Considerations for Data Warehousing

Hardware and database considerations for data warehousing

When purchasing hardware infrastructure for a data warehouse and setting up the databases, what factors should we take into considerations? What RAID levels are suitable for stage, ODS and DDS databases? Do we need to put the fact tables on separate disks? How would the indexing strategy affect the disk layout? Is it necessary to setup a failover clustering? Is the memory requirement for data warehousing larger than OLTP systems? What are the optimum database settings for data warehouse databases? Are they different from transactional processing databases? For dimensional database, should we set recovery mode to full or simple?

It seems that a lot of DBAs have good understanding about the hardware considerations and database settings for transactional processing systems. But a few understand the difference between the requirements for transactional databases and data warehouse databases. “Aren’t they the same?” seems to be quite a common response. The purpose of this article is to describe the characteristic of data warehouse databases compared to transactional system databases and their impacts to the hardware considerations and database configurations.

A dimensional database for data warehouse can be of 2 forms: a relational format and multidimensional format. See this article [link] for further details. This relational format of dimensional database (from here on I will be referring to it as data warehouse database) has different characteristics to a transactional database in several ways:

1. Data warehouse databases are mostly static or read only. The only time we write into them is when we run the batch to update and populate the database, i.e. the database updates are normally performed in a batch. An exception to this is when we are running an active data warehouse [link], where the data warehouse is updated continuously in near real time fashion.

This is different to a transactional processing database where the proportion of updates, inserts and deletes are a lot higher. Probably almost every minute or even every few seconds the application performs transactions against the database to make changes to the data.

This difference affects the considerations for selecting the RAID level for the disk of the database server. It may also affect the choice of recovery model and redo log archive. Susan E. Hauser et all studied the RAID performance.

As Thomas Mercer-Hursh
once put it, for performance we use striping, for security we use mirroring, and for both we use both. Time and time again I admired how his statement simplified RAID understanding. Because in data warehouse databases (DW DBs) updates occur rarely (say daily) whilst in transactional processing databases (TP DBs) updates occur frequently (say every second), DW DBs needs performance more than TP DBs and TP DBs needs resilience more than DW DBs. In TP DBs we must ensure than the DBs are always available as updates are happening every second. In DW DBs, we must ensure that the batch finishes within time window and that large queries are completed within seconds, therefore stripping is essential. RAID 10 is the ideal RAID level for DW DBs, which provides both stripping and mirroring. If the budget is not permitting, the second best option is RAID 5, which also use stripping but no mirroring.

2. The fact tables in data warehouse databases are usually large tables. To understand the order of magnitude it is necessary to illustrate with numbers here. Transactional fact tables with 100 million rows are common and fact tables with 10 billion rows do happens. Snapshot fact tables (not the accumulative ones) could get into 100 billion rows in a few years.

Whilst we do create summary fact tables (a.k.a aggregates) to increase query performance, these huge base fact tables needs to be looked after properly. For example, we may consider putting them into separate physical files, or even a separate set of disks. We may want to put them on disks with highest RPM with different RAID level. And of course indexing strategy, table partitioning, etc. This will also affect the backup strategy, i.e. between full, differential and incremental/transaction logs.

The backup strategy will also be affected by whether we use Operational Data Store (ODS) to consolidate data

sources in a 3rd normal form database or not. If we do, then practically speaking we can recreate the dimensional data warehouse database from scratch. In this case we may want to do a full backup of ODS every day when the batch finishes and a full backup of the dimensional data warehouse database perhaps every Sunday. If we don't use ODS, then we better take a full backup of the dimensional data warehouse database every day.

3. Data warehouse databases are arranged in one of dimensional schema formats: star, snowflake, starflake and semi-star, with the former two being the most widely used whilst the latter is the least popular.

Transactional processing databases on the other hand are structured in one of normalised schema formats: third normal form, Boyd Codd form, 4th normal form or 5th normal form, with the former two being the most widely used whilst the latter two are the least popular.

The fact that data warehouse databases are in dimensional schema, that the dimensional tables use surrogate keys, that the queries contain a lot of joins to a central fact table, and that the fact tables have a composite primary keys, may affect the considerations for indexing strategy and maintenance (unique clustered/non-clustered, when to drop & rebuild indexes, etc), and for imposing referential integrity. The management of surrogate keys may affect some database engine configuration such as allowing table pinning into memory.

4. The size and the growth rate of a data warehouse database affect the disk layout, how flexible the SAN infrastructure should be, how much memory do we need and processor requirements. It also affect the requirements for backup infrastructure and restore procedures.

Capacity planning is bread and butter in DBA world, nothing new. Some database configurations are set based on capacity planning, for example: data and log file allocation, database growth size/percentage, etc. But the way we arrive to the conclusion that, for example, the size of the production dimensional database would probably be around 750 GB in 2 years time, requires understanding of data warehousing. The formula behind this capacity planning is the one that differentiate it from capacity planning for transaction processing database.

Simply speaking, most of the space requirement in data warehouse database comes from the fact tables. The fact tables are perhaps accountable for 80-90% of the disk space, the fact table indexes perhaps 5-10% and the dimensional tables perhaps 5-10% of the space. Then we need to factor-in the cubes (or the multidimensional databases), log files, staging and backup files. For an example of these calculations on how to estimated the required disk space for a data warehouse, please contact me.

5. Where would we stage the data from the source system: in the file system or in a database? It does depends on which ETL software we use. Some software like SSIS has a raw file format which is faster than dumping into database tables, some have its own internal staging storage, and with some ETL software it is more user friendly if we stage into a database, for example ability to do timestamping.

This affect the decision around where to put the stage and which RAID level to use for stage area, e.g. should we use RAID 0 to increase performance, or do we need resilience and use RAID 10, that kind of questions. We may also want to consider the network bandwidth between the source systems and the stage. Also the indexing and referential integrity if we stage into a database.

This is somewhat different to transactional processing system in which the concept of staging is not known in practice.

6. One important difference between data warehouse database and transactional system database is that data warehouse database is not real time. In OLTP systems, whilst some users read from database, some other users update the database; some of them targeting the same tables. If user A updates the order table at 10:23, user B expect to see the updates at 10:24. To support this kind of database usage, RDBMS provides several facilities:

- Locking and blocking mechanism, along with deadlock solving algorithm.
- Database transaction concurrency level, e.g. serializable, repeatable read, read committed and read uncommitted.
- Transactions processing, i.e. ability to commit or perform together several database transactions as a unit.
- Transaction log files, i.e. ability to rollback database transactions.

Data warehouse databases, on the other hand, are not real time. Well, that is, unless we talk about 'active data warehouse'. Updates in the source system are not expected to be in data warehouse immediately. User A updates the order table in OLTP at 10:23, and user B expect to see the updates in data warehouse the next day.

With this ‘non real time’ characteristic, we need to consider some related factors when choosing hardware. For example, do we need to provide a high degree of resilience and high database availability? Dual power supply? Disk RAID level? Clustering, replication, log shipping, hot standby? Database mirroring? Grid? Hot backup? Teamed NICs?

As most of the time during the day users perform select to database, we could probably do without the above OLTP facilities: locking, blocking, transaction level, log files, commit, etc. Some database engines such as Netezza and Teradata are more data warehousing oriented.

7. Large query output

In OLTP systems, the queries are relatively simple. A large degree of database transactions are selecting or updating or inserting or deleting just 1 or several records from a set of tables. For example, the process creating an order: it inserts a few records into order header and order detail tables. Viewing an order: select [columns] from order_header oh left join order_detail od on oh.order_id = od.order_id where order_id = [value]. Process of displaying orders placed recently: select [columns] from order_header where order_date < date_add(‘d’, getdate(), -5).

In data warehouse system, the SQL queries are a lot larger. When populating, truncate and repopulate the whole table is not an uncommon approach. Selecting several measures columns from a large fact table and many attributes from many dimension tables with many left joins are common. For example: select [fact measures, dimensions attributes] from fact_table left join dimension_table[1-N] on [fact key = dimension key] where dimension_attributes = [values]. And these queries can return a large number of records, i.e. large query output.

This impacts our considerations when selecting hardware for data warehouse. This large query output translates almost directly to the striping requirements of the disk system, as well as the amount of memory we need to make available to the database query engine.

8. Multidimensional database
On SQL Server 2005, the hardware considerations for Analysis Services are described in this page on Microsoft web site. Some tips for optimising SQL Server Analysis Services are posted on SQL Server Performance. On Oracle 10g, Oracle OLAP is described in this page, especially the first 2 links under