lower bound on total staleness provided that the processors are sufficiently fast.

Another paper Defining and Measuring Data-Driven Quality Dimension of Staleness by Oleksiy Chayka, Themis Palpanas, and Paolo Bouquet provides a definition of a data-driven notion of staleness for information systems with frequently updatable data. For
such a data, we demonstrate an efficient exponential smoothing method of staleness measurement, compared to naïve approaches, using the same limited amount of memory, based on averaging of frequency of updates.

Moreover Bouzeghoub and Peralta studied freshness-related metrics; they have concentrated on analysis of definitions of data freshness in literature. We study those papers and other articles and try find out a probable way to find out the stale data from a data base and how to remove these stale data from the database.

III. PRESENT WORK

In order to find the stale data we need to find out the database or table which was accessed most for a given interval time say about month or week. The data which has list number of access can be consider as stale data and this data should be remove from database or data ware house. The best way to find the number access of a data or database or table is to fire a trigger. Consider the below trigger

```
Create table user_logins (userid varchar(30),
logon_month number, logon_cnt number);
Then create or replace trigger catch_logins after logon on
database:
declare curr_month number(6);
cnt_month number(8);
begin
select to_char(sysdate,'YYYYMM') into curr_month from
dual;
begin
select count(*) into cnt_month from user_logins where
userid = user and logon_month = curr_month;
if cnt_month = 0 then
Insert into user_logins values (user, curr_month, 1);
Else
update user_logins set logon_cnt = logon_cnt + 1 where
userid = user and logon_month = curr_month;
end if;
end;
```

Now as soon as a database is login or the trigger is fired and we can get the date when it was last accessed. From this date or time we can easily say the data or the database is fresh or not. We can use these type of trigger to find number of time a particular data or table is being access. After we the number of times a data is being access we can measure the freshness of a data according to the some factor and metric

1) Generally the freshness of a data is measured according to the following factor and metrics:

**Currency:** The time pass since data was extracted from the source, i.e. suppose a data was extracted from a database at some time but since then that data was not being extracted till current time so we consider it to be a stale data.

**Obsolescence:** More the number of update, transaction, operation is done on a data more the data is fresh.

**Fresh rate:** The percentage of tuples in the view that are up-to-date (have not been updated since extraction time).

**Timeliness:** The time pass since the last update is done to the source of the data, i.e. the time difference between the query time and last update time.

2) Requirements For Time-Related Quality Metrics

**NORMALIZATION**

One may need normalization of a data quality metric while operating with different metrics. Data values normalization in this context means mapping of all possible measurement values of a data quality metric to interval $[0,1]$. For example, consider a system where each user must change own age every 1 year. Those ages without been updated during more than 1 year, are not valid in the system, and can be treated as absolute stale elements.

**INTERVAL SCALE**

To support interpretability of quality measurement results for their comparison with measurements of other dimensions, those results should support interval scale property. This means that the same interval should denote the same quality improvement or degradation.

By This property we can say that difference of an element’s staleness between values 5 and 6 (days) means the same as the one between 8 and 9 (days).

**INTERPRETABILITY**

Easiness of interpretability of measurement results by end-users is also important for definition of a quality metric. For example, when an analyst has data with freshness metric equals to 0, does it mean to have fresh data at hand? What about freshness equals to 10 (suppose, if a stick to the notion is proposed) Is it even fresher? Unless specific notion of freshness is communicated to the end-user, interpretation of that may be ambiguous. To reduce such an ambiguity, we came with a notion that comprehends time-related characteristics of data, simplifying its perception by end user. with staleness $S=0$ we speak about absence of (a time-related) negative feature, while $S<0$ clearly indicates problems with data.

**AGGREGATION**

While measuring quality metric at one level, it is important to get aggregate value at higher one(s). For example, having result of staleness measurement for attribute “age”, how it will influence staleness of a corresponding entity, table and entire database?

**ADAPTIVITY**

Usually, to interpret measurements of quality metrics, those metrics should be adopted for a given context. While this is true only for some metrics, as [5] noted, most of them are context-independent and can be objectively evaluated without such an adaptation.

By definition, staleness is one of those objective metrics. However, as we have mentioned before, one can enforce adaptability of low-level measurement results for higher-level DQ assurance goals. For example, data
administrator may set a warning if attribute’s staleness reaches a certain threshold, and may set an automated request for update if staleness will reach even higher threshold.

FEASIBILITY

Techno-economical requirements of applications where quality measurement takes place, imply feasibility of getting the results. For example, getting a measure of reputation of an external source may be infeasible in some cases.

As we will show in the next section, our approach for getting staleness measure relies on parameters that are essential and normally easy to get for a data element at a source system – total number of updates, timestamps of first and last update, etc.

3) Enhanced Averaging Method

Consider a data element that has nearly periodic updates committed by users of an information system. We call such updates “real” ones. To measure data staleness of element at current time (or another test point), we need to have an instant of time of predicted update, that most probably should have taken place for in the period between instant of last update and current time.

For periodically updatable data element, this task seems to be straightforward. Number of all updates committed to a data element up to the test point, gives us an average update rate during past time interval. By adding (extrapolating) one more period, we can have an instant of potential update, and the staleness.

IV. CONCLUSION

In this paper we analyzed the problem of data staleness by introducing a trigger approach to find the accessibility time period of a database in data warehouse. But most important is that data warehouse administrator must have a option to remove these stale data and maintain the data quality in the data warehouse. And it can be consider as future work in order make the warehouse stale data free. Having staleness measurement mechanism of data at hand, one can either communicate the staleness level of data elements, or set necessary synchronization techniques with external data sources in such a way that own data would satisfy requirements imposed by a time-related quality dimension identified for an application.

REFERENCES