The dimensional model of a business process is made up of two components: measurements and their context. Known as facts and dimensions, these components are organized into a database design that facilitates a wide variety of analytic usage. Implemented in a relational database, the dimensional model is called a star schema. Implemented in a multidimensional database, it is known as a cube. If any part of your data warehouse includes a star schema or a cube, it leverages dimensional design.

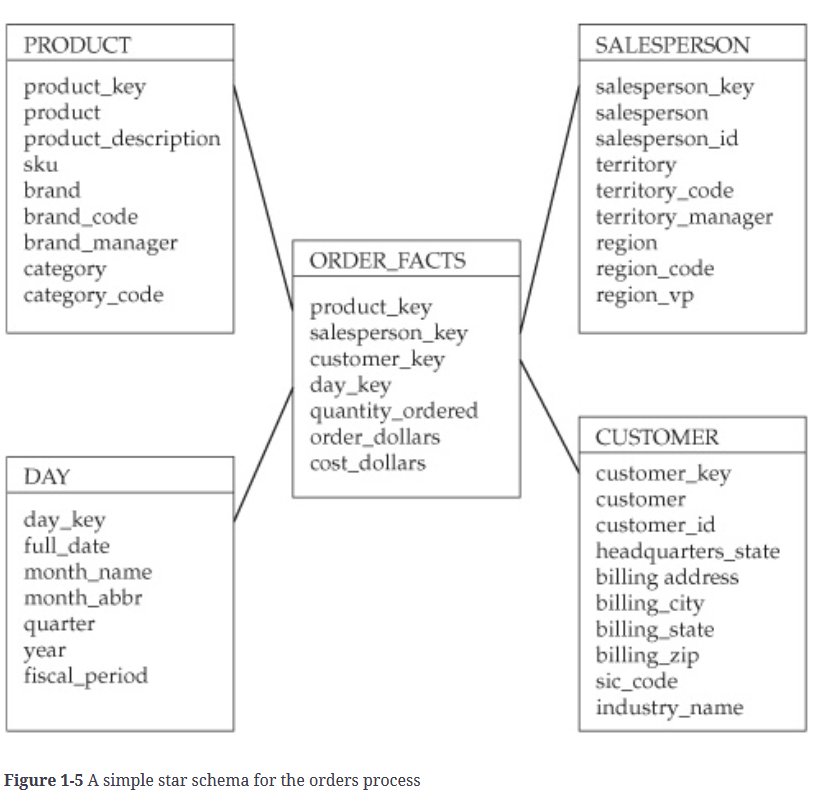
The principles of dimensional design have evolved as a direct response to the unique requirements of analytic systems.

Analytic systems and operational systems serve fundamentally different purposes. An operational system supports the execution of a business process, while an analytic system supports the evaluation of the process.

The founding principle of dimensional design is disarmingly simple. Dimensional design supports analysis of a business process by modelling how it is measured.

A dimensional design for a relational database is called a star schema. Related dimensions are grouped as columns in dimension tables, and the facts are stored as columns in a fact table. The star schema gets its name from its appearance: when drawn with the fact table in the centre, it looks like a star or asterisk. In a star schema, a dimension table contains columns representing dimensions.

The dimension tables serve to provide the rich context needed for the study of facts. In queries and reports, the dimensions will be used to specify how facts will be rolled up—their level of aggregation. Dimension values may be used to filter reports. They will be used to provide context for each measurement, usually in the form of textual labels that precede facts on each row of a report. They may also be used to drive master-detail relationships, subtotalling, cross-tabulation, or sorts.



It is not necessary to isolate repeating values in an environment that does not support transaction processing. Designers do occasionally perform additional normalization within dimensions, although they usually avoid doing so. In such cases, the schema is referred to as a snowflake. The additional tables that result are sometimes called outriggers.

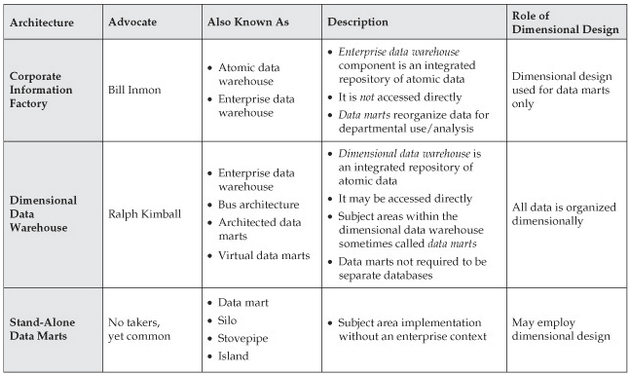
In a star schema, each dimension table is given a surrogate key. This column is a unique identifier, created exclusively for the data warehouse. Surrogate keys are assigned and maintained as part of the process that loads the star schema. The surrogate key has no intrinsic meaning; it is typically an integer. Surrogate keys are sometimes referred to as warehouse keys. The surrogate key is the primary key of the dimension table.

Dimension tables also contain key columns that uniquely identify something in an operational system. These key columns are referred to as natural keys. The separation of surrogate keys and natural keys allows the data warehouse to track changes, even if the originating operational system does not. While it would also be possible to support change tracking by supplementing a natural key with a sequence number, the surrogate key allows fact and dimension tables to be joined based on a single column.

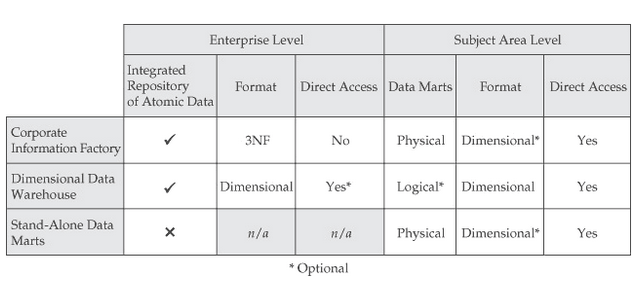
The term slowly changing dimension refers to the manner in which a dimensional schema responds to changes in a source system.

Each row in the fact table stores facts at a specific level of detail. This level of detail is known as the fact table’s grain.

The concept of a data mart becomes a logical distinction; the data mart is a subject area within the data warehouse.



As a result of their common scope, the two enterprise architectures share an architectural characteristic in common: each has a single integrated repository of atomic data. In the Corporate Information Factory, this repository is called the enterprise data warehouse. In the dimensional data warehouse, this repository is called... the dimensional data warehouse. The integrated nature of the central repository is consistent with an enterprise focus.



A well-developed set of dimension tables provides powerful and diverse analytic capabilities. The dimensions provide contextual information, without which reports would be meaningless. Successful dimension design hinges on the proper use of keys, the development of a richly detailed set of dimension columns, and a rejection of the urge to save space.

**Dimension Table Features**

A well-developed set of dimension tables provides powerful and diverse analytic capabilities. Successful dimension design hinges on the proper use of keys, the development of a richly detailed set of dimension columns, and a rejection of the urge to save space.

**TIP** Assign each dimension table a surrogate key. This single column will be used to uniquely identify each row in the table

##### **Surrogate Keys and Natural Keys**

In a star schema, each dimension table is assigned a surrogate key. This key is not a carryover from an operational system. It is created especially for the data warehouse or data mart. Surrogate keys are usually integers, generated and managed as part of the extract, transform, load (ETL) process that loads the star schema. The key values have no intrinsic meaning and are not of interest to users of the data warehouse. In each row of the dimension table, the surrogate has a unique value.

The natural keys are identifiers carried over from source systems. They may not uniquely identify a row in the data warehouse, but they do identify a corresponding entity in the source system.

Were the customer dimension table to use the same customer\_id to identify unique rows, it would be able to store only one row for customer\_id 404777. It would be unable to maintain the history of the address. By using a surrogate key, it becomes possible to maintain two versions of customer\_id 404777. This technique is known as a type 2 slow change.

It is not common practice to use version numbers or time stamps as part of a unique identifier. Surrogate keys simplify the schema design and allow for clean, single-column joins. Time stamps are frequently included in dimension tables, but not as part of the unique identifier.

##### **Rich Set of Dimensions**

Dimensions provide context for facts. Without context, facts are impossible to interpret.

Dimensions and their values add meaning in many ways:

• They are used to filter queries or reports.

• They are used to control the scope of aggregation for facts.

• They are used to order or sort information.

• They accompany facts to provide context on reports.

• They are used to define master–detail organization, grouping, subtotalling, and summarization.

Put to these uses, dimensions unlock the value of facts. Dimensions can be added to queries in different combinations to answer a wide variety of questions. The larger the set of dimension attributes, the more ways that facts can be analysed.

TIP It is not always clear whether a numeric data element is a fact or a dimension. When in doubt, pay close attention to how it will be used. If the element values are used to filter queries, order data, control aggregation, or drive master–detail relationships, it is most likely a dimension.

While unit amounts are dimensions, extended amounts are facts. Multiplying a unit amount by the transaction quantity produces a value that can be aggregated or summarized. The unit amount is a useful dimension, and the extended amount is a useful fact. Both have their place in the dimensional design.

Dimension attributes are grouped into tables that represent major categories of reference information. Junk dimensions collect miscellaneous attributes that do not share a natural affinity. When principles of normalization are applied to a dimension table, the result is called a snowflake. Though not the norm, snowflakes may be useful in the presence of specific software tools. Unlike entity-relationship design, dimensional design fully embraces redundant storage of information.

Fact Table Features

Every fact table represents a business process by capturing measurements that describe it. It is crucial that all relevant measurements be represented, even when some appear redundant. Some facts, however, cannot be aggregated. These non-additive facts are usually broken down into other facts that can.

The level of detail at which the fact table records information is referred to as its grain. It is important to identify the grain of each fact table clearly and avoid situations of mixed grain. Fact tables do not contain rows for every combination of dimension values. Instead, they exhibit a characteristic called sparsity. On occasion, the fact table may host one or more degenerate dimensions. As you will see, these columns may participate in the grain of the fact table.

Grain

The level of detail represented by a fact table row is referred to as its grain. Declaring the grain of a fact table is an important part of the schema design process. It ensures there is no confusion about the meaning of a fact table row, and guarantees all facts will be recorded at the same level of detail.

Grain may be described in a number of ways. Many schema designers describe grain simply by enumerating the associated dimensions.

Set the fact table grain at the lowest level of detail possible. This guideline helps ensure maximum analytic flexibility. It can be relaxed if there is a separate repository for granular data, but may limit future utility.

Fact Tables and Processes

The fact table is the engine for business process measurement. It is the locus for storage of the detailed measurements that describe the process. The facts are accompanied by foreign keys that provide dimensional context for each measurement.

Capturing Facts

As the locus for process measurement, the fact table should contain every fact relevant to the process it describes, even if some of the facts can be derived from others. Facts are stored at a specific level of detail but can be rolled up to various levels of dimensionality.

Example: Some designers may be tempted to eliminate the margin dollars, allowing it to be computed within reports, within a view, or through the semantic layer provided by a business intelligence product. Storage of the fact in the fact table, however, allows margin dollars to be computed as part of the ETL process. This guarantees consistent computation of margin dollars and consistent representation regardless of the tools being used.

Explicit storage of the relevant fact enables performance and consistency.

Degenerate Dimensions

Sometimes, it is not possible to sort all the dimensions associated with a business into a neat set of tables. In situations like this, it may be appropriate to store one or more dimensions in the fact table. When this is done, the dimension column is called a degenerate dimension.

Sparsity

Rows are recorded in fact tables to represent the occurrence of business activities. This means that fact tables do not contain a row for every possible combination of dimension values. The number of combinations that appear in the fact table is relatively small in comparison to the number of possible combinations. This characteristic of fact tables is called sparsity.

Dimension tables

• Dimension tables contain natural keys and surrogate keys. This allows the analytic schema to track history independently of the source.

• Dimension tables should be wide. A rich set of dimensional attributes enables a powerful analytic environment. Columns should be provided for codes and their associated descriptions, concatenated fields as well as their parts, common combinations of values, and descriptive representation of flags.

• Some dimensions are numeric; they can be distinguished from facts based on how they are used.

• Dimension tables are not placed in third normal form.

• Junk dimensions accumulate unrelated dimension attributes.

• Behavioural dimensions are derived from facts to produce powerful analytic options.

Fact tables

• Fact tables contain compact rows composed of foreign key references to dimensions, and facts.

• Fact tables should contain all facts relevant to a process, even if some can be computed from others.

• Non-additive facts such as ratios should be decomposed into fully additive components, and computed at report creation time.

• Fact tables are sparse; they record rows only when something happens.

• It is crucial that the grain of a fact table can be stated, either in dimensional terms or with respect to a business term.

• A dimension stored in a fact table is called a degenerate dimension. This technique is usually reserved for transaction identifiers that exhibit high cardinality.

Slow changes

• The warehouse responds to changes in source data through a process known as slowly changing dimensions.

• A type 1 slow change overwrites a dimension attribute when its corresponding source changes. The dimension table does not reflect history, and the historic context of existing facts is altered.

• A type 2 slow change creates a new version of the dimension row when the source value for one of its attributes changes. The dimension table maintains a version history, although it is not tied to time. The historic context of historic facts is preserved.

Cubes

• A dimensional model can also be implemented in a multidimensional database, where it is known as a cube.

• Cubes enable a fast and powerful form of interaction known as OLAP.

• The languages that support interaction with cubes support some types of analysis that are hard to express using SQL.

• Storage requirements increase as dimension attributes are added or the number of transactions increases.

• Cubes can serve as primary dimensional data stores but have limited scalability.

• Cubes can serve as a powerful supplement to a star schema, enabling focused and interactive analysis.

**TIP** For a given pair of facts, ask these questions:

1. Do these facts occur simultaneously?

2. Are these facts available at the same level of detail (or grain)?

If the answer to either of these questions is “no,” the facts represent different processes.

When two facts do not describe events at the same point in time, or are not specified at the same grain, they describe different processes.

For example, consider measurements such as quantity ordered and quantity shipped. Orders and shipments do not necessarily occur simultaneously. When an order is placed, information about shipments has yet to be determined. Shipment information is finalized later. Quantity ordered and quantity shipped also fail to share the same level of detail or grain. Shipment quantities are associated with specific shippers, while order quantities are not.

In this case, quantity ordered and quantity shipped failed both tests. Orders and shipments are two separate processes. If there will be people who want to analyse either process on its own, it will necessitate multiple fact tables. To understand why, we will look at these examples in more detail.

A dimension table can be thought of as the parent in a parent–child relationship with a fact table. If the dimension is related to other fact tables, child rows in each of the fact tables can be thought of as siblings; they share a common parent. For example, a given product may have multiple corresponding rows, or “children,” in an order\_facts table. The same product may also have one or more child rows in shipment\_facts.

When a SQL query attempts to join siblings together, either directly or through a common parent, the RDBMS will match each child from one table with each of its siblings in the other. The result is known as a Cartesian product. This occurs when two fact tables are joined directly together or through a common dimension.

**TIP** Never attempt to join to two fact tables, either directly or through a common dimension. This can produce inaccurate results.

In SQL terms, a full outer join is required. That is to say, it is important to include all data from each result set, even if there is a row in one set without a corresponding row in the other set.