Star-Schema Design [closed]

Asked 16 years, 3 months ago Modified 1 year, 4 months ago

Viewed 37k times

63

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 8 years ago.

Improve this question

Is a Star-Schema design essential to a data warehouse? Or can you do data warehousing with another design pattern?

database design-patterns data-warehouse star-schema dimensional-modeling

Share

Improve this question

Follow

edited Nov 18, 2011 at 14:48

MOLAP

804 ● 4 ● 14 ● 28

asked Sep 21, 2008 at 2:22

S.Lott
391k • 82 • 517 • 788

You could technically put everything in one table, ie a fact table with no dimension tables, and the actual dimension data instead of keys. But this would get very large very quickly, hence the single level of normalization.
– Neil McGuigan Feb 21, 2013 at 20:13

8 Answers

Sorted by: Highest score (default)

97

Using star schemas for a data warehouse system gets you several benefits and in most cases it is appropriate to use them for the top layer. You may also have an operational data store (ODS) - a normalised structure that holds 'current state' and facilitates operations such as data conformation. However there are reasonable situations where this is not desirable. I've had occasion to build systems with and without ODS layers, and had specific reasons for the choice of architecture in each case.

Without going into the subtlties of data warehouse architecture or starting a Kimball vs. Inmon flame war the main benefits of a star schema are:

- Most database management systems have facilities in the query optimiser to do 'Star Transformations' that use Bitmap Index structures or Index Intersection for fast predicate resolution. This means that selection from a star schema can be done

without hitting the fact table (which is usually much bigger than the indexes) until the selection is resolved.

- Partitioning a star schema is relatively straightforward as only the fact table needs to be partitioned (unless you have some biblically large dimensions). Partition elimination means that the query optimiser can ignore patitions that could not possibly participate in the query results, which saves on I/O.
- Slowly changing dimensions are much easier to implement on a star schema than a snowflake.
- The schema is easier to understand and tends to involve less joins than a snowflake or E-R schema. Your reporting team will love you for this
- Star schemas are much easier to use and (more importantly) make perform well with ad-hoc query tools such as Business Objects or Report Builder. As a developer you have very little control over the SQL generated by these tools so you need to give the query optimiser as much help as possible. Star schemas give the query optimiser relatively little opportunity to get it wrong.

Typically your reporting layer would use star schemas unless you have a specific reason not to. If you have multiple source systems you may want to implement an Operational Data Store with a normalised or snowflake schema to accumulate the data. This is easier because

an ODS typically does not do history. Historical state is tracked in star schemas where this is much easier to do than with normalised structures. A normalised or snowflaked Operational Data Store reflects 'current' state and does not hold a historical view over and above any that is inherent in the data.

ODS load processes are concerned with data scrubbing and conforming, which is easier to do with a normalised structure. Once you have clean data in an ODS, dimension and fact loads can track history (changes over time) with generic or relatively simple mechanisms relatively simply; this is much easier to do with a star schema, Many ETL tools (for example) provide built-in facilities for slowly changing dimensions and implementing a generic mechanism is relatively straightforward.

Layering the system in this way providies a separation of responsibilities - business and data cleansing logic is dealt with in the ODS and the star schema loads deal with historical state.

edited Mar 9, 2017 at 16:19

community wiki
11 revs, 2 users 99%
ConcernedOfTunbridgeWells

9

There is an ongoing debate in the datawarehousing litterature about ***where*** in the datawarehouse-architecture the Star-Schema design should be applied.

In short Kimball advocates very highly for using only the Star-Schema design in the datawarehouse, while Inmon first wants to build an Enterprise Datawarehouse using normalized 3NF design and later use the Star-Schema design in the datamarts.

In addition here to you could also say that Snowflake schema design is another approach.

A fourth design could be the Data Vault Modeling approach.

Share Improve this answer

Follow

answered Nov 18, 2011 at 14:19

MOLAP

804 ● 4 ● 14 ● 28

8

Star schemas are used to enable high speed access to large volumes of data. The high performance is enabled by reducing the amount of joins needed to satsify any query that may be made against the subject area. This is done by allowing data redundancy in dimension tables.

You have to remember that the star schema is a pattern for the top layer for the warehouse. All models also involve staging schemas at the bottom of the warehouse stack, and some also include a persistant transformed

merged staging area where all source systems are merged into a 3NF modelled schema. The various subject areas sit above this.

Alternatives to star schemas at the top level include a variation, which is a snowflake schema. A new method that may bear out some investigation as well is Data Vault Modelling proposed by Dan Linstedt.

Share Improve this answer Follow

edited Aug 17, 2023 at 15:36

community wiki
2 revs, 2 users 93%
Mike McAllister

7

The thing about star schemas is they are a natural model for the kinds of things most people want to do with a data warehouse. For instance it is easy to produce reports with different levels of granularity (month or day or year for example). It is also efficient to insert typical business data into a star schema, again a common and important feature of a data warehouse.

You certainly can use any kind of database you want but unless you know your business domain very well it is likely that your reports will not run as efficiently as they could if you had used a star schema.

answered Sep 21, 2008 at 2:25

Share Improve this answer

Follow

Mike
414 ● 3 ● 9

It is basically object-oriented design in SQL ;)
– Hamish Grubijan Aug 7, 2010 at 16:14

6

Star schemas are a natural fit for the last layer of a data warehouse. How you get there is another question. As far as I know, there are two big camps, those of Bill Inmon and Ralph Kimball. You might want to look at the theories of these two guys if/when you decide to go with a star.

Also, some reporting tools really like the star schema setup. If you are locked into a specific reporting tool, that might drive what the reporting mart looks like in your warehouse.

Share Improve this answer

Follow

answered Sep 21, 2008 at 2:54

Josh McAdams
605 ● 4 ● 5

2 +1 - Kimball vs. Inmon is one of the great religious wars. IMHO the presence of a religious divide of this sort is a clear indicator that neither argument is compelling. I've built systems with and without ODS layers - and had good reasons for the architectural decisions.
– ConcernedOfTunbridgeWells Nov 12, 2008 at 22:19

1 There is also Data Vault Modelling (en.wikipedia.org/wiki/Data_Vault_Modeling) now as a layer below your data marts. – Marcus D Jun 3, 2011 at 15:58

4

Star schema is a logical data model for relational databases that fits the regular data warehousing needs; if the relational environment is given, a star or a snowflake schema will be a good design pattern, hard-wired in lots of DW design methodologies.

There are however other than relational database engines too, and they can be used for efficient data warehousing. Multidimensional storage engines might be very fast for OLAP tasks (TM1 eg.); we can not apply star schema design in this case. Other examples requiring special logical models include XML databases or column-oriented databases (eg. the experimental C-store)).

Share Improve this answer Follow

edited Jan 25, 2012 at 12:51

answered Jun 9, 2009 at 16:15

csaba
680 ● 5 ● 7

"other than relational database engines"... Interesting. What design pattern do they use for the data? A star schema or some other kind of design? – S.Lott Jun 9, 2009 at 17:14

Multidimensional (MOLAP) databases store their data in various multidimensional array structures. Conceptually, in my interpretation, when building a data warehouse we build a conceptual data model first (with dimensions and data cubes), then we map it to the logical level (tables and constraints), which is then implemented on phisical level (files on disks, handled by the DBMS). MOLAP engines however

map the conceptual model directly to phisical level. As star schema is the logical model of relational dws, therefore it is omitted in a MOLAP environment. – csaba Jun 10, 2009 at 8:27

3

It's possible to do without. However, you will make life hard for yourself -- your organization will want to use standard tools that live on top of DWs, and those tools will expect a star schema -- a lot of effort will be spent fitting a square peg in a round hole.

A lot of database-level optimizations assume that you have a star schema; you will spend a lot of time optimizing and restructuring to get the DB to do "the right thing" with your not-quite-star layout.

Make sure that the pros outweigh the cons..

(Does it sound like I've been there before?)

-D

Share Improve this answer

Follow

answered Sep 22, 2008 at 3:04

SquareCog

19.6k ● 8 ● 51 ● 63

2

There are three problems we need to solve.

1) How to get the data out of the operational source system without putting undue pressure on them by joining

tables within and between them, cleaning data as we extract, creating derivations etc.

2) How to merge data from disparate sources - some legacy, some file based, from different departments into an integral, accurate, efficiently stored whole that models the business, and does not reflect the structures of the source systems. Remember, systems change / are replaced relatively quickly, but the basic model of the business changes slowly.

3) How to structure the data to meet specific analytical and reporting requirements for particular people/departments in the business as quickly and accurately as possible.

The solution to these three very different problems require different architectural layers to solve them

Staging Layer We replicate the structures of the sources, but only changed data from the sources are loaded each night. once the data is taken from the staging layer into the next layer, the data is dropped. Queries are single table queries with a simple data_time filter. Very little effect on the source.

Enterprise Layer This is a business oriented 3rd normal form database. Data is extracted (and afterward dropped) from the staging layer into the enterprise layer, where it is cleaned, integrated and normalised.

Presentation (Star Schema) Layer Here, we model dimensionally to meet specific requirements. Data is deliberately de-normalise to reduce the number of joins. Hierarchies that may occupy several tables in the Enterprise Layer are collapsed into a single dimension tables, and multiple transactional tables may be merged into single fact tables.

You always face these three problems. If you choose to do away with the enterprise layer, you still have to solve the second problem, but you have to do it in the star schema layer, and in my view, this is the wrong place to do it.

answered Aug 31, 2012 at 11:11

Steve

245 ● 1 ● 6