**Mastering Dimensional Modeling: Techniques and Best Practices**

Dimensional modelling is a critical aspect of data warehousing and business intelligence. It involves structuring data to facilitate efficient query performance and easy access to information. This guide will explore the essentials of dimensional modelling, its components, and best practices to ensure your data warehouse is optimized for business analytics.

**Key Components of Dimensional Modelling**

**Fact Tables**

**Definition:** Fact tables are the cornerstone of dimensional modelling, storing quantitative data for analysis. They typically contain metrics or measures, such as sales amounts, quantities, and other key performance indicators (KPIs).

**Characteristics:**

Central tables in star and snowflake schemas.

Include foreign keys to dimension tables.

Store numeric data for analytical processing.

**Example:** A sales fact table might include columns for SalesAmount, UnitsSold, ProductID, CustomerID, StoreID, and DateID.

**Dimension Tables**

**Definition:** Dimension tables provide context to the data stored in fact tables. They include descriptive attributes that explain the facts.

**Characteristics:**

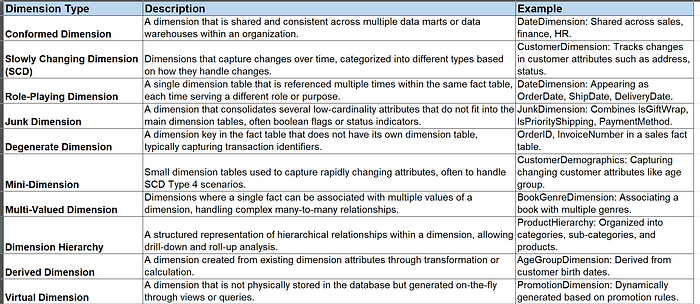
Contain textual or descriptive data.

Connected to fact tables via foreign keys.

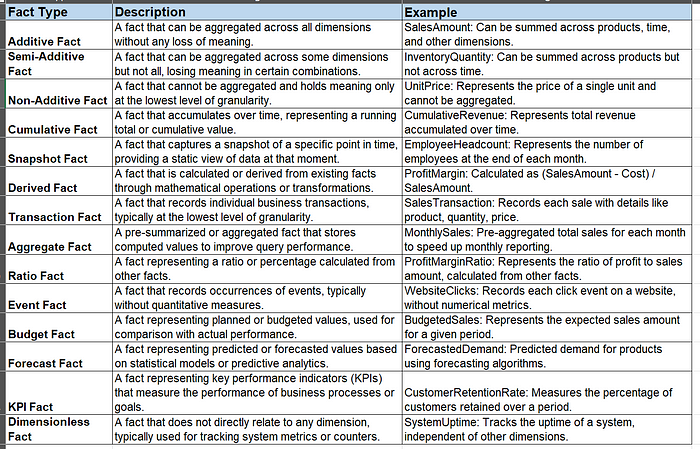
Help filter, group, and label facts for analysis.

**Example:** A product dimension table might include ProductID, ProductName, Category, Brand, and Price.

**Types of Dimensions**

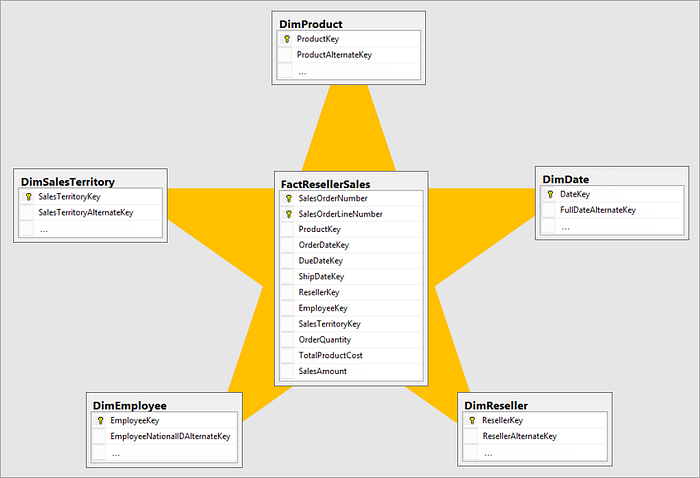


**Types of Facts**



**Dimensional Modelling Schemas**

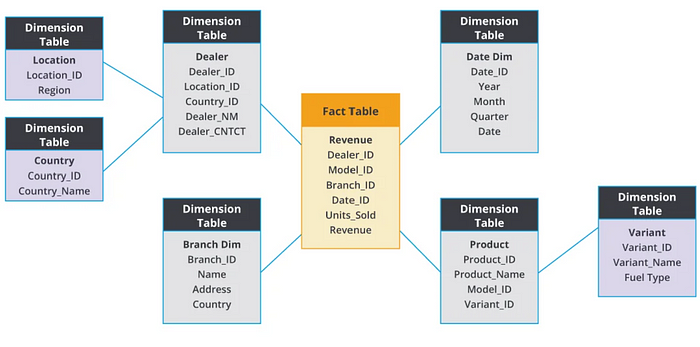
**Star Schema**



Star Schema (Image Source: Microsoft Documentation <https://learn.microsoft.com/en-us/power-bi/guidance/star-schema>)

Structure: The star schema is the simplest form of dimensional modelling. It consists of a central fact table connected to multiple dimension tables.

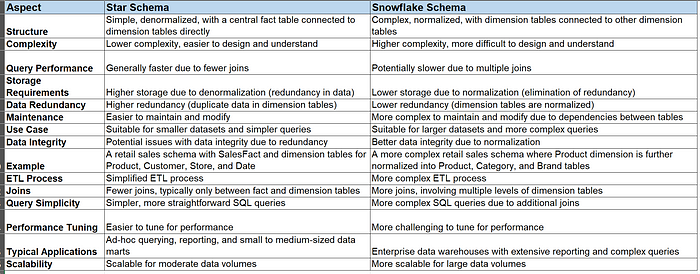
**Snowflake Schema**



Snowflake Schema (Image Source: Streamsets <https://streamsets.com/blog/schemas-data-warehouses-star-galaxy-snowflake/>)

Structure: The snowflake schema is a more normalized version of the star schema. Dimension tables are further normalized into related tables.

**Difference Between Star Schema and Snowflake Schema**



**Benefits of Dimensional Modelling**

1) **Enhanced Performance:** Optimized for fast query performance, allowing users to retrieve data quickly.

2) **Simplified Design:** Easy to understand and use, making it accessible for business users.

3) **Scalability:**Flexible structure that adapts to growing business needs and new data sources.

4) **Improved Data Quality:** Ensures consistent and accurate reporting, crucial for business decision-making.

**Best Practices in Dimensional Modelling**

**Granularity**

**Definition:** Granularity refers to the level of detail in the fact table. Higher granularity means more detailed data, which can provide deeper insights but requires more storage.

**Considerations:** Determine the granularity based on business needs and storage capabilities. Common granularities include daily, monthly, or transaction-level data.

**Generalized Design Pattern Steps for Star or Snowflake Schema**

1. **Identify Business Process:** Determine the core business process to be analyzed (e.g., sales, inventory).

2. **Define the Grain:** Establish the level of detail for the fact table (e.g., daily sales transactions).

3. **Identify Facts:**Determine the quantitative data to be stored (e.g., sales amount, units sold).

4. **Identify Dimensions:** Define the descriptive attributes needed for analysis (e.g., product, customer, store, date).

5. **Normalize Dimensions (Only applicable for Snowflake Schema):** Break down dimension tables into sub-tables to eliminate redundancy.

6. **Create Schema Diagram:**Design the schema with the fact table in the centre and dimension tables radiating out.

**Performance optimization in dimensional models**

Performance optimization in dimensional models is crucial for ensuring that data warehouses deliver timely and accurate insights, even as the volume of data grows. Here are some strategies and best practices to enhance the performance of dimensional models:

1. Create indexes on primary keys of dimension tables and foreign keys in fact tables.

2. Use clustered indexes on columns that are frequently used in range queries, such as date columns.

3. Partition fact tables by date ranges (e.g., by month or year) to improve query performance on date-based queries.

4. Combine range and list partitioning to further optimize performance.

5. Use materialized views to store pre-computed aggregates for common queries.

6. Set up appropriate refresh strategies (e.g., on-demand, scheduled) to keep materialized views updated.

7. Create indexes on materialized views to enhance query performance further.

8. Combine dimension tables into a single table if they are frequently joined.

9. Include hierarchical data (e.g., product categories) directly in the dimension table.

10. Include key attributes from related tables to minimize joins.

11. Create aggregation tables at various levels (e.g., daily, monthly, yearly).

12. Implement ETL processes to update aggregation tables regularly.

13. Implement incremental data loads to update only changed data rather than reloading entire datasets.

14. Use parallel processing to handle large data volumes more efficiently.

15. Ensure robust data validation to maintain data quality and integrity.

16. Use appropriate join types and ensure join conditions are indexed.

17. Ensure sufficient CPU, memory, and disk I/O capabilities.

18. Optimize database settings such as buffer cache size, temp space, and parallel query settings.

19. Distribute the query load across multiple servers or nodes.

20. Use columnar storage formats that are more efficient for read-heavy workloads typical in data warehousing.

**Real-world applications of dimensional modelling in business intelligence**

**1) Financial Services**

**Financial Reporting and Analysis**

**Application:**Banks and financial institutions use dimensional modelling for comprehensive financial reporting and performance analysis.

**Example:**A bank employs a star schema to consolidate transaction data from various branches and ATMs. This helps in generating consolidated financial statements, tracking account balances, and analysing transaction patterns to detect fraudulent activities.

**Risk Management**

**Application:**Financial firms use dimensional models to assess and manage risk.

**Example:**An investment firm implements a dimensional model to evaluate the risk exposure of different portfolios. By analyzing historical performance data and market trends, the firm can make informed decisions to mitigate risks and optimize returns.

**2) Healthcare**

**Patient Care and Outcomes**

**Application:**Hospitals use dimensional modelling to improve patient care and outcomes by analyzing clinical and operational data.

**Example:**A hospital employs a star schema to integrate patient records, treatment histories, and outcomes data. This enables healthcare providers to identify patterns in patient care, optimize treatment plans, and improve overall healthcare quality.

**Resource Management**

**Application:**Healthcare organizations use dimensional models to manage resources such as staff, equipment, and facilities.

**Example:**A healthcare system uses a dimensional model to track the utilization of hospital beds, medical equipment, and staff schedules. This helps in optimizing resource allocation, reducing wait times, and improving operational efficiency.

**Summary**

Dimensional modelling is a foundational technique in data warehousing, crucial for building efficient and user-friendly data repositories. By understanding and implementing the key components and best practices, businesses can unlock the full potential of their data for insightful analytics and decision-making. Choosing between star and snowflake schemas depends on the specific requirements of the data warehouse, including performance needs, storage considerations, and complexity management. Understanding and implementing these design patterns effectively can greatly enhance the efficiency and usability of your data warehousing solutions.

A student attending one of Kimball Group’s recent onsite dimensional modeling classes asked me for a list of “Kimball’s Commandments” for dimensional modeling. We will refrain from using religious terminology, but let us just say the following are not-to-be-broken rules together with less stringent rule-of-thumb recommendations.

**Rule #1: Load detailed atomic data into dimensional structures.**

Dimensional models should be populated with bedrock atomic details to support the unpredictable filtering and grouping required by business user queries. Users typically do not need to see a single record at a time, but you cannot predict the somewhat arbitrary ways they will want to screen and roll up the details. If only summarized data is available, then you have already made assumptions about data usage patterns that will cause users to run into a brick wall when they want to dig deeper into the details. Of course, atomic details can be complemented by summary dimensional models that provide performance advantages for common queries of aggregated data, but business users cannot live on summary data alone; they need the gory details to answer their ever-changing questions.

**Rule #2: Structure dimensional models around business processes.**

Business processes are the activities performed by your organization; they represent measurement events, like taking an order or billing a customer. Business processes typically capture or generate unique performance metrics associated with each event. These metrics translate into facts, with each business process represented by a single atomic fact table. In addition to single process fact tables, consolidated fact tables are sometimes created that combine metrics from multiple processes into one fact table at a common level of detail. Again, consolidated fact tables are a complement to the detailed single-process fact tables, not a substitute for them.

**Rule #3: Ensure that every fact table has an associated date dimension table.**

The measurement events described in Rule #2 always have a date stamp of some variety associated with them, whether it’s a monthly balance snapshot or a monetary transfer captured to the hundredth of a second. Every fact table should have at least one foreign key to an associated date dimension table, whose grain is a single day, with calendar attributes and nonstandard characteristics about the measurement event date, such as the fiscal month and corporate holiday indicator. Sometimes multiple date foreign keys are represented in a fact table.

**Rule #4: Ensure that all facts in a single fact table are at the same grain or level of detail.**

There are three fundamental grains to categorize all fact tables: transactional, periodic snapshot, or accumulating snapshot. Regardless of its grain type, every measurement within a fact table must be at the exact same level of detail. When you mix facts representing multiple levels of granularity in the same fact table, you are setting yourself up for business user confusion and making the BI applications vulnerable to overstated or otherwise erroneous results.

**Rule #5: Resolve many-to-many relationships in fact tables.**

Since a fact table stores the results of a business process event, there’s inherently a many-to-many (M:M) relationship between its foreign keys, such as multiple products being sold in multiple stores on multiple days. These foreign key fields should never be null. Sometimes dimensions can take on multiple values for a single measurement event, such as the multiple diagnoses associated with a health care encounter or multiple customers with a bank account. In these cases, it’s unreasonable to resolve the many-valued dimensions directly in the fact table, as this would violate the natural grain of the measurement event. Thus, we use a many-to-many, dual-keyed bridge table in conjunction with the fact table.

**Rule #6: Resolve many-to-one relationships in dimension tables.**

Hierarchical, fixed-depth many-to-one (M:1) relationships between attributes are typically denormalized or collapsed into a flattened dimension table. If you’ve spent most of your career designing entity-relationship models for transaction processing systems, you’ll need to resist your instinctive tendency to normalize or snowflake a M:1 relationship into smaller subdimensions; dimension denormalization is the name of the game in dimensional modeling.

It is relatively common to have multiple M:1 relationships represented in a single dimension table. One-to-one relationships, like a unique product description associated with a product code, are also handled in a dimension table. Occasionally many-to-one relationships are resolved in the fact table, such as the case when the detailed dimension table has millions of rows and its roll-up attributes are frequently changing. However, using the fact table to resolve M:1 relationships should be done sparingly.

**Rule #7: Store report labels and filter domain values in dimension tables.**

The codes and, more importantly, associated decodes and descriptors used for labeling and query filtering should be captured in dimension tables. Avoid storing cryptic code fields or bulky descriptive fields in the fact table itself; likewise, don’t just store the code in the dimension table and assume that users don’t need descriptive decodes or that they’ll be handled in the BI application. If it’s a row/column label or pull-down menu filter, then it should be handled as a dimension attribute.

Though we stated in Rule #5 that fact table foreign keys should never be null, it’s also advisable to avoid nulls in the dimension tables’ attribute fields by replacing the null value with “NA” (not applicable) or another default value, determined by the data steward, to reduce user confusion if possible.

**Rule #8: Make certain that dimension tables use a surrogate key.**

Meaningless, sequentially assigned surrogate keys (except for the date dimension, where chronologically assigned and even more meaningful keys are acceptable) deliver a number of operational benefits, including smaller keys which mean smaller fact tables, smaller indexes, and improved performance. Surrogate keys are absolutely required if you’re tracking dimension attribute changes with a new dimension record for each profile change. Even if your business users don’t initially visualize the value of tracking attribute changes, using surrogates will make a downstream policy change less onerous. The surrogates also allow you to map multiple operational keys to a common profile, plus buffer you from unexpected operational activities, like the recycling of an obsolete product number or acquisition of another company with its own coding schemes.

**Rule #9: Create conformed dimensions to integrate data across the enterprise.**

Conformed dimensions (otherwise known as common, master, standard or reference dimensions) are essential for enterprise data warehousing. Managed once in the ETL system and then reused across multiple fact tables, conformed dimensions deliver consistent descriptive attributes across dimensional models and support the ability to drill across and integrate data from multiple business processes. The Enterprise Data Warehouse Bus Matrix is the key architecture blueprint for representing the organization’s core business processes and associated dimensionality. Reusing conformed dimensions ultimately shortens the time-to-market by eliminating redundant design and development efforts; however, conformed dimensions require a commitment and investment in data stewardship and governance, even if you don’t need everyone to agree on every dimension attribute to leverage conformity.

**Rule #10: Continuously balance requirements and realities to deliver a DW/BI solution that’s accepted by business users and that supports their decision-making.**

Dimensional modelers must constantly straddle business user requirements along with the underlying realities of the associated source data to deliver a design that can be implemented and that, more importantly, stands a reasonable chance of business adoption. The requirements-versus-realities balancing act is a fact of life for DW/BI practitioners, whether you’re focused on the dimensional model, project strategy, technical/ETL/BI architectures or deployment/maintenance plan.

If you’ve read our Intelligent Enterprise articles, [Toolkit books](http://www.kimballgroup.com/data-warehouse-and-business-intelligence-resources/data-warehouse-books/) or Design Tips regularly, these rules shouldn’t be news to you, but here we’ve consolidated our rules into a single rulebook that you can refer to when you are gathered to design or review your models.