: What are physical database design tradeoffs for decision support data?

by Dan Power

Editor, DSSResources.com

Implementing a decision support database involves tradeoffs. A designer needs to transform an ideal, logical data model into a model that exploits effectively the capabilities and constraints of the actual hardware computing environment and a specific database management system. The overall goal is designing the database component of a data-driven DSS to minimize query times. Efficient use of storage space is much less a concern. Tradeoffs involve reasonably safe design changes, more aggressive compromises and technical optimization changes (cf., Haughey, 2006). There is no single best physical design. DSS analysts must ask probing questions and understand the design tradeoffs.

Harkins and Fuller (2002) in discussing moving from a logical to physical data model note we "must reexamine the design in the light of the capabilities offered by the target database." Choosing a database technology constrains physical design decisions and most DBMS have defaults for normal, anticipated query demands. A complete physical data model includes all the database artifacts required to create relationships between tables and achieve performance goals, such as indexes, constraint definitions, linking tables, partitioned tables or clusters (cf. Wikipedia).

The Applied Information Science website notes "A physical data model is a single logical model instantiated in a specific database management product (e.g., Sybase, Oracle, Informix, etc.) in a specific installation. The physical data model specifies implementation details which may be features of a particular product or version, as well as configuration choices for that database instance. These include index construction, alternate key declarations, modes of referential integrity (declarative or procedural), constraints, views, and physical storage objects such as tablespaces."

Ambler argues "Normalized data schemas, when put into production, often suffer from performance problems." That is the reality for decision support logical data models. A DSS analyst needs to understand the logical, ideal model and make it work in a production environment. To effectively evaluate the tradeoffs, a DSS analyst must ask questions like the following:

(c) 2022 Daniel J. Power, Power Enterprises <power@dssresources.com>
URL: http://dssresources.com/faq/index.php?action=artikel&cat=&id=176&artlang=en

: What are physical database design tradeoffs for decision support data?

1) What are user expectations for response time?

2) What are the security requirements for the data and the system?

3) What are the hardware platform limitations?

4) Will an existing database management system be used, if so what are its limitations?

5) What is the anticipated data volume? How much data will be stored in approximately how many tables?

6) How often will the decision support data store be accessed?

7) Will queries primarily be preplanned or ad hoc?

8) How many concurrent users are anticipated?

9) Will some data be accessed more often than other data?

10) In the logical design, are there many small lookup tables or other tables that may be extensively involved in joins?

11) Will add, delete and update queries be possible for users?

(c) 2022 Daniel J. Power, Power Enterprises <power@dssresources.com>

12) Do natural partitions exist for the data like stores, regions or fiscal quarters that might impact design decisions?

Tradeoffs primarily involve choices about normalization and table construction, partitioning, indexing, and storage redundancy.

So what are reasonably safe design changes and compromises for DSS data storage? If the logical model is normalized and users can NOT add, delete and update data, then splitting a table, combining two tables, adding a calculated field or creating a primary key index is usually "safe" and an easy tradeoff to improve query performance.

More aggressive compromises to the logical model need to be evaluated more carefully to avoid unanticipated consequences. For example, adding redundant data is a more aggressive strategy that can backfire. If the data comes from different source systems and is not "really" redundant, results from queries will vary. Aggressive trade-offs can reduce data quality and compromise the integrity of the DSS. Also, some denormalization can reduce the understandability of the physical data model and lead to query design errors. Data integrity is very important and should not be compromized in trade-offs.

Deciding to implement technical optimizations like a hash or join index seems easy, but may be more difficult in some DBMS products that others. Also, many indexing approaches exist and have advantages and disadvantages. The wrong type of indexing can create problem. So unless the data quantity is very large, technical optimization may NOT be needed to provide acceptable query performance. Scalability is a serious problem with enterprise-wide, data-driven DSS, but the DBMS and the architecture impact scalability more than indexes and DBMS "tuning". Remember database "tuning" may lead to poorer performance rather than improving query response.

Selecting appropriate indexes and database architectures is very important for a decision support data store. Choices should be made following planned data gathering and thoughtful analysis and evaluation. Creating a scenario of how the system will be used can assist in estimating loads and data storage needs.

(c) 2022 Daniel J. Power, Power Enterprises <power@dssresources.com>
URL: http://dssresources.com/faq/index.php?action=artikel&cat=&id=176&artlang=en

: What are physical database design tradeoffs for decision support data?

In general, if DSS data is physically stored in tables where some relations have been denormalized, it can be more efficient for some anticipated queries (cf., Power, 2002). If the data store is an historical archive,
the denormalization is not likely to lead to anomolies, but it may be harder to
write appropriate queries or create drop downs for queries or support ad hoc queries. Also, creating a dynamic or materialized view may be a better long term alternative than denormalizing the base tables and may gain the same query performance improvements. Denormalization and views involve maintenance and performance tradeoffs.

In many cases, it is appropriate to horizonally partition decision support data. Partitioning involves placing data rows in different files based on categories like date created,
store where transaction occurred or region. Hoffer et al (2008) note "partitioned files can be placed on seperate disk drive to reduce contention for the same
drive and hence improve query performance ..." Also, in some cases vertical partitioning or a combination of the two may improve performance given the anticipated query load.

Indexing is a very important structuring tool for decision support data. An index is a table or other data structure used to determine the location of rows in a file that satisfy some condition. In general,
a decision support database is changed in clumps with batch updates every
day or week. Complex queries are then run against the historical database which may be very large.

For decision support data, in some situations a hierarchical index speeds queries. In other situations an "extended join index" can improve query performance. If the join
is used frequently in queries, then the index is used repeatedly. According to Burleson, "Oracle benchmarks claim that bitmap join indexes can run a query more than
eight times faster than traditional indexing methods." Hoffer et al (2009) note
a hash index table is found "in some data warehousing database technologies that use parallel processing (p. 281)".

Indexes are most useful when associated with very large tables
that have many values for attributes. Also, indexing fields that appear in the WHERE
clause of queries can be very useful. Some indexes are automatically created. For example, Oracle creates an index for each
UNIQUE or PRIMARY KEY declaration. Designers should be careful when indexing attributes that have NULL values. Indexes are primarily used to enhance database performance,
but inappropriate use will result in slower performance.

(c) 2022 Daniel J. Power, Power Enterprises <power@dssresources.com>

URL: http://dssresources.com/faq/index.php?action=artikel&cat=&id=176&artlang=en

: What are physical database design tradeoffs for decision support data?

In terms of hardware and data storage, RAID-5 or Rotating Parity Array storage is generally best for read-intensive applications accessing very large data sets spread over multiple disk drives. In general, larger block sizes are used for decision support databases (cf., Hoffer et al, 2009).

In conclusion, we all need to improve our data modeling skills and especially skill in translating normalized logical data models into effective physical models for a particular hardware platform and database product. How do we improve our data modeling skills? According to Ambler, "practice, practice, practice." I would add GET A MENTOR. Failure in data modeling is not acceptable on large scale, data-driven DSS projects.

References

Ambler, S. W., "Data Modeling 101," AgileData.org, URL http://www.agiledata.org/essays/dataModeling101.html .

Applied Information Science website, URL http://www.aisintl.com/case/CDM-PDM.html

Ben-Gan, I., "Ensure Data Integrity with Cascading DRI," SQL Server Magazine, June 1, 2002, InstantDoc #25120, URL http://www.sqlmag.com/Article/ArticleID/25120/sql_server_25120.html .

Burleson, D., "How to use Oracle9i bitmap join indexes," November 12, 2002, Burleson Consulting, URL http://www.dba-oracle.com/art_builder_bitmap_join_idx.htm

Cohen, R., "Seven Principles for Enterprise Data Warehouse Design," DM Review Online, January 12, 2006, URL http://www.dmreview.com/news/1045818-1.html .

Date, C.J., Database in Depth: Relational Theory for Practitioners, O'Reilly, 2005.

(c) 2022 Daniel J. Power, Power Enterprises <power@dssresources.com>

URL: http://dssresources.com/faq/index.php?action=artikel&cat=&id=176&artlang=en

: What are physical database design tradeoffs for decision support data?

Harkins, S. and A. Fuller, "The transition from logical to physical data model," TechRepublic.com, August 21, 2002, URL http://articles.techrepublic.com.com/5100-10878_11-1050841.html

Haughey, T., "Transforming a Logical Data Model to a Physical Database Design - an Overview," TDAN.com, February 1, 2006, URL http://www.tdan.com/view-special-features/5389 .

Hoffer, J. A., M.B. Prescott and H. Topi, Modern Database Management (9th edition), Upper Saddle River, NJ: Pearson/Prentice Hall, 2009.

Power, D., Is a Data Warehouse a DSS? What is a star schema? How does a snowflake schema differ from a star schema? DSS News, Vol. 3, No. 4, February 17, 2002.

Power, D., "Is parallel database technology needed for data-driven DSS?" DSS News, Vol. 7, No. 8, April 9, 2006.

w3schools, "SQL Tutorial," URL http://www.w3schools.com/sql .

Wikipedia, "Index (database)," URL http://en.wikipedia.org/wiki/Index_(database)

Wikipedia, "Physical data model," URL http://en.wikipedia.org/wiki/Physical_data_model

Author: Daniel Power
Last update: 2008-10-01 10:36

(c) 2022 Daniel J. Power, Power Enterprises <power@dssresources.com>
URL: http://dssresources.com/faq/index.php?action=artikel&cat=&id=176&artlang=en