DOWNLOAD PDF STAR SCHEMA DIMENSIONAL MODELING

Chapter 1 : Dimensional Modeling Schema

In computing, the star schema is the simplest style of data mart schema and is the approach most widely used to develop data warehouses and dimensional data marts. The star schema consists of one or more fact tables referencing any number of dimension tables.

Introduction to star schema Star schema is a dimensional design for a relational database often used in a data warehouse system. The star schema has a fact table at the center, and the fact table is surrounded by a number of dimension tables. The star schema name comes from the appearance of the schema, which looks like a star. In the star schema, related dimensions grouped together as columns in dimension tables and used to store the context of the facts stored in the fact table. Star schema example The following is a star schema based on dimensions and facts for the sale process. Dimension tables A dimension table consists of columns that represent dimensions which provide the context needed for studying the facts. A dimension table typically stores characters that describe facts. A dimension table often has many columns, each for an attribute of interest. The dimension table is not necessary for the third normal form 3NF. The primary key of a dimension table is a single surrogate key that is a part of the composite primary key of the fact table. It is always a surrogate key, which data warehouse system generate and manage. For more information on dimension table, check it out dimension table article. Normally facts are numbers that can be aggregated, summarized or rolled up. The fact table contains surrogate keys as a part of its primary key. Those keys are the foreign key of the dimension tables. For more information on the fact table, please refer to the fact table article. Star schema notes Star schema can help business analysts to answer questions that might not have been asked during the design process by looking through different dimensions. Star schema often stores data at a great level of detail however it can be rolled up at various levels of detail based on aggregations. The capability to study facts depends on the level of detail that the fact table stores. The more dimension tables that star schema has, the more reporting possibilities it provides. We also gave you some helpful notes about the star schema.

Chapter 2 : Dimensional Modeling and Kimball Data Marts in the Age of Big Data and Hadoop - Sonra

Dimensional models focus on process measurement events, dividing data into either measurements or the "who, what, where, when, why, and how" descriptive context. Dimensional models can be instantiated in both relational databases, referred to as star schemas, or multidimensional databases, known as online analytical processing (OLAP) cubes.

Model[edit] The star schema separates business process data into facts, which hold the measurable, quantitative data about a business, and dimensions which are descriptive attributes related to fact data. Examples of fact data include sales price, sale quantity, and time, distance, speed and weight measurements. Related dimension attribute examples include product models, product colors, product sizes, geographic locations, and salesperson names. A star schema that has many dimensions is sometimes called a centipede schema. Fact tables[edit] Fact tables record measurements or metrics for a specific event. Fact tables generally consist of numeric values, and foreign keys to dimensional data where descriptive information is kept. This can result in the accumulation of a large number of records in a fact table over time. Fact tables are defined as one of three types: Transaction fact tables record facts about a specific event e. This key is a simple primary key. Dimension tables[edit] Dimension tables usually have a relatively small number of records compared to fact tables, but each record may have a very large number of attributes to describe the fact data. Dimensions can define a wide variety of characteristics, but some of the most common attributes defined by dimension tables include: Time dimension tables describe time at the lowest level of time granularity for which events are recorded in the star schema Geography dimension tables describe location data, such as country, state, or city Product dimension tables describe products Employee dimension tables describe employees, such as sales people Range dimension tables describe ranges of time, dollar values or other measurable quantities to simplify reporting Dimension tables are generally assigned a surrogate primary key , usually a single-column integer data type, mapped to the combination of dimension attributes that form the natural key. Benefits[edit] Star schemas are denormalized , meaning the normal rules of normalization applied to transactional relational databases are relaxed during star schema design and implementation. The benefits of star schema denormalization are: Simpler queries - star schema join logic is generally simpler than the join logic required to retrieve data from a highly normalized transactional schema. Simplified business reporting logic - when compared to highly normalized schemas, the star schema simplifies common business reporting logic, such as period-over-period and as-of reporting. Query performance gains - star schemas can provide performance enhancements for read-only reporting applications when compared to highly normalized schemas. Fast aggregations - the simpler queries against a star schema can result in improved performance for aggregation operations. Disadvantages[edit] The main disadvantage of the star schema is that data integrity is not enforced well since it is in a highly de-normalized state. One-off inserts and updates can result in data anomalies which normalized schemas are designed to avoid. Generally speaking, star schemas are loaded in a highly controlled fashion via batch processing or near-real time "trickle feeds", to compensate for the lack of protection afforded by normalization. Star schema is also not as flexible in terms of analytical needs as a normalized data model. Star schemas tend to be more purpose-built for a particular view of the data, thus not really allowing more complex analytics. Typically these relationships are simplified in star schema to conform to the simple dimensional model. Example[edit] Star schema used by example query. Consider a database of sales, perhaps from a store chain, classified by date, store and product. The image of the schema to the right is a star schema version of the sample schema provided in the snowflake schema article. For example, the following query answers how many TV sets have been sold, for each brand and country, in

Chapter 3 : Data Warehouse Modeling: Star Schema vs. Snowflake Schema

Star schema is probably most popular schema in dimensional modeling because of its simplicity and flexibility. In a Star schema design, any information can be obtained just by traversing a single join, which means this type of schema will be ideal for information retrieval (faster query processing).

Identify the fact Choose the business process The process of dimensional modeling builds on a 4-step design method that helps to ensure the usability of the dimensional model and the use of the data warehouse. The basics in the design build on the actual business process which the data warehouse should cover. Therefore, the first step in the model is to describe the business process which the model builds on. This could for instance be a sales situation in a retail store. Declare the grain After describing the business process, the next step in the design is to declare the grain of the model. The grain of the model is the exact description of what the dimensional model should be focusing on. To clarify what the grain means, you should pick the central process and describe it with one sentence. Furthermore, the grain sentence is what you are going to build your dimensions and fact table from. You might find it necessary to go back to this step to alter the grain due to new information gained on what your model is supposed to be able to deliver. Identify the dimensions The third step in the design process is to define the dimensions of the model. The dimensions must be defined within the grain from the second step of the 4-step process. Dimensions are the foundation of the fact table, and is where the data for the fact table is collected. Typically dimensions are nouns like date, store, inventory etc. These dimensions are where all the data is stored. For example, the date dimension could contain data such as year, month and weekday. Identify the facts After defining the dimensions, the next step in the process is to make keys for the fact table. This step is to identify the numeric facts that will populate each fact table row. This step is closely related to the business users of the system, since this is where they get access to data stored in the data warehouse. Therefore, most of the fact table rows are numerical, additive figures such as quantity or cost per unit, etc. The neutrality of this section is disputed. Relevant discussion may be found on the talk page. Please do not remove this message until conditions to do so are met. June Learn how and when to remove this template message Dimensional normalization or snowflaking removes redundant attributes, which are known in the normal flatten de-normalized dimensions. Dimensions are strictly joined together in sub dimensions. Snowflaking has an influence on the data structure that differs from many philosophies of data warehouses. If you are only going to do operational reports then you may be able to get by with 3NF because your operational user will be looking for very fine grain data. There are some arguments on why normalization can be useful. For example, a geographic dimension may be reusable because both the customer and supplier dimensions use it. Benefits of dimensional modeling[edit] This section may rely excessively on sources too closely associated with the subject , potentially preventing the article from being verifiable and neutral. Please help improve it by replacing them with more appropriate citations to reliable, independent, third-party sources. June Learn how and when to remove this template message Benefits of the dimensional model are the following: Compared to the normalized model, the dimensional model is easier to understand and more intuitive. In dimensional models, information is grouped into coherent business categories or dimensions, making it easier to read and interpret. Simplicity also allows software to navigate databases efficiently. In normalized models, data is divided into many discrete entities and even a simple business process might result in dozens of tables joined together in a complex way. Dimensional models are more denormalized and optimized for data querying, while normalized models seek to eliminate data redundancies and are optimized for transaction loading and updating. The predictable framework of a dimensional model allows the database to make strong assumptions about the data which may have a positive impact on performance. Each dimension is an equivalent entry point into the fact table, and this symmetrical structure allows effective handling of complex queries. Query optimization for star-joined databases is simple, predictable, and controllable. Dimensional models are scalable and easily accommodate unexpected new data. Existing tables can be changed in place either by simply adding new data rows into the table or executing SQL alter table commands. No queries or applications that sit on top of the data warehouse need to be

reprogrammed to accommodate changes. Old queries and applications continue to run without yielding different results. But in normalized models each modification should be considered carefully, because of the complex dependencies between database tables. June Learn how and when to remove this template message We still get the benefits of dimensional models on Hadoop and similar big data frameworks. However, some features of Hadoop require us to slightly adapt the standard approach to dimensional modelling. We can only add but not update data. As a result we can only append records to dimension tables. Slowly Changing Dimensions on Hadoop become the default behavior. In order to get the latest and most up to date record in a dimension table we have three options. First, we can create a View that retrieves the latest record using windowing functions. Second, we can have a compaction service running in the background that recreates the latest state. Third, we can store our dimension tables in mutable storage, e. HBase and federate queries across the two types of storage. The way data is distributed across HDFS makes it expensive to join data. In a distributed relational database MPP we can co-locate records with the same primary and foreign keys on the same node in a cluster. This makes it relatively cheap to join very large tables. No data needs to travel across the network to perform the join. On HDFS tables are split into big chunks and distributed across the nodes on our cluster. As a result joins on Hadoop for two very large tables are quite expensive as data has to travel across the network. We should avoid joins where possible. For a large fact and dimension table we can de-normalize the dimension table directly into the fact table. For two very large transaction tables we can nest the records of the child table inside the parent table and flatten out the data at run time. Literature[edit] Kimball, Ralph ; Margy Ross The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling 3rd ed. Margy Ross Kimball Group Kimball Group, Design Tips Archived from the original on

Chapter 4 : Understanding Star Schemas

The star schema is the simplest model used in DWH. Because the fact table is in the center of the schema with dimension tables around it, it looks roughly like a star. This is especially apparent when the fact table is surrounded by five dimension tables.

Next, remove the remaining partial dependencies. Figure 2 illustrates a data structure in 3NF. The normalized relations are beneficial for transaction processing. On the other hand, the normalized data may cause significant inefficiencies in the analytical processing, which infrequently updates data, and always retrieves many rows. Dimensional Modeling A dimensional model is a data model structured to deliver maximum query performance and ease of use. A typical dimensional model consists of a fact table surrounding by a set of dimension tables. This data structure is often called a star schema. Dimensional models have proved to be understandable, predictable, extendable, and highly responsive to ad hoc demands because of their predictable symmetric nature [2]. The term fact refers to performance measurements from business processes or events. The invoice, shown in Figure 1, represents a sales activity, the quantity of the product and the extended amount are the measurements of this activity. Even if a customer changed an order, a separate business transaction took place, and corresponding measurements were recorded in other fact tables. Thus, data in the fact table is nonvolatile. DBA may update a fact table due to some technical problems. Designers should not consider updating fact tables at design time. Dimension tables provide context to describe the activity, for example, what was involved in the activity, when, where and why it happened, who perform the activity and how to make it happen. When creating a dimensional model, we should eliminate anything that does not have any business analysis values. For example, the customer phone number is not necessary in the customer dimension. Kimball introduced a four-step dimensional design process for designing dimensional data models [1][2]. When an entity relationship model is available, we can obtain dimensional models through a four-step transformation process [8]. We use the data captured in the Figure 1 to practice these design processes. Four-Step Dimensional Design Process Step 1 - Select the Business Process A business process is a low-level activity performed by an organization and frequently expressed as action verb [1]. The measurements from each business process are usually represented in a fact table and sometimes several related fact tables [2]. All facts in a fact table should correspond to the same key, in other words, all facts in the fact table should have same granularity and dimensionality. If a fact links to a different key from others, the fact may be from different processes, therefore it should go to other fact table. The invoice was generated in the business process, selling products. The data collected in the invoice enables business users to analyze sales revenue. Step 2 - Declare the Grain The grain exactly specifies the level of detail with the fact table measurements [1]. Step 3 - Identify Dimensions The grain statement implied the primary dimensionality of the fact tables More dimensions can be added to the fact table if these dimensions naturally take on only one value under each combination of the primary dimensions [6]. Here is a list of dimensions to describe the facts: A table in 1NF has partial dependencies, which indicates different entities are mixed into the same dimension table, as demonstrated in Table 2. This introduces the insert anomaly. For example, we cannot add new products to the table if the products have not been sold. Typical facts are numeric additive figures. Here is the list of measurements identified in the selected business process. Figure 3 - Dimensional Model Diagram Four-Step Transformation Process Several techniques of data warehouse construction from transactional data models have been briefly discussed in [9]. We adopt a four-step process introduced in [8] to convert the 3NF diagram in Figure 2 to a star schema. Step 1 - Classify entities In this step, we classify all entities in Figure 2 to three categories: CITY Step 2 - Designing high-level star schema This step is to designate transaction entities as fact tables and component entities as dimension tables. However, the transaction entity to fact table mapping is not always one-to-one and the correspondence between component entities and dimensions is not always one to one as well. A star schema, shown in Figure 4, can easily be identified from the Figure 2. For complicate diagram, [8] presented a solution with three steps: Figure 4 - High-level Star Schema for Selling Product Transaction Step 3 - Designing detailed fact table A fact table usually contains a super-key, which consists of

degenerate dimensions and all foreign keys linked to dimension tables. The nonprime attributes in the transaction entity can be defined as facts if they have analysis value and they are additive. Figure 5 demonstrates the fact table produced in this step. The single-part key is a surrogate key generated to ensure it remains unique over time. All the dimension tables are presented in Figure 6. Figure 6 - Dimension Tables for Selling Product Transaction The star schema obtained from this four-step design process is the same as the one shown in Figure 3. An experienced designer can make a trade-off. The example we used may derive different star schemas. In a summary, ease-of-use and query performance are two primary reasons that dimensional modeling is the widely accepted best practice for data warehousing tools [2]. To achieve the goals, we should resist the temptation to normalize dimension tables. Reference [1] Kimball R. Dimensional modeling, aiming to analysis processing, is quite different from 3NF modeling. Each of them has own systematic modeling processes and techniques. Check out these related tips:

Chapter 5 : Star Schema Foundations | Pluralsight

The most obvious characteristic of the star schema is that dimension tables are not normalized. In the model above, the pink fact_sales table stores aggregated data created from our operational database(s).

Requirement of different design schema In Dimensional modeling, we can create different schema to suit our requirements. We need various schema to accomplish several things like accommodating hierarchies of a dimension or maintaining change histories of information etc. In this article we will discuss about 3 different schema, namely - Star, Snowflake and Conformed and we will also discuss how hierarchical information are modeled in these schemata. We will reserve the discussion on maintaining change histories for our next article. Storing hierarchical information in dimension tables From our previous article, we already know what is a dimension. Simply put, a dimension is something that qualifies a measure number. But if I say, "McDonalds sell burgers per month" - then that would make perfect sense. Here, "burger" and "month" are the members of dimensions and they are qualifying the number in this sentence. It is important to notice that "burger" and "month" are not dimension themselves - they are just the members of the dimensions "food" and "time" respectively. Typically a dimension will have several members and those members will be stored in separate rows in the dimension table. So the "food" dimension table of McDonalds will have one row for burger, one row for fries, one row for "drinks" etc. Similarly, "time" dimension may contain 12 different months as the members of that dimension. Often we may find that there are hierarchical relations among the members of a dimension. That is certain members of the dimension can be grouped under one group whereas other members can be grouped into a separate group. Consider this - french fries and twister fries both are "fries" and hence can be grouped under the same group "fries". Similarly chicken burger and fish burger both can be grouped as "burger". French Fries Twister Fries This type of hierarchical relations can be stored in the model by following two different approaches. We can either store them in the same "food" dimension table star schema approach or we can create a separate dimension table in addition to "food" dimension - just to store the type of the foods snowflake schema approach. In a star schema all the dimension tables are connected only with the fact table and no dimension table is connected with any other dimension table. Benefit of Star Schema Design Star schema is probably most popular schema in dimensional modeling because of its simplicity and flexibility. In a Star schema design, any information can be obtained just by traversing a single join, which means this type of schema will be ideal for information retrieval faster query processing. Here, note that all the hierarchies or levels of the members of a dimension are stored in the single dimension table - that means, lets say if you wish to group veggie burger and chicken burger in "burger" category and french fries and twister fries in "fries" category, you have to store that category information in the same dimension table. Star schema provides a de-normalized design Storing Hierarchy in star schema As depicted above, we will store hierarchical information in a flattened pattern in the single dimension table in star schema. So our food dimension table will look like this:

Chapter 6 : Relational vs. Dimensional Databases, what's the difference? - Stack Overflow

Star Schema is a relational database schema for representing multidimensional data. It is the simplest form of data warehouse schema that contains one or more dimensions and fact tables. It is called a star schema because the entity-relationship diagram between dimensions and fact tables resembles a star where one fact table is connected to.

Snowflake Schema by Database designer and developer, financial analyst Posted: April 28, In the previous two articles, we considered the two most common data warehouse models: The star schema and the snowflake schema are ways to organize data marts or entire data warehouses using relational databases. Both of them use dimension tables to describe data aggregated in a fact table. Everyone sells something, be it knowledge, a product, or a service. Storing this information, either in an operational system or in a reporting system, is also a need. So we can expect to find some type of sales model inside the data warehouse of nearly every company. The Star Schema The most obvious characteristic of the star schema is that dimension tables are not normalized. The light blue tables are dimension tables. We decided to use these five dimensions because we need to create reports using them as parameters. The granulation inside each dimension is also determined by our reporting needs. The Snowflake Schema This snowflake schema stores exactly the same data as the star schema. The fact table has the same dimensions as it does in the star schema example. The most important difference is that the dimension tables in the snowflake schema are normalized. Interestingly, the process of normalizing dimension tables is called snowflaking. Once again, visually the snowflake schema reminds us of its namesake, with several layers of dimension tables creating an irregular snowflake-like shape. Normalization As mentioned, normalization is a key difference between star and snowflake schemas. Regarding this, there are a couple of things to know: Snowflake schemas will use less space to store dimension tables. This is because as a rule any normalized database produces far fewer redundant records. Denormalized data models increase the chances of data integrity problems. These issues will complicate future modifications and maintenance as well. To experienced data modelers, the snowflake schema seems more logically organized than the star schema. This is my personal opinion, not a hard fact. Query Complexity In our first two articles, we demonstrated a query that could be used on the sales model to get the quantity of all phone-type products sold in Berlin stores in The star schema query looks like this: Because the dimension tables are normalized, we need to dig deeper to get the name of the product type and the city. We have to add another JOIN for every new level inside the same dimension. In the star schema, we only join the fact table with those dimension tables we need. Joining two tables takes time because the DMBS takes longer to process the request. There is a better possibility that data will be physically closer on the disk if it lives inside the same table. Basically, a query ran against a snowflake schema data mart will execute more slowly. Speeding Things Up To speed up reporting, we can: Aggregate data to the level we need in reports. This will compress the data significantly. Only give users the data they need for analysis and reports. Which Should You Use? Consider using the snowflake schema: As the warehouse is Data Central for the company, we could save lot of space this way. When dimension tables require a significant amount of storage space. In most cases, the fact tables will be the ones that take most of the space. For instance, the dimension tables could contain a lot of redundant-but-needed attributes. In our example, we used the city attribute to describe the city where the store is located. What if we wanted a much more detailed description of the city, including the population, postal code, demographic data, etc.? If you use tools that require a snowflake schema in the background. Fortunately, most modern tools support both schemas and even the galaxy schema. Consider using the star schema: Data marts are subsets of data taken out of the central data warehouse. In this setting, saving storage space is not a priority. On the other hand, the star schema does simplify analysis. This is not just about query efficiency but also about simplifying future actions for business users. They may understand databases and know how to write queries, but why complicate things and include more joins if we can avoid it? A business user could have a template query that joins the fact table with all the dimension tables. Then they only need to add the appropriate selections and groupings. If you use tools that require a star schema in the background. No matter how similar they are, they demonstrate two different approaches and have their own benefits and

disadvantages. Personally, I would go with the snowflake schema when implementing a data warehouse to save storage space and with the star schema for data marts to make life easier for business users.

Chapter 7 : Dimensional Modeling - Star Schema [Gerardnico]

A star schema really lies at the intersection of the relational model of data and the dimensional model of data. It's really a way of starting with a dimensional model, and mapping it into SQL tables that somewhat resemble the SQL tables you get if you start from a relational model.

This chapter explains dimensional database modeling and the concepts of star schemas. In this chapter, the following topics are discussed: Modifying records is generally known as online transaction processing OLTP. Data retrieval is referred to as online analytical processing OLAP or decision support, because the information is often used to make business decisions. This section describes these data models and their structural requirements. When database records are modified, the most important requirements are update performance and data integrity. These needs are addressed by the entity relation model of organizing data. Entity relation schemas are highly normalized. This means that data redundancy is eliminated by separating the data into multiple tables. The process of normalization results in a complex schema with many tables and join paths. When database records are retrieved, the most important requirements are query performance and schema simplicity. These needs are best addressed by the dimensional model. Another name for the dimensional model is the star schema. A diagram of a star schema resembles a star, with a fact table at the center. The following figure is a sample star schema. A fact table usually contains numeric measurements, and is the only type of table with multiple joins to other tables. Surrounding the fact table are dimension tables, which are related to the fact table by a single join. Dimension tables contain data that describe the different characteristics, or dimensions, of a business. Data warehouses and data marts are usually based on a star schema. In a star schema, subjects are either facts or dimensions. You define and organize subjects according to how they are measured and whether or not they change over time. Facts change regularly, and dimensions do not change, or change very slowly. Separating facts and dimensions yields a subject-oriented design where data is stored according to logical relationships, not according to how the data was entered. This structure is easier for both users and applications to understand and navigate.

Chapter 8 : What is a Star Schema

Dimensional modeling creates a database schema that is optimized for high performance. This means fewer joins, minimized data redundancy, and operations on numbers instead of text which is almost always a more efficient use of CPU and memory.

The purpose of this article is threefold 1 Show that we will always need a data model either done by humans or machines 2 Show that physical modelling is not the same as logical modelling. In fact it is very different and depends on the underlying technology. We need both though. I illustrated this point using Hadoop at the physical layer 3 Show the impact of the concept of immutability on data modelling. Is dimensional modeling dead? Why do we need to model our data? Contrary to a common misunderstanding, it is not the only purpose of data models to serve as an ER diagram for designing a physical database. Data models represent the complexity of business processes in an enterprise. They document important business rules and concepts and help to standardise key enterprise terminology. They provide clarity and help to uncover blurred thinking and ambiguities about business processes. Furthermore, you can use data models to communicate with other stakeholders. You would not build a house or a bridge without a blueprint. So why would you build a data application such as a data warehouse without a plan? Why do we need dimensional models? Dimensional modelling is a special approach to modelling data. We also use the words data mart or star schema as synonyms for a dimensional model. Star schemas are optimised for data analytics. Have a look at the dimensional model below. It is quite intuitive to understand. We immediately see how we can slice and dice our order data by customer, product or date and measure the performance of the Orders business process by aggregating and comparing metrics. One of the core ideas about dimensional modelling is to define the lowest level of granularity in a transactional business process. Put in another way, the lowest level of granularity in a star schema is a join of the fact to all dimension tables without any aggregations. Data Modelling vs Dimensional Modelling In standard data modelling we aim to eliminate data repetition and redundancy. When a change happens to data we only need to change it in one place. This also helps with data quality. Have a look at the model below. It contains various tables that represent geographic concepts. In a normalised model we have a separate table for each entity. In a dimensional model we just have one table: In this table, cities will be repeated multiple times. Once for each city. If the country changes its name we have to update the country in many places Note: Standard data modelling is also referred to as 3NF modelling. The standard approach to data modelling is not fit for purpose for Business Intelligence workloads. A lot of tables result in a lot of joins. Joins slow things down. In data analytics we avoid them where possible. In dimensional models we de-normalize multiple related tables into one table, e. So why do some people claim that dimensional modelling is dead? I think you would agree that data modelling in general and dimensional modelling in particular is quite a useful exercise. So why do some people claim that dimensional modelling is not useful in the era of big data and Hadoop? As you can imagine there are various reasons for this. The Data Warehouse is dead Confusion First of all, some people confuse dimensional modelling with data warehousing. They claim that data warehousing is dead and as a result dimensional modelling can be consigned to the dustbin of history as well. This is a logically coherent argument. However, the concept of the data warehouse is far from obsolete. We always need integrated and reliable data for populating our BI dashboards. In the course I go into the details and explain how the data warehouse is as relevant as ever. I will also show how emerging big data tools and technologies are useful for data warehousing. In my opinion, the concept of schema on read is one of the biggest misunderstandings in data analytics. I agree that it is useful to initially store your raw data in a data dump that is light on schema. However, this argument should not be used as an excuse to not model your data altogether. The schema on read approach is just kicking down the can and responsibility to downstream processes. Someone still has to bite the bullet of defining the data types. Each and every process that accesses the schema-free data dump needs to figure out on its own what is going on. This type of work adds up, is completely redundant, and can be easily avoided by defining data types and a proper schema. The physical aspects of the model. Are there actually some valid arguments for declaring dimensional models obsolete?

There are indeed some better arguments than the two I have listed above. They require some understanding of physical data modelling and the way Hadoop works. Earlier on I briefly mentioned one of the reasons why we model our data dimensionally. In standard data modelling each real world entity gets its own table. We do this to avoid data redundancy and the risk of data quality issues creeping into our data. The more tables we have the more joins we need. Table joins are expensive, especially when we join a large numbers of records from our data sets. When we model data dimensionally we consolidate multiple tables into one. We say that we pre-join or de-normalise the data. We now have less tables, less joins, and as a result lower latency and better query performance. Take part in the discussion of this post on LinkedIn Taking de-normalization to its full conclusion Why not take de-normalisation to its full conclusion? Get rid of all joins and just have one single fact table? Indeed this would eliminate the need for any joins altogether. However, as you can imagine, it has some side effects. First of all, it increases the amount of storage required. We now need to store a lot of redundant data. With the advent of columnar storage formats for data analytics this is less of a concern nowadays. The bigger problem of de-normalization is the fact that each time a value of one of the attributes changes we have to update the value in multiple places - possibly thousands or millions of updates. One way of getting around this problem is to fully reload our models on a nightly basis. Often this will be a lot quicker and easier than applying a large number of updates. Columnar databases typically take the following approach. They first store updates to data in memory and asynchronously write them to disk. Data distribution on a distributed relational database MPP When creating dimensional models on Hadoop, e. When distributing data across the nodes in an MPP we have control over record placement. Based on our partitioning strategy, e. Have a look at the example below. Data Distribution on Hadoop This is very different from Hadoop based systems. In order to join, we need to send data across the network, which impacts performance. One strategy of dealing with this problem is to replicate one of the join tables across all nodes in the cluster. This is called a broadcast join and we use the same strategy on an MPP. As you can imagine, it only works for small lookup or dimension tables. So what do we do when we have a large fact table and a large dimension table, e. Or indeed when we have two large fact tables. Dimensional Models on Hadoop In order to get around this performance problem we can de-normalize large dimension tables into our fact table to guarantee that data is co-located. We can broadcast the smaller dimension tables across all of our nodes. For joining two large fact tables we can nest the table with the lower granularity inside the table with the higher granularity, e. Modern query engines such as Impala or Drill allow us to flatten out this data This strategy of nesting data is also useful for painful Kimball concepts such as bridge tables for representing M: N relationships in a dimensional model. In other words you can only insert and append records. If you are coming from a relational data warehouse background this may seem to be a bit odd at first. However, under the hood databases work in a similar way.

Chapter 9 : Unsupported SSL/TLS Version

Another name for the dimensional model is the star schema. A diagram of a star schema resembles a star, with a fact table at the center. The following figure is a sample star schema.

Database designer and developer, financial analyst Posted: April 14, Today, reports and analytics are almost as important as core business. Reports can be built out of your live data; often this approach will do the trick for small- and medium-sized companies without lots of data. Before we tackle basic data modeling, we need some background on the systems involved. We can roughly divide systems in two categories: OLTP systems support business processes. On the other hand, the primary purpose of the OLAP systems is analytics. These systems use summarized data, which is usually placed in a denormalized data warehousing structure like the star schema. Data Marts A data warehouse DWH is a system used to store information for use in data analysis and reporting. Data marts are areas of a data warehouses used to store information needed by a single department or even by an individual user. Think of the DWH as a building, and data marts as offices inside the building. Why are data marts needed? All relevant data is stored inside the company DWH. Most users, however, only need to access certain subsets of data, like those relating to sales, production, logistics or marketing. There are two different approaches to the data warehouse-data mart relationship: Data marts are created from the data warehouse. Data marts are created first, then combined into a data warehouse. This approach is closer to what Ralph Kimball, a data warehouse and dimensional modeling expert, advocates. Dimensional modeling, which is part of data warehouse design, results in the creation of the dimensional model. There are two types of tables involved: Dimension tables are used to describe the data we want to store. Each dimension table is its own category date, employee, store and can have one or more attributes. For each store, we can save its location at the city, region, state and country level. For each date, we can store the year, month, day of the month, day of the week, etc. This is related to the hierarchy of attributes in the dimension table. This redundancy is deliberate and done in the name of better performance. We could use date, location, and sales agent dimensions to aggregate the transform part of the ETL process and store data inside DWH. Fact tables contain the data we want to include in reports, aggregated based on values within the related dimension tables. A fact table has only columns that store values and foreign keys referencing the dimension tables. Combining all the foreign keys forms the primary key of the fact table. For instance, a fact table could store a number of contacts and the number of sales resulting from these contacts. With this info in place, we can now dig into the star schema data model. Because the fact table is in the center of the schema with dimension tables around it, it looks roughly like a star. This is especially apparent when the fact table is surrounded by five dimension tables. A variant of the star schema the centipede schema, where the fact table is surrounded by a large number of small dimension tables. Star schemas are very commonly used in data marts. We can relate them to the top-down data model approach. As we mentioned before, in most cases we could generate sales reports from the live system. After designing our star schema, an ETL process will get the data from operational database s , transform the data into the proper format for the DWH, and load the data into the warehouse. The model presented above contains of one fact table colored light red and five dimension tables colored light blue. The tables in the model are: Note that all five foreign keys together form the primary key of the table. It contains five attributes besides the primary key. Here we can clearly notice that the star schema is denormalized. Supply Orders There are a lot of similarities between this model, shown below, and the sales model. This model is intended to store the history of placed orders. We have one fact table and four dimension tables. However, the following tables are different: Advantages and Disadvantages to the Star Schema There are plenty of advantages to using the star schema. That simplifies queries and decreases query execution time. We could produce the same report directly from our OLTP system, but the query would be much more complex and it could impinge on the overall performance of the system. The following sample query for the sales model will return the quantity of all phone-type products type sold in Berlin stores in Each dimension is stored in a separate dimension table, and this causes denormalization. The Galaxy Schema We can look at the two previous models as two data marts, one for the sales department and the other for the supply department.

Each of them consists of only one fact table and a few dimensional tables. If we wanted, we could combine these two data marts into one model. This type of schema, containing several fact tables and sharing some dimension tables, is called a galaxy schema. Sharing dimension tables can reduce database size, especially where shared dimensions have many possible values. Ideally, in both data marts the dimensions are defined in the same manner. A galaxy schema, built out of our two example data marts, is shown below: The star schema is one approach to organizing a data warehouse. It is very straightforward and is most often used in data marts. If not, we should think of another approach.