DOWNLOAD PDF DATA WAREHOUSE DIMENSIONAL MODELING

Chapter 1 : Dimensional Modeling and Kimball Data Marts in the Age of Big Data and Hadoop - Sonra

Dimensional Modeling Techniques Ralph Kimball introduced the data warehouse/business intelligence industry to dimensional modeling in with his seminal book, The Data Warehouse Toolkit. Since then, the Kimball Group has extended the portfolio of best practices.

At the end of this tutorial you will become a confident dimensional data modeler. Prerequisite No previous knowledge in dimensional modeling or any other modeling is a prerequisite for this tutorial. However I assume that you already know what is a data warehouse, you have working knowledge in database and preferably you have seen or worked in a data warehousing project before. What is dimensional modelling? Definition Dimensional modeling is one of the methods of data modeling, that help us store the data in such a way that it is relatively easy to retrieve the data from the database. All the modeling techniques give us different ways to store the data. Different ways of storing data gives us different advantages. For example, ER Modeling gives us the advantage of storing data is such a way that there is less redundancy. Dimensional modeling, on the other hand, give us the advantage of storing data in such a fashion that it is easier to retrieve the information from the data once the data is stored in database. This is the reason why dimensional modeling is used mostly in data warehouses built for reporting. On the other side, dimensional model is not a good solution if your primary purpose of your data modeling is to reduce storage space requirement, reduce redundancy, speed-up loading time etc. Later on the tutorial we will learn why is it so. In dimensional model, everything is divided in 2 distinct categories - dimension or measures. Anything we try to model, must fit in one of these two categories. And then we will have 2 different categories of tables i. If you want to understand how to classify data in dimensions and facts in greater detail, please read Classify data for successful modelling In the following examples we will choose a practical business scenario and see how to identify dimensions and facts to model the scenario Step by Step Approach to Dimensional Modeling Consider the business scenario for a fastfood chain below The business objective is to create a data model that can store and report number of burgers and fries sold from a specific McDonalds outlet per day. What will be our modelling Approach in this case? The whole modeling approach is divided in 4 or 5 steps as depicted below. Identify the dimensions Dimensions are the object or context. In the above statement, we are speaking about 3 different things - we are speaking about some "food", some specific McDonalds "store" and some specific "day". So we have 3 dimensions - "food" e. Burgers and fries are 2 different members of "food" dimension. We will have to create separate tables for separate dimensions. Identify the measures Measures are the quantifiable subjects and these are often numeric in nature. Measures are not stored in the dimension tables. A separate table is created for storing measures. This table is called Fact Table. Identify the attributes or properties of dimensions Now that we have decided we need 3 tables to store the information of 3 dimensions, next we need to know what are the properties or attributes of each dimension that we need to store in our table. This is important since knowing the properties let us decide what columns are required to be created in each dimension table. As you might have guessed, each dimension might have number of different properties, but for a given context, not all of them are relevant for us. We can think of so many different attributes of food - e. But as I said, we need to check which of these attributes are relevant to us - that is - which of these attributes are required for reporting on this data. As for the given statement above, we just need to know only one attribute of the "food" dimension - i. So the structure of our food dimension will be rather easy.

Chapter 2 : What is Dimensional Model in Data Warehouse?

A Dimensional Model is a database structure that is optimized for online queries and Data Warehousing tools. It is comprised of "fact" and "dimension" tables. A "fact" is a numeric value that a business wishes to count or sum.

Conformed dimension[edit] A conformed dimension is a set of data attributes that have been physically referenced in multiple database tables using the same key value to refer to the same structure, attributes, domain values, definitions and concepts. A conformed dimension cuts across many facts. Dimensions are conformed when they are either exactly the same including keys or one is a perfect subset of the other. Most important, the row headers produced in two different answer sets from the same conformed dimension s must be able to match perfectly. Dimension tables are not conformed if the attributes are labeled differently or contain different values. Conformed dimensions come in several different flavors. At the most basic level, conformed dimensions mean exactly the same thing with every possible fact table to which they are joined. The date dimension table connected to the sales facts is identical to the date dimension connected to the inventory facts. By creating an abstract dimension, these flags and indicators are removed from the fact table while placing them into a useful dimensional framework. The nature of these attributes is usually text or various flags, e. These kinds of attributes are typically remaining when all the obvious dimensions in the business process have been identified and thus the designer is faced with the challenge of where to put these attributes that do not belong in the other dimensions. One solution is to create a new dimension for each of the remaining attributes, but due to their nature, it could be necessary to create a vast number of new dimensions resulting in a fact table with a very large number of foreign keys. The designer could also decide to leave the remaining attributes in the fact table but this could make the row length of the table unnecessarily large if, for example, the attributes is a long text string. The solution to this challenge is to identify all the attributes and then put them into one or several Junk Dimensions. An example would be an indicator about whether a package had arrived: The designer can choose to build the dimension table so it ends up holding all the indicators occurring with every other indicator so that all combinations are covered. This sets up a fixed size for the table itself which would be 2x rows, where x is the number of indicators. This solution is appropriate in situations where the designer would expect to encounter a lot of different combinations and where the possible combinations are limited to an acceptable level. In a situation where the number of indicators are large, thus creating a very big table or where the designer only expect to encounter a few of the possible combinations, it would be more appropriate to build each row in the junk dimension as new combinations are encountered. To limit the size of the tables, multiple junk dimensions might be appropriate in other situations depending on the correlation between various indicators. Junk dimensions are also appropriate for placing attributes like non-generic comments from the fact table. Such attributes might consist of data from an optional comment field when a customer places an order and as a result will probably be blank in many cases. Therefore, the junk dimension should contain a single row representing the blanks as a surrogate key that will be used in the fact table for every row returned with a blank comment field [5] Degenerate dimension[edit] A degenerate dimension is a key, such as a transaction number, invoice number, ticket number, or bill-of-lading number, that has no attributes and hence does not join to an actual dimension table. Degenerate dimensions are very common when the grain of a fact table represents a single transaction item or line item because the degenerate dimension represents the unique identifier of the parent. This is often referred to as a "role-playing dimension". Dimension table[edit] In data warehousing , a dimension table is one of the set of companion tables to a fact table. The fact table contains business facts or measures , and foreign keys which refer to candidate keys normally primary keys in the dimension tables. Contrary to fact tables, dimension tables contain descriptive attributes or fields that are typically textual fields or discrete numbers that behave like text. These attributes are designed to serve two critical purposes: Dimension attributes should be: Verbose labels consisting of full words Descriptive Complete having no missing values Discretely valued having only one value per dimension table row Quality assured having no misspellings or impossible values Dimension table rows are uniquely identified by a single key field. It is recommended that the key field be a simple integer

because a key value is meaningless, used only for joining fields between the fact and dimension tables. Dimension tables often use primary keys that are also surrogate keys. Surrogate keys are often auto-generated e. The use of surrogate dimension keys brings several advantages, including: Join processing is made much more efficient by using a single field the surrogate key Buffering from operational key management practices. This prevents situations where removed data rows might reappear when their natural keys get reused or reassigned after a long period of dormancy Mapping to integrate disparate sources Handling unknown or not-applicable connections Tracking changes in dimension attribute values Although surrogate key use places a burden put on the ETL system, pipeline processing can be improved, and ETL tools have built-in improved surrogate key processing. Every fact table is filtered consistently, so that query answers are labeled consistently. Queries can drill into different process fact tables separately for each individual fact table, then join the results on common dimension attributes. Reduced development time to market. The common dimensions are available without recreating them. Over time, the attributes of a given row in a dimension table may change. For example, the shipping address for a company may change. Kimball refers to this phenomenon as Slowly Changing Dimensions. Strategies for dealing with this kind of change are divided into three categories: Simply overwrite the old value s. Add a new row containing the new value s , and distinguish between the rows using Tuple-versioning techniques. Add a new attribute to the existing row. Common patterns[edit] Date and time [7] Since many fact tables in a data warehouse are time series of observations, one or more date dimensions are often needed. One of the reasons to have date dimensions is to place calendar knowledge in the data warehouse instead of hard-coded in an application. Having both the date and time of day in the same dimension, may easily result in a huge dimension with millions of rows. If a high amount of detail is needed it is usually a good idea to split date and time into two or more separate dimensions. A time dimension with a grain of seconds in a day will only have rows. As examples, date dimensions can be accurate to year, quarter, month or day and time dimensions can be accurate to hours, minutes or seconds. As a rule of thumb, time of day dimension should only be created if hierarchical groupings are needed or if there are meaningful textual descriptions for periods of time within the day ex. If the rows in a fact table are coming from several timezones, it might be useful to store date and time in both local time and a standard time. The standard time chosen can be a global standard time ex.

DOWNLOAD PDF DATA WAREHOUSE DIMENSIONAL MODELING

Chapter 3 : Dimensional modeling - Wikipedia

A dimensional model is a data structure technique optimized for Data warehousing tools. The concept of Dimensional Modelling was developed by Ralph Kimball and is comprised of "fact" and "dimension" tables. A Dimensional model is designed to read, summarize, analyze numeric information like values.

Getting Data into the Data Warehouse Subsystem 1: Data Profiling Subsystem 2: Change Data Capture System Subsystem 3: Data Cleansing System Subsystem 5: Error Event Schema Subsystem 6: Audit Dimension Assembler Subsystem 7: Deduplication System Subsystem 8: Conforming System Delivering: Prepare for Presentation Subsystem 9: Slowly Changing Dimension Manager Subsystem Surrogate Key Generator Subsystem Hierarchy Manager Subsystem Special Dimensions Manager Subsystem Fact Table Builders Subsystem Surrogate Key Pipeline Subsystem Late Arriving Data Handler Subsystem Dimension Manager System Subsystem Fact Provider System Subsystem Aggregate Builder Subsystem Job Scheduler Subsystem Backup System Subsystem Recovery and Restart System Subsystem Version Control System Subsystem Version Migration System Subsystem Workflow Monitor Subsystem Sorting System Subsystem Lineage and Dependency Analyzer Subsystem Problem Escalation System Subsystem Security System Subsystem Compliance Manager Subsystem Draw the High-Level Plan Step 2: Develop Default Strategies Step 4: Dimension Table Incremental Processing Step 8: Fact Table Incremental Processing Step 9:

Chapter 4 : Understanding Dimensional Modeling| Data Warehouse Information Center

Dimensional modeling (DM) is part of the Business Dimensional Lifecycle methodology developed by Ralph Kimball which includes a set of methods, techniques and concepts for use in data warehouse design.

Summary of a Business Process Suppose your organization wants to analyze customer buying trends by product line and region so that you can develop more effective marketing strategies. In this scenario, the subject area for your data model is sales. After many interviews and thorough analysis of your sales business process, suppose your organization collects the following information: Customer-base information has changed. Previously, sales districts were divided by city. Now the customer base corresponds to two regions: Region 1 for California and Region 2 for all other states. The following reports are most critical to marketing: Monthly revenue, cost, net profit by product line per vendor Revenue and units sold by product, by region, by month Monthly customer revenue Quarterly revenue per vendor Most sales analysis is based on monthly results, but you can choose to analyze sales by week or accounting period at a later date. A data-entry system exists in a relational database. To develop a working data model, you can assume that the relational database of sales information has the following properties: The product code that analysts use is stored in the catalog table as the catalog number. The product line code is stored in the stock table as the stock number. The product line name is stored as description. The product hierarchies are somewhat complicated. Each product line has many products, and each manufacturer has many products. All the cost data for each product is stored in a flat file named costs. The region information has not yet been added to the database. An important characteristic of the dimensional model is that it uses business labels familiar to end users rather than internal tables or column names. After the business process is completed, you should have all the information you need to create the measures, dimensions, and relationships for the dimensional data model. To do this you must decide what an individual low-level record in the fact table should contain. The components that make up the granularity of the fact table correspond directly with the dimensions of the data model. Thus, when you define the granularity of the fact table, you identify the dimensions of the data model. How Granularity Affects the Size of the Database The granularity of the fact table also determines how much storage space the database requires. For example, consider the following possible granularities for a fact table: Product by day by region Product by month by region The size of a database that has a granularity of product by day by region would be much greater than a database with a granularity of product by month by region because the database contains records for every transaction made each day as opposed to a monthly summation of the transactions. You must carefully determine the granularity of your fact table because too fine a granularity could result in an astronomically large database. Conversely, too coarse a granularity could mean the data is not detailed enough for users to perform meaningful queries against the database. Using the Business Process to Determine the Granularity A careful review of the information gathered from the business process should provide what you need to determine the granularity of the fact table. To summarize, your organization wants to analyze customer-buying trends by product line and region so that you can develop more effective marketing strategies. Customer by Product The granularity of the fact table always represents the lowest level for each corresponding dimension. When you review the information from the business process, the granularity for customer and product dimensions of the fact table are apparent. Customer and product cannot be reasonably reduced any further: In some cases, product might be further reduced to the level of product component because a product could be made up of multiple components. Customer by Product by District Because the customer buying trends your organization wants to analyze include a geographical component, you still need to decide the lowest level for region information. The business process indicates that in the past, sales districts were divided by city, but now your organization distinguishes between two regions for the customer base: Nonetheless, at the lowest level, your organization still includes sales district data, so district represents the lowest level for geographical information and provides a third component to further define the granularity of the fact table. Customer by Product by District by Day Customer-buying trends always occur over time, so the granularity of the fact table must include a time component. Suppose your organization decides to create

reports by week, accounting period, month, quarter, or year. At the lowest level, you probably want to choose a base granularity of day. This granularity allows your business to compare sales on Tuesdays with sales on Fridays, compare sales for the first day of each month, and so forth. The granularity of the fact table is now complete. The decision to choose a granularity of day means that each record in the time dimension table represents a day. In terms of the storage requirements, even 10 years of daily data is only about 3, records, which is a relatively small dimension table. Identifying the Dimensions and Hierarchies After you determine the granularity of the fact table, it is easy to identify the primary dimensions for the data model because each component that defines the granularity corresponds to a dimension. Figure 39 shows the relationship between the granularity of the fact table and the dimensions of the data model. The Granularity of the Fact Table Corresponds to the Dimensions of the Data Model With the dimensions customer, product, geography, time for the data model in place, the schema diagram begins to take shape. At this point, you can add additional dimensions to the primary granularity of the facttable, where the new dimensions take on only a single value under each combination of the primary dimensions. If you see that an additional dimension violates the granularity because it causes additional records to be generated, then you must revise the granularity of the facttable to accommodate the additional dimension. For this data model, no additional dimensions need to be added. You can now map out dimension elements and hierarchies for each dimension. Figure 40 shows the relationships among dimensions, dimension elements, and the inherent hierarchies. The Relationships Between Dimensions, Dimension Elements, and the Inherent Hierarchies In most cases, the dimension elements need to express the lowest possible granularity for each dimension, not because queries need to access individual low-level records, but because queries need to cut through the database in precise ways. In other words, even though the questions that a data warehousing environment poses are usually broad, these questions still depend on the lowest level of product detail. Choosing the Measures for the Fact Table The measures for the data model include not only the data itself, but also new values that you calculate from the existing data. When you examine the measures, you might discover that you need to make adjustments either in the granularity of the fact table or the number of dimensions. Another important decision you must make when you design the data model is whether to store the calculated results in the fact table or to derive these values at runtime. The information that you gather from analysis of the sales business process results in the following list of measures:

Chapter 5 : Dimensional Modeling tutorial - OLAP, data warehouse design - Knowledge Hills

Many data warehouse designers use Dimensional modeling design concepts to build data warehouses. Dimensional model is the underlying data model used by many of the commercial OLAP products available today in the market.

This could be Marketing, Sales, HR, etc. The selection of the Business process also depends on the quality of data available for that process. It is the most important step of the Data Modelling process, and a failure here would have cascading and irreparable defects. It is the process of identifying the lowest level of information for any table in your data warehouse. If a table contains sales data for every day, then it should be daily granularity. If a table contains total sales data for each month, then it has monthly granularity. During this stage, you answer questions like Do we need to store all the available products or just a few types of products? This decision is based on the business processes selected for Datawarehouse Do we store the product sale information on a monthly, weekly, daily or hourly basis? This decision depends on the nature of reports requested by executives How do the above two choices affect the database size? So, the grain is "product sale information by location by the day. These dimensions are where all the data should be stored. For example, the date dimension may contain data like a year, month and weekday. Product, Location and Time Attributes: Country, State, City, Street Address, Name Step 4 Identify the Fact This step is co-associated with the business users of the system because this is where they get access to data stored in the data warehouse. Most of the fact table rows are numerical values like price or cost per unit, etc. The fact here is Sum of Sales by product by location by time. A schema is nothing but the database structure arrangement of tables. There are two popular schemas Star Schema The star schema architecture is easy to design. It is called a star schema because diagram resembles a star, with points radiating from a center. The center of the star consists of the fact table, and the points of the star is dimension tables. The fact tables in a star schema which is third normal form whereas dimensional tables are de-normalized. Snowflake Schema The snowflake schema is an extension of the star schema. In a star schema, each dimension are normalized and connected to more dimension tables. Rules for Dimensional Modelling Load atomic data into dimensional structures. Build dimensional models around business processes. Need to ensure that every fact table has an associated date dimension table. Ensure that all facts in a single fact table are at the same grain or level of detail. Dimension tables store the history of the dimensional information. It allows to introduced entirely new dimension without major disruptions to the fact table. Dimensional also to store data in such a fashion that it is easier to retrieve the information from the data once the data is stored in the database. Compared to the normalized model dimensional table are easier to understand. Information is grouped into clear and simple business categories. The dimensional model is very understandable by the business. This model is based on business terms, so that the business knows what each fact, dimension, or attribute means. Dimensional models are deformalized and optimized for fast data querying. Many relational database platforms recognize this model and optimize query execution plans to aid in performance. Dimensional modeling creates a schema which is optimized for high performance. It means fewer joins and helps with minimized data redundancy. The dimensional model also helps to boost query performance. It is more denormalized therefore it is optimized for querying. Dimensional models can comfortably accommodate change. Dimension tables can have more columns added to them without affecting existing business intelligence applications using these tables. A dimensional model is a data structure technique optimized for Data warehousing tools. Dimension provides the context surrounding a business process event. The Attributes are the various characteristics of the dimension. A fact table is a primary table in a dimensional model. A dimension table contains dimensions of a fact. There are three types of facts 1. Five steps of Dimensional modeling are 1. Identify Business Process 2. Identify Grain level of detail 3. Build Star In Dimensional modeling, there is need to ensure that every fact table has an associated date dimension table.

Chapter 6 : Dimensional Data Model

Dimensional modeling, on the other hand, give us the advantage of storing data in such a fashion that it is easier to retrieve the information from the data once the data is stored in database. This is the reason why dimensional modeling is used mostly in data warehouses built for reporting.

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.

Chapter 7 : Dimensional Modeling

Dimensional data model is most often used in data warehousing systems. This is different from the 3rd normal form, commonly used for transactional (OLTP) type systems. As you can imagine, the same data would then be stored differently in a dimensional model than in a 3rd normal form model.

Data warehouse experts jokingly say that the world is divided into two types of people: This article will focus on the basics of dimensional modeling to help you understand data warehousing concepts better. Data can be modeled in a variety of different ways, and dimensional modeling one of them. The primary reason dimensional modeling is its ability to allow data to be stored in a way that is optimized for information retrieval once it has been stored in a database. When building a data warehouse for business intelligence, dimensional modeling allows: Dimension Table Dimension tables store the descriptive details of each business process; the what, who, where, and when. Fact Table A fact table stores transaction or event data in a numeric format. The dimensions, therefore, associated with our fact tables are product, time, store, employee, and sales. Primary and Foreign Keys Each dimension has a primary key, which is stored in the dimension table as a separate column. To reference these primary keys and therefore associated records in a dimension table, foreign keys are used, which are stored in a separate column. As seen in the fact table diagram above, foreign keys that reference primary keys to each of the associated dimensions are contained. Tying the Components Together When we bring dimensions, facts, and primary and foreign keys together, we get a dimensional modeling schema: This diagram illustrates a star schema, the most popular dimensional modeling schema used in data warehousing. In a star schema, there can be one or more fact tables, and each fact table is associated with multiple dimensions. This schema is the simplest form of a dimensional model. So how is a dimension model designed? Building a Dimensional Model Building a dimensional model requires a thorough understanding of business processes and a deliberative approach to database design. Identify a business process The first step is to gather requirements and identify a business process to be modeled. You will need to hold discussions with business users to understand their reporting requirements, how they perceive the business process, what data metrics would they want to use in their reports, etc. The second part of this stage is to consult with system experts who work with data sources for your business processes. Profile data in the source databases so you can determine how to model your dimensions. Establish granularity This refers to the level of detail you want in your reports, allowing you to determine the facts and dimensions for your dimensional model. By describing granularity in a sentence like this, you can figure out needed facts and dimensions. Identify dimensions From the granularity example above, the level of detail you require will necessitate at least time, sales, product, store, and employee dimensions. Think about how you would potentially want to use each dimension and then design attributes. For instance, the time dimension could have daily, weekly, and monthly attributes. Identify Facts Once dimensions are identified, you can move to the fact table. The fact table will contain foreign keys to each of the dimensions. Now identify the facts that will populate fact table records. In our ongoing example, facts could be price and quantity. Once you have the data warehouse design down theoretically, you will need to actually create the model on a data warehousing platform. This may require using a modeling tool like Erwin, then an ETL tool to load your data into the data warehouse, and a host of other tools to go all the way to generating reports from your dimensional models. A smarter way is to use a data warehouse automation platform where you can go from conceptualizing models all the way to getting business intelligence from them.

Chapter 8 : The Guide to Dimensional Data Modeling

Dimensional modeling, which is part of data warehouse design, results in the creation of the dimensional model. There are two types of tables involved: There are two types of tables involved: Dimension tables are used to describe the data we want to store.

This is different from the 3rd normal form, commonly used for transactional OLTP type systems. As you can imagine, the same data would then be stored differently in a dimensional model than in a 3rd normal form model. A category of information. For example, the time dimension. A unique level within a dimension. For example, Month is an attribute in the Time Dimension. The specification of levels that represents relationship between different attributes within a dimension. A fact table is a table that contains the measures of interest. For example, sales amount would be such a measure. This measure is stored in the fact table with the appropriate granularity. For example, it can be sales amount by store by day. In this case, the fact table would contain three columns: A date column, a store column, and a sales amount column. The lookup table provides the detailed information about the attributes. For example, the lookup table for the Quarter attribute would include a list of all of the quarters available in the data warehouse. Each row each quarter may have several fields, one for the unique ID that identifies the quarter, and one or more additional fields that specifies how that particular quarter is represented on a report for example, first quarter of may be represented as "Q1 " or " Q1". A dimensional model includes fact tables and lookup tables. Fact tables connect to one or more lookup tables, but fact tables do not have direct relationships to one another. Dimensions and hierarchies are represented by lookup tables. Attributes are the non-key columns in the lookup tables. Whether one uses a star or a snowflake largely depends on personal preference and business needs. Personally, I am partial to snowflakes, when there is a business case to analyze the information at that particular level.

Chapter 9 : Data design and dimensional modeling - Tutorial

Basics of Dimensional Modeling. Data warehouse and OLAP tools are based on a dimensional data model. A dimensional model is based on dimensions, facts, cubes, and schemas such as.

Data Warehouse Design â€" Inmon versus Kimball Posted on by Sakthi Rangarajan with 8 Comments Introduction We are living in the age of a data revolution, and more corporations are realizing that to leadâ€"or in some cases, to surviveâ€"they need to harness their data wealth effectively. The data warehouse, due to its unique proposition as the integrated enterprise repository of data, is playing an even more important role in this situation. There are two prominent architecture styles practiced today to build a data warehouse: This paper attempts to compare and contrast the pros and cons of each architecture style and to recommend which style to pursue based on certain factors. Background In terms of how to architect the data warehouse, there are two distinctive schools of thought: They both view the data warehouse as the central data repository for the enterprise, primarily serve enterprise reporting needs, and they both use ETL to load the data warehouse. The key distinction is how the data structures are modeled, loaded, and stored in the data warehouse. This difference in the architecture impacts the initial delivery time of the data warehouse and the ability to accommodate future changes in the ETL design. When a data architect is asked to design and implement a data warehouse from the ground up, what architecture style should he or she choose to build the data warehouse? The Inmon Approach The Inmon approach to building a data warehouse begins with the corporate data model. This model identifies the key subject areas, and most importantly, the key entities the business operates with and cares about, like customer, product, vendor, etc. From this model, a detailed logical model is created for each major entity. For example, a logical model will be built for Customer with all the details related to that entity. There could be ten different entities under Customer. All the details including business keys, attributes, dependencies, participation, and relationships will be captured in the detailed logical model. The key point here is that the entity structure is built in normalized form. Data redundancy is avoided as much as possible. This leads to clear identification of business concepts and avoids data update anomalies. The next step is building the physical model. The physical implementation of the data warehouse is also normalized. This normalized model makes loading the data less complex, but using this structure for querying is hard as it involves many tables and joins. So, Inmon suggests building data marts specific for departments. The data marts will be designed specifically for Finance, Sales, etc. Any data that comes into the data warehouse is integrated, and the data warehouse is the only source of data for the different data marts. This ensures that the integrity and consistency of data is kept intact across the organization. The data warehouse truly serves as the single source of truth for the enterprise, as it is the only source for the data marts and all the data in the data warehouse is integrated. Data update anomalies are avoided because of very low redundancy. This makes ETL process easier and less prone to failure. The business processes can be understood easily, as the logical model represents the detailed business entities. Very flexible â€" As the business requirements change or source data changes, it is easy to update the data warehouse as one thing is in only one place. Can handle varied reporting needs across the enterprise. Here are some of the disadvantages of Inmon method: The model and implementation can become complex over time as it involves more tables and joins. Need resources who are experts in data modeling and of the business itself. These type of resources can be hard to find and are often expensive. The initial set-up and delivery will take more time, and management needs to be aware of this. More ETL work is needed as the data marts are built from the data warehouse. A fairly large team of specialists need to be around to successfully manage the environment Breslin, The Kimball Approach The Kimball approach to building the data warehouse starts with identifying the key business processes and the key business questions that the data warehouse needs to answer. The key sources operational systems of data for the data warehouse are analyzed and documented. ETL software is used to bring data from all the different sources and load into a staging area. From here, data is loaded into a dimensional model. Here the comes the key difference: The fundamental concept of dimensional modeling is the star schema. In the star schema, there is typically a fact table surrounded by many dimensions. The fact table has all the measures that

are relevant to the subject area, and it also has the foreign keys from the different dimensions that surround the fact. The dimensions are denormalized completely so that the user can drill up and drill down without joining to another table. Multiple star schemas will be built to satisfy different reporting requirements. So, how is integration achieved in the dimensional model? The key dimensions, like customer and product, that are shared across the different facts will be built once and be used by all the facts Kimball et al. This ensures that one thing or concept is used the same way across the facts. This is the document where the different facts are listed vertically and the conformed dimensions are listed horizontally. Where ever the dimensions play a foreign key role in the fact, it is marked in the document. This serves as an anchoring document showing how the star schemas are built and what is left to build in the data warehouse. Quick to set-up and build, and the first phase of the data warehousing project will be delivered quickly. The star schema can be easily understood by the business users and is easy to use for reporting. Most BI tools work well with star schema. The foot print of the data warehousing environment is small;it occupies less space in the database and it makes the management of the system fairly easier. The performance of the star schema model is very good. This is known to be a very effective database operation. A small team of developers and architects is enough to keep the data warehouse performing effectively Breslin, Works really well for department-wise metrics and KPI tracking, as the data marts are geared towards department-wise or business process-wise reporting. Drill-across, where a BI tool goes across multiple star schemas to generate a report can be successfully accomplished using confirmed dimensions. Here are some of the disadvantages of the Kimball method: Redundant data can cause data update anomalies over time. Adding columns to the fact table can cause performance issues. This is because the fact tables are designed to be very deep. If new columns are to be added, the size of the fact table becomes much larger and will not perform well. This makes the dimensional model hard to change as the business requirements change. Cannot handle all the enterprise reporting needs because the model is oriented towards business processes rather than the enterprise as a whole. Integration of legacy data into the data warehouse can be a complex process. Deciding Factors Now that we have seen the pros and cons of the Kimball and Inmon approaches, a question arises. Which approach should be used when? This question is faced by data warehouse architects every time they start building a data warehouse. Here are the deciding factors that can help an architect choose between the two: Reporting Requirements â€” If the reporting requirements are strategic and enterprise-wide and integrated reporting is needed, then Inmon works best. Project Urgency â€” If the organization has enough time to wait for the first delivery of the data warehouse say 4 to 9 months , then Inmon approach can be followed. If there is very little time for the data warehouse to be up and running say, 2 to 3 months then the Kimball approach is best Breslin, Future Staffing Plan â€” If the company can afford to have a large sized team of specialists to maintain the data warehouse, then the Inmon method can be pursued. If the future plan for the team is to be thin, then Kimball is more suited. Frequency of Changes â€” If the reporting requirements are expected to change more rapidly and the source systems are known to be volatile, then the Inmon approach works better, as it is more flexible. If the requirements and source systems are relatively stable, the Kimball method can be used. Organization Culture â€” If the sponsors of the data warehouse and the managers of the firm understand the value proposition of the data warehouse and are willing to accept long-lasting value from the data warehouse investment, the Inmon approach is better. If the sponsors do not care about the concepts but want a solution to get better at reporting, then the Kimball approach is enough. Conclusion It has been proven that both the Inmon and Kimball approach work for successfully delivering data warehouses. In a hybrid model, the data warehouse is built using the Inmon model, and on top of the integrated data warehouse, the business process oriented data marts are built using the star schema for reporting. We cannot generalize and say that one approach is better than the other; they both have their advantages and disadvantages, and they both work fine in different scenarios. The architect has to select an approach for the data warehouse depending on the different factors; a few key ones were identified in this paper. Accessed May 22, Building the Data Warehouse, Fourth Edition. Accessed May 23, The Data Warehouse Toolkit: Accessed May 26, Accessed May 25,