Near Real-time Data Warehousing with Multi-stage Trickle & Flip

Janis Zuters

University of Latvia, 19 Raina blvd.,
LV-1586 Riga, Latvia
janis.zuters@lu.lv

Abstract. A data warehouse typically is a collection of historical data designed for decision support, so it is updated from the sources periodically, mostly on a daily basis. Today's business however asks for fresher data. Real-time warehousing is one of the trends to accomplish this, but there are a number of challenges to move towards true real-time. This paper proposes 'Multi-stage Trickle & flip' methodology for data warehouse refreshment. It is based on the 'Trickle & flip' principle and extended in order to further insulate loading and querying activities, thus enabling both of them to be more efficient.

Keywords: Real-time data warehousing, data refreshment, data loading

1 Introduction

A data warehouse (DW) is a collection of technologies aimed at enabling the knowledge worker (executive, manager, and analyst) to make better and faster decisions. Traditional online transaction processing (OLTP) systems are inappropriate for decision support. Data warehousing has become an important strategy to integrate heterogeneous data sources and to enable online analytic processing (OLAP) [1].

Traditionally a data warehouse is refreshed periodically (e.g. weekly, daily) and can be considered as a window on the past. Until recently, using periodically updated data was not a crucial issue. However, with enterprises such as e-business, stock brokering, online telecommunications, and health systems, for instance, relevant information needs to be delivered as fast as possible to knowledge workers or decision systems who rely on it to react in a near real-time manner, according to the new and most recent data captured by an organization's information system [2]. This makes supporting near real-time data warehousing a critical issue for such applications.

Today's business demands are trying to gradually eliminate the inscription for history. Technically, one of the main roles a data warehouse is to separate the analytical part of the system from the operational one, in order to satisfy heavy performance demands in both of them. Providing a data warehouse with ever fresher data makes this task more and more difficult – a frequent transportation of data from sources to data warehouse increases the collision risk between data loading and querying activities.

Real-time warehousing is an evitable stage of evolution of data warehouses. Unfortunately this involves a lot of challenges and trade-offs, especially those with regard to state-of-the-art technologies; the most typical ones are faced when data refreshment can no longer be postponed to off-peak hours.

On the one hand, end-users of data warehouses need ever fresher data; on the other hand this involves additional requirements to hardware and possibly changes to data analysis behaviour due to frequent loadings. Because of more frequent and more complex operations related to data loading, OLAP performance could degrade dramatically.

Real-time data warehouses aim for decreasing the time it takes to make decisions and try to attain zero latency between cause and effect for that decision, closing the gap between intelligent reactive systems and systems processes. [3]

Moving towards *real-time* involves a number of issues/requirements:

– When loading data (near) continuously in real-time, there can't be any (transactional) system downtime [4];
– By loading data continuously (more frequently than daily), Query/OLAP downtimes and other inconveniences should be taken into account;
– The system becomes more complicated.

This makes the triple trade-off for the whole system for the sake of real-time:

– Data extraction should be as lightweight as possible;
– OLAP activities shouldn't be much affected due to more frequent loading;
– Data warehouse designers/users shouldn't suffer a lot from additional complicacy of the system.

To solve this, a good deal of real-time approaches, methodologies and technologies have been proposed in each of the stages, int. al. ETL (extract-transform-load), data modelling, and data analysis.

The proposed approach of 'Multi-stage Trickle & flip' lies beyond actual data transportation issues between the source systems and the data warehouse (e.g. change data capture, CDC) and assumes conventional ETL adapted to get near real-time warehousing put into effect.

2 Related Work

In general, the principle of 'real-time' is approached in several ways – from technologies of data transportation to DW architectures and OLAP query issues.

One of the focuses is that of true real-time data transportation between the operational systems and the data warehouse. Such technologies, approaches and protocols, concerning data transport and integration, lie beyond the scope of this paper. Here well matured ETL systems are assumed, and the described approaches address the aspects of DW affecting structures and manipulations over the data after they have been loaded from the sources.

Traditionally an ETL system (and thus DW refreshment, in general) works in a *batch mode*, but *continuous ETL* technologies are also being evolved.

Near real-time data warehousing addresses the challenge of need for fresh data by simply shortening the data warehouse refreshment intervals and hence, delivering source data to the data warehouse with lower latency [5]. This approach is referred to as near real-time data warehousing or *microbatch* ETL [6]. In contrast to “true” real-time solutions this approach builds on the mature and proven ETL system and does not require the re-implementation of the transformation logic.

A workable solution to it is *near real-time ETL*, where the loading frequencies are simply increased without any other changes on the system (Fig. 1). This is the cheapest and the easiest way to solve the problem and is a good choice for relatively small data warehouses until off-peak hour loadings become an issue.

As refreshment schedule is changed, additional side effects appear. One is that of refreshment anomalies during the ETL process, addressed in [5].

If the refreshment rate exceeds microbatch basis and loading is going to become continuous, this will require new approaches and technologies lying beyond the scope of this paper.

The ‘Trickle and Flip’ approach helps avert the scalability issues associated with querying tables of a data warehouse that are being simultaneously updated. [4] Here, instead of continuous (or very frequent microbatch) loading of data directly into warehouse tables a staging area is used. To make things easier, the staging tables are exactly of the same format as the target tables. Applying the principle of ‘Trickle & Flip’, on a periodic basis the staging tables are duplicated and the copy is swapped with the data warehouse tables. For smaller data warehouses the staging tables can contain a complete copy of all data while for bigger ones a typical case would be the data for the current day being put into a separate partition.

[3] proposes an integrated data warehouses loading methodology that covers as many as four different areas of operation: (a) data warehouse schema adaptation, (b) ETL loading procedures, (c) OLAP query adaptation, and (d) DW database packing and reoptimization. As previously, here also the principle is used to have the same format for temporary tables as of the DW.

3 Trickle & Flip

This Section goes deeper into the ‘Trickle & Flip’ approach, as further in this paper this particular principle is being evolved.

If using ‘Trickle & Flip’, only very small data warehouses can do without a real-time partition (Fig. 2). Applying the principle of ‘Trickle & Flip’ with real-time partitions, on a periodic basis the staging tables are duplicated and the copy is swapped with the real-time tables (Figs. 3 and 4).

When used, real-time partitions imply additional requirements to the system. A real-time partition must [7]:

– Contain all the activity occurred since the last update of the static data;
– Link as seamlessly as possible to the static data tables;
– Be so lightly indexed that incoming data can be continuously fed in;
– Support high performance querying.

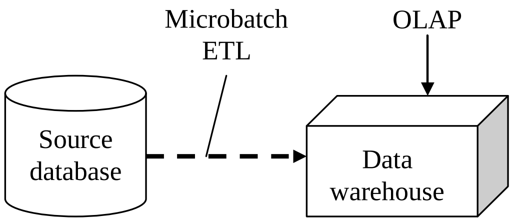


Fig. 1. Near real-time warehousing with near real-time ETL

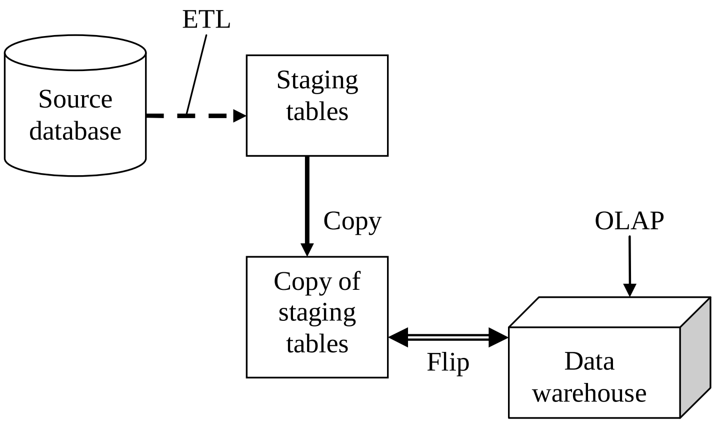


Fig. 2. Data loading using 'Trickle & Flip' with complete copy of data in the staging tables. Typical cycle times range from hourly to every minute or even faster

In particular, the latter two requirements are competing ones and thus potentially leading to awkward trade-offs or workarounds.

If the integration of real-time partition is implemented through views, the swap operation would only consist of changing the view definition. Even though flipping operation is a lightweight one, most likely it might be advisable to temporally pause the OLAP server.

In general, integrating real-time partitions into a system is very a complex task in terms of engineering. The success is largely up to ability of the query tool to encapsulate this complexity and to hide it from end-users.

A separate real-time data partition is a valuable technique to reduce loading/querying conflicts, yet at each flip (and this is to occur many times a day) it is advisable to restrict querying. If the warehouse is advancing towards real-time, the cycle times can be as frequent as of several minutes, and this would put a heavy burden to the end-users working on real-time data.

In this Section and further, it is assumed that the real-time data is loaded into the static data warehouse on a daily basis (in peak-off hours), so this particular loading is not covered by the described refreshment methodologies.

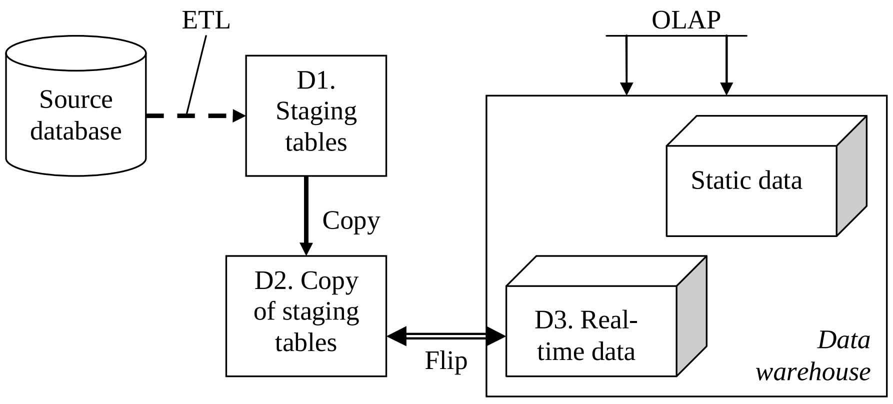


Fig. 3. Data loading using 'Trickle & Flip' with a separate real-time partition. Real-time data will typically contain the data of the current day only (A letter 'D' in D1, D2, and D3 stands for 'day'), thus all the three partitions (D1, D2, and D3) contain full data from the beginning of the current day with potentially different latencies

```
ALGORITHM trickle_and_flip_refresh (R)
    D3 – real-time partition of the DW (Fig. 3)
    D1, D2 – staging partitions with the same data format as D3 (Fig. 3)
    R – refreshment rate (e.g., 1 hour)
    D1 is being continuously (or in a microbatch mode) fed from the source
BEGIN
    Do Every R  % e.g., every hour
        Copy D1 to D2  % i.e., do backup of D1 into D2
        Flip D2 and D3  % D3 should not be locked by querying
```

Fig. 4. DW refreshment algorithm using 'Trickle & Flip' (related to Fig. 3). In off-peak time, real-time data is added to the static DW and the staging tables are emptied.

Summary of the 'Trickle & Flip' approach (assuming the refreshment rate to be one hour):

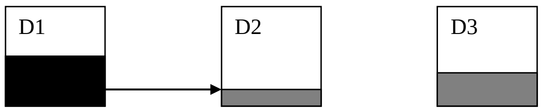
– All the three partitions contain approximately the data from the beginning of the day up to now;
– Normally, the data of the real-time partition is not older than 1 hour;
– Each hour the process of duplicating the data of the current day is being performed;
– Each hour it is advisable to restrict querying (on real-time data)

The issues of the 'Trickle & Flip' approach:

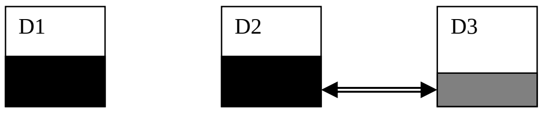
– For very large DWs, copying the full day data every hour could matter;
– Real-time querying being impeded every hour;
– By reducing the refreshment rate, the issues get heavier.

The ‘Multi-stage Trickle & Flip’ approach, proposed in the next Section, applies additional intermediate real-time areas in order to aver the listed drawbacks of the ‘pure’ ‘Trickle & Flip’. The main focus is put on reducing the refreshment rate to get closer to real-time.

D1 is being fed from the source and has the most fresh data



Copy D1 to D2



Flip D2 and D3

Now the real-time partition D3 has fresh data; normally, the latency is not greater than of the actual refreshment rate

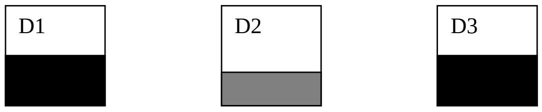


Fig. 5. An example. One iteration of a ‘Trickle & Flip’ loading (according to Fig. 4). D1 and D2 are staging areas, D3 is the real-time part of the DW

3 Multi-stage Trickle & Flip

3.1 The Main Idea

For the pure ‘Trickle & Flip’, if the cycle times are as frequent as of several minutes, this could put a heavy burden to querying, so to the end-users working on real-time data.

To overcome this, we propose ‘Multi-stage Trickle & Flip’ approach (Fig. 6) by adding additional intermediate stages to the system, thus mitigating the competition between loading and querying processes. In particular, real-time data is divided into two (or potentially more) subpartitions, each one with a different latency (and thus, also amount of stored data).

The main objective of the new approach:

– Collisions between data loading and querying activities should be reduced.

3.2. Setup and Operation

Evolving the approach of 'Trickle & Feed' is based on the following assumptions:

– Adding data to a smaller table (i.e., with less data) is faster;
– Updating last changes to a table is faster than making full copy of the last version.

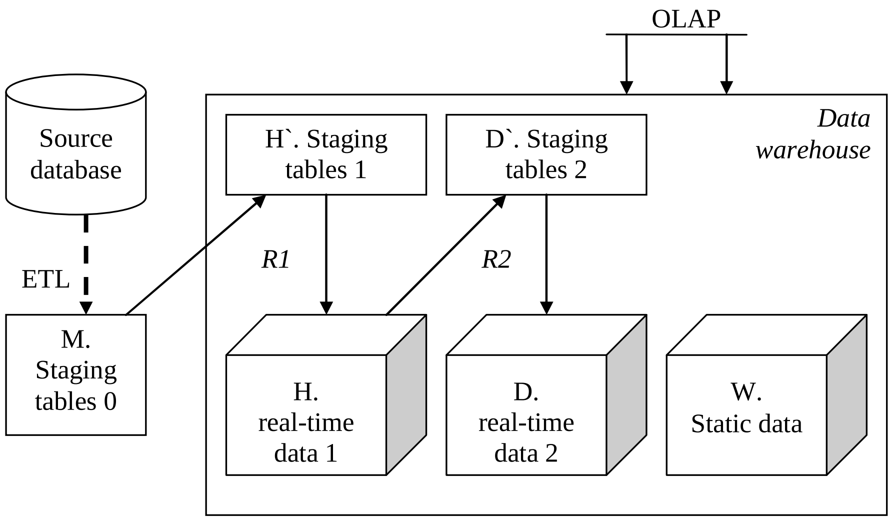


Fig. 6. Multi-stage Trickle & Flip with one intermediate level. In partition names, a letter 'M' stands for minute, 'H' stands for hour, 'D' stands for day. Data warehouse consists of partitions H, D, and W, while M, H`, and D` are staging partitions. R1 – the first stage refreshment rate, e.g., 5 minutes, R2 – the second stage refreshment rate, e.g., 1 hour

Construction of 'Multi-stage Trickle & Flip' infrastructure (Fig. 6):

– DW is represented by static data (W) and two (or potentially more) real time partitions (D and H). Each of real time partitions is designed to contain a different degree of amount of data (e.g., of the last day or the last hour);
– The staging area M is continuously fed with the source data;
– There are two additional staging areas (for each of the real-time partitions) – D` and H` which are not affected by querying, thus always available for loading activities;
– For both stages, refreshment with two different refreshment rates is performed (See the algorithm in Fig. 7 for the first stage refreshment: the same algorithm suits second stage refreshment with parameters *R2, H, D`, D*). For simplicity of description, it is assumed that refreshment rates R1 and R2 are 5 minutes and 1 hour respectively;
– There are 3 possible levels of querying: 1) W, 2) W+D, 3) W+D+H (i.e. of latencies of 1 day, 1 hour, or 5 minutes respectively).

An example of refreshment with this methodology is given in Fig. 8.

3.3. Benefits and Issues

The ‘Multi-stage Trickle & Flip’ approach reduces amount of data to be copied and possible collisions between loading and querying activities. A brief comparison between the two approaches is given in Table 1.

```
ALGORITHM multiple_trickle_and_flip_refresh (R1, M, H`, H)
    H – real-time partition for the current hour
    M, H` – staging partitions with the same data format as W (Fig. 6)
    R1 – first stage refreshment rate (e.g., 5 minutes)
    M is being continuously (or in a microbatch mode) fed from the source
BEGIN
    Do Every R1  % e.g., every 5 minutes
        Add M to H`
        Empty M
        If H is available  % not locked by querying
            Add H` to H
            Empty H`
```

Fig. 7. The first stage DW refreshment algorithm using ‘Multi-stage Trickle & Flip’ (related to Fig. 6). *H* will typically contain the data of the current hour (with latency of 5 minutes). *H`* will typically be empty unless *H* is not available for some time – in this case *H`* would contain one or several portions of 5 minutes data. The algorithm is visualized in Fig. 8. Exactly the same algorithm suits the second stage, but with parameters *R2, H, D`, D* respectively.

The summary obtained features of the new approach:

– Total amount of data copying is reduced;
– Collisions between data loading and querying activities have been reduced;
– By advancing the querying system (e.g., balancing needs for queries with a certain latency, separate caching for each stage) these collisions would have been more reduced;
– The approach is open to additional stages. Having appropriate querying tools available this could significantly help to approach true real-time refreshment.

The main technical issue of the methodology is vulnerability menace for query integrity if data is temporarily stored in the staging area waiting for the appropriate partition (see step #4 in Fig. 8); for the pure ‘Trickle & Flip’ this will simply cause higher latency.

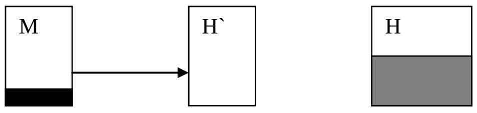
The main challenge of the approach is that of a very complicated multi-stage querying system. However the case is not “over-complex”, moreover, multi-stage querying is nothing new to have come with this approach: also the pure ‘Trickle & Flip’ needs such in order to combine static data with real-time data. Such querying tools and methodologies are already available. [8] describes a similar principle in generating reports on a data warehouse of multiple versions (i.e., stages, in terms of this paper).

5 Conclusion

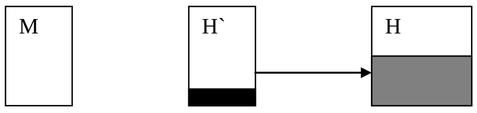
The data warehouse refreshing approach of ‘Multi-stage Trickle & Flip’ is proposed to mitigate the competition between the loading and querying processes when trying to approach real-time operation. It is designed as an extension to the well known principle of ‘Trickle & Flip’. Although the methodology doesn’t directly cover all the areas of ETL and querying, still it requires additional engineering efforts to obtain the maximum benefits, in particular, those of querying mechanism:

– Separate caching for each stage;
– A mechanism to balance needs for higher latency and the loading process.

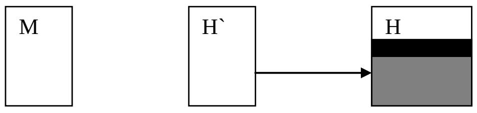
M is being fed from the source and has the freshest data of the last 5 minutes; normally, H\` is empty unless H is unavailable for some time due to querying. H contains the data of the last hour:



Copy M to H\`
Empty M



If H is available then copying is performed further to H
Now the real-time partition H has fresh data; normally, the latency is not greater than of 5 minutes:



If H is unavailable (due to querying activities), H\` is likely to contain more then one portion of M (while waiting for H)

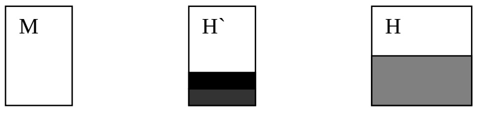


Fig. 8. An example. One iteration of a ‘Multi-stage Trickle & Flip’ loading; the first stage (according to the schema in Fig. 6 and the algorithm in Fig. 7). M and H\` are staging areas, H is the first real-time part of the DW. The same way, the loading is performed in the second stage.

Acknowledgements. This work has been supported by ESF project No. 2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044.

References

1. Jarke, M., Lenzerini, M., Vassiliou, Y.: Panos Vassiliadis. Fundamentals of Data Warehouses. Springer Verlag, 2nd, rev. and extended ed., XIV (2003)
2. Inmon, W. H., Terdeman, R. H., Norris-Montanari, J., and Meers, D.: Data Warehousing for E-Business, J. Wiley & Sons (2001)
3. Santos, R.J., Bernardino, J.: Real-time Warehouse Loading Methodology. Proceedings of the 2008 international symposium on Database engineering & applications (IDEAS '08), ACM, New York, NY, USA (2008)
4. Langseth, J.: Real-time data warehousing: Challenges and solutions, DSSResources.COM, http://dssresources.com/papers/features/langseth/langseth02082004.html (2004)
5. Jorg, T., Dessloch, S.: Near Real-time data warehousing using state-of-the-art ETL tools. Enabling Real-Time Business Intelligence, Lecture Notes in Business Information Processing, Volume 41. ISBN 978-3-642-14558-2. Springer-Verlag Heidelberg (2010)
6. Kimball, R., Caserta, J.: The data warehouse ETL toolkit: Practical techniques for extracting, cleaning, conforming, and delivering data. John Wiley & Sons (2004)
7. Kimball, R.; Ross, M.; Thornthwaite, W.; Mundy, J.; Becker, B.: The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence. John Wiley & Sons (2010)
8. Solodovnikova, D.: Building Queries on Multiple Versions of Data Warehouse. Proceedings of the Eighth International Baltic Conference (DB&IS 2008), Tallinn, Estonia, pp. 75-86 (2008)