**Dimensional data modelling:**

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.

To understand dimensional data modeling, let's define some of the terms commonly used in this type of modeling:

Dimension: A category of information. For example, the time dimension.

Attribute: A unique level within a dimension. For example, Month is an attribute in the Time Dimension.

Hierarchy: The specification of levels that represents relationship between different attributes within a dimension. For example, one possible hierarchy in the Time dimension is Year → Quarter → Month → Day.

[Fact Table](https://www.1keydata.com/datawarehousing/fact-table-granularity.html): 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.

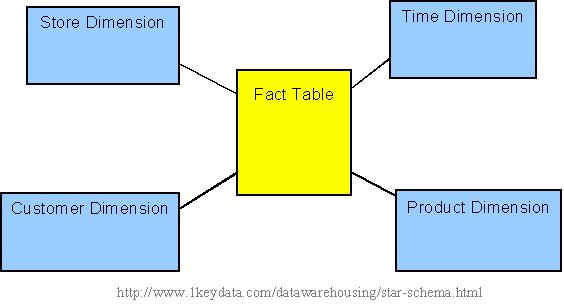
Lookup Table: The lookup table provides the detailed information about the attributes. For example, the lookup table for the Quarter attribute would include a list of all of the quarters available in the data warehouse. Each row (each quarter) may have several fields, one for the unique ID that identifies the quarter, and one or more additional fields that specifies how that particular quarter is represented on a report (for example, first quarter of 2001 may be represented as "Q1 2001" or "2001 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.

In designing data models for data warehouses / data marts, the most commonly used schema types are [Star Schema](https://www.1keydata.com/datawarehousing/star-schema.html)and [Snowflake Schema](https://www.1keydata.com/datawarehousing/snowflake-schema.html).

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*

In the star schema design, a single object (the [fact table](https://www.1keydata.com/datawarehousing/fact-table-types.html)) sits in the middle and is radically connected to other surrounding objects (dimension lookup tables) like a star. Each dimension is represented as a single table. The primary key in each dimension table is related to a foreign key in the fact table.

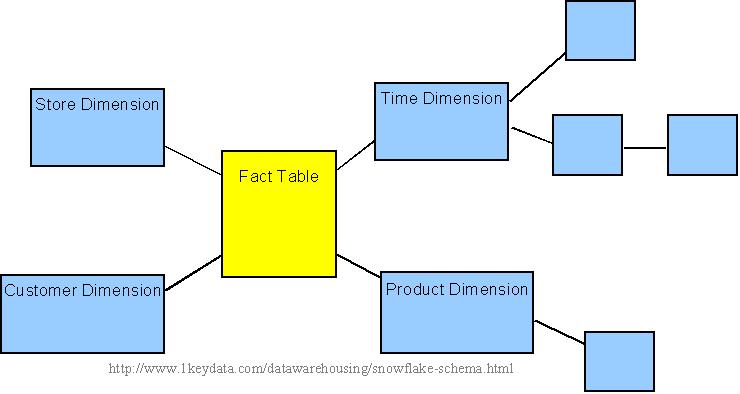
  Sample star schema  
  


All measures in the fact table are related to all the dimensions that fact table is related to. In other words, they all have the same level of granularity.

A star schema can be simple or complex. A simple star consists of one fact table; a complex star can have more than one fact table.

Let's look at an example: Assume our data warehouse keeps store sales data, and the different dimensions are time, store, product, and customer. In this case, the figure on the left represents our star schema. The lines between two tables indicate that there is a primary key / foreign key relationship between the two tables. Note that different dimensions are not related to one another.

The snowflake schema is an extension of the [star schema](https://www.1keydata.com/datawarehousing/star-schema.html), where each point of the star explodes into more points. In a star schema, each dimension is represented by a single dimensional table, whereas in a snowflake schema, that dimensional table is normalized into multiple lookup tables, each representing a level in the dimensional hierarchy.

Sample snowflake schema

For example, the Time Dimension that consists of 2 different hierarchies:

1. Year → Month → Day   
2. Week → Day

We will have 4 lookup tables in a snowflake schema: A lookup table for year, a lookup table for month, a lookup table for week, and a lookup table for day. Year is connected to Month, which is then connected to Day. Week is only connected to Day. A sample snowflake schema illustrating the above relationships in the Time Dimension is shown to the right.

The main advantage of the snowflake schema is the improvement in query performance due to minimized disk storage requirements and joining smaller lookup tables. The main disadvantage of the snowflake schema is the additional maintenance efforts needed due to the increase number of lookup tables

The first step in designing a fact table is to determine the granularity of the fact table. By granularity, we mean the lowest level of information that will be stored in the fact table. This constitutes two steps:

1. Determine which dimensions will be included.
2. Determine where along the hierarchy of each dimension the information will be kept.

The determining factors usually goes back to the requirements.

Which Dimensions To Include

Determining which dimensions to include is usually a straightforward process, because business processes will often dictate clearly what are the relevant dimensions.

For example, in an off-line retail world, the dimensions for a sales fact table are usually time, geography, and product. This list, however, is by no means a complete list for all off-line retailers. A supermarket with a Rewards Card program, where customers provide some personal information in exchange for a rewards card, and the supermarket would offer lower prices for certain items for customers who present a rewards card at checkout, will also have the ability to track the customer dimension. Whether the data warehousing system includes the customer dimension will then be a decision that needs to be made.

What Level Within Each Dimension To Include

Determining which part of hierarchy the information is stored along each dimension is not an exact science. This is where user requirement (both stated and possibly future) plays a major role.

In the above example, will the supermarket wanting to do analysis along at the hourly level? (*i.e.*, looking at how certain products may sell by different hours of the day.) If so, it makes sense to use 'hour' as the lowest level of granularity in the time dimension. If daily analysis is sufficient, then 'day' can be used as the lowest level of granularity. Since the lower the level of detail, the larger the data amount in the fact table, the granularity exercise is in essence figuring out the sweet spot in the tradeoff between detailed level of analysis and data storage.

Note that sometimes the users will not specify certain requirements, but based on the industry knowledge, the data warehousing team may foresee that certain requirements will be forthcoming that may result in the need of additional details. In such cases, it is prudent for the data warehousing team to design the fact table such that lower-level information is included. This will avoid possibly needing to re-design the fact table in the future. On the other hand, trying to anticipate all future requirements is an impossible and hence futile exercise, and the data warehousing team needs to fight the urge of the "dumping the lowest level of detail into the data warehouse" symptom, and only includes what is practically needed. Sometimes this can be more of an art than science, and prior experience will become invaluable here.

Types of Facts

There are three types of facts:

* Additive: Additive facts are facts that can be summed up through all of the dimensions in the fact table.
* Semi-Additive: Semi-additive facts are facts that can be summed up for some of the dimensions in the fact table, but not the others.
* Non-Additive: Non-additive facts are facts that cannot be summed up for any of the dimensions present in the fact table.

Let us use examples to illustrate each of the three types of facts. The first example assumes that we are a retailer, and we have a fact table with the following columns:

|  |
| --- |
| Date |
| Store |
| Product |
| Sales\_Amount |

The purpose of this table is to record the sales amount for each product in each store on a daily basis. Sales\_Amountis the fact. In this case, Sales\_Amount is an additive fact, because you can sum up this fact along any of the three dimensions present in the fact table -- date, store, and product. For example, the sum of Sales\_Amount for all 7 days in a week represents the total sales amount for that week.

Say we are a bank with the following fact table:

|  |
| --- |
| Date |
| Account |
| Current\_Balance |
| Profit\_Margin |

The purpose of this table is to record the current balance for each account at the end of each day, as well as the profit margin for each account for each day. Current\_Balance and Profit\_Margin are the facts. Current\_Balance is a semi-additive fact, as it makes sense to add them up for all accounts (what's the total current balance for all accounts in the bank?), but it does not make sense to add them up through time (adding up all current balances for a given account for each day of the month does not give us any useful information). Profit\_Margin is a non-additive fact, for it does not make sense to add them up for the account level or the day level.

Types of Fact Tables

Based on the above classifications, there are two types of fact tables:

* Cumulative: This type of fact table describes what has happened over a period of time. For example, this fact table may describe the total sales by product by store by day. The facts for this type of fact tables are mostly additive facts. The first example presented here is a cumulative fact table.
* Snapshot: This type of fact table describes the state of things in a particular instance of time, and usually includes more semi-additive and non-additive facts. The second example presented here is a snapshot fact table.
* The "Slowly Changing Dimension" problem is a common one particular to data warehousing. In a nutshell, this applies to cases where the attribute for a record varies over time. We give an example below:
* Christina is a customer with ABC Inc. She first lived in Chicago, Illinois. So, the original entry in the customer lookup table has the following record:

|  |  |  |
| --- | --- | --- |
| Customer Key | Name | State |
| 1001 | Christina | Illinois |

* At a later date, she moved to Los Angeles, California on January, 2003. How should ABC Inc. now modify its customer table to reflect this change? This is the "Slowly Changing Dimension" problem.
* There are in general three ways to solve this type of problem, and they are categorized as follows:
* [Type 1](https://www.1keydata.com/datawarehousing/slowly-changing-dimensions-type-1.html): The new record replaces the original record. No trace of the old record exists.
* [Type 2](https://www.1keydata.com/datawarehousing/slowly-changing-dimensions-type-2.html): A new record is added into the customer dimension table. Therefore, the customer is treated essentially as two people.
* [Type 3](https://www.1keydata.com/datawarehousing/slowly-changing-dimensions-type-3.html): The original record is modified to reflect the change.
* We next take a look at each of the scenarios and how the data model and the data looks like for each of them. Finally, we compare and contrast among the three alternatives.