



TenStep Supplemental Paper

4 February 2004

Learn When to use Live Transactional Data Versus a Data Warehouse

(The following contains a case study that is based on personal experience by the author, Tom Mochal.)

IT business applications need data. They create data and they use data. If they didn't need data, they would be pretty static and boring. Most organizations have more data than they can effectively manage; however, when you need something, it never seems to be all together or in the right format.

Over the past decade, the IT development organization has tried to bring this application data closer to the business users. This trend has allowed the users to have access to more timely information, and in many cases has removed the need to go through the IT organization for queries and reports. However, there are some fundamental architecture decisions that need to be made regarding these data stores and how they can be optimized for different purposes.

Transactional data must still be protected

Transactional data is the live data that the applications need to operate. For instance, the accounts receivable system is being updated all day by Accounts Receivable users. Inventory is being updated as orders are received and shipped. The sales system is updated as orders are received during the day. All of the data that these production applications need to run is referred to as transactional data.

In the push to bring data closer to the users, the IT organization must still be protective of this transactional data. Allowing business users broader access to the data cannot compromise the integrity of the information or negatively impact the production applications.

A case study

In a previous company where I worked, we had a desire to make all our business application data available to the business users. This included financial data, customer data, license revenue, education offerings, etc. We had reporting tools available for the end users so that they could write their own queries and reports. All of the data was currently available and used in the transactional systems, some which ran online during the day, and some which only ran at night. The question was how best to provide access to the users.

One alternative was to allow the users to access the data directly from the production applications. This was rejected for two reasons. First, the transactional data was stored in a manner that was optimized for the computer application. This was not the most intuitive way for business users to access the information. One of our applications, for instance, seemed from the outside to store data in two or three database tables. However, on closer inspection, the data was actually broken down into dozens and dozens of tables that were all connected by a set of primary and foreign keys. This database structure probably made



TenStep Supplemental Paper

perfect sense for the application. However, if we would have turned the users loose on these tables, there is no way they could have understood the data and the relationships in the tables. If they could have written reports that accessed the data, there is no telling whether they would have received the reporting results they expected.

The second reason that the transactional data was not used was because of contention issues. The business users were located all over the United States. We did not want these users trying to access the transactional data while the production systems were running live. Reporting can be very resource intensive, especially when accessing multiple databases. We felt that we were at risk of having the production processes slow down noticeably if users were cranking out reports from the same databases. This would not have been acceptable when people were utilizing the applications to get their jobs done.

Use a reporting datamart when instant access is not required

We quickly realized that we would not be able to make all this data available to the business users directly from the transactional systems. Instead, we looked at an approach using datamarts. Since most of the relevant data that the business people wanted was customer related, we proposed the creation of a Customer Datamart. We called this a datamart instead of a data warehouse since our data was not very resource intensive, and we did not utilize formal data warehouse techniques or tools.

Every night, we dumped the relevant transactional database tables into our Customer Datamart. The Customer Datamart was structured and optimized for reporting ease, and it was disengaged from the production systems. This allowed us to provide access to all of the data that our business people would be interested in reporting in a format that was much easier to understand from a human perspective.

There are tradeoffs to building a redundant data store

There were two tradeoffs to this approach. First, this approach requires you to create redundant data stores. In other words, the same data that was in the production transactional tables was also in the Customer Datamart. Why is redundancy a concern? With the size of the tables we were working with, it really has nothing to do with the extra disk space, since the cost of storage is so cheap. The concern comes in having to keep multiple data stores in sync. For instance, if our transactional system said that we had revenue of twenty million dollars last month, we needed the Customer Datamart to reflect that same amount. If you have duplicate data stores that produce different results for whatever reason, the support staff will spend a lot of time tracking down the discrepancies. The other concern is that if the structure of the transactional data changes, you have to change the datamart structure as well. This could end up causing twice the effort whenever a data structure changes.

We were able to overcome the first redundancy problem of keeping the data in synch by reloading a fresh copy of the transactional tables every night. In other words, we would load the current year's sales data today, and then load a completely new copy of the current year's sales data tomorrow and a completely new copy the following night. This is a much cleaner approach than having to worry about processing add/changes/deletes.



TenStep Supplemental Paper

However, our database tables were small enough that this was a practical solution. If your data stores are too large, this is not always an effective option.

The second tradeoff of building a separate Customer Datamart was one of timeliness. Obviously, if you update your reporting data store on a nightly basis, the data can be as much as one day old. This is a small tradeoff, and actually was more than acceptable in our organization. However, if you need the reporting data available in real time, this solution might not be right.

Some business users still need real time access

The last part of the solution involved allowing some users who actually needed real-time access to the live production data. For instance, our financial analysts needed real-time access during the monthly closeout process. They needed to provide management reports and make sure everything was properly closed out and balanced. Sometimes, this required multiple system updates during the day, as well as manual entries. They needed to be able to run a system update, and then query the data immediately. If the data was not correct, they might make updates and then come right back in and validate that everything worked. These people were given real-time access to the production data. However, this level of access was kept to a minimum and did not seem to have a negative performance impact on the production application.

Summary

The purpose of this case study was to describe why you might want to establish a separate data store for reporting purposes and end user queries. You would need to look at the benefits of keeping users off your production and transactional data versus the extra costs and risks associated with data redundancy. In the past, redundancy was viewed as a strict no-no. However, today it seems to make great sense to isolate data in a separate data store for reporting and analytical purposes, trading the cost of redundancy against the higher benefits associated with having the data directly accessible to the business users.