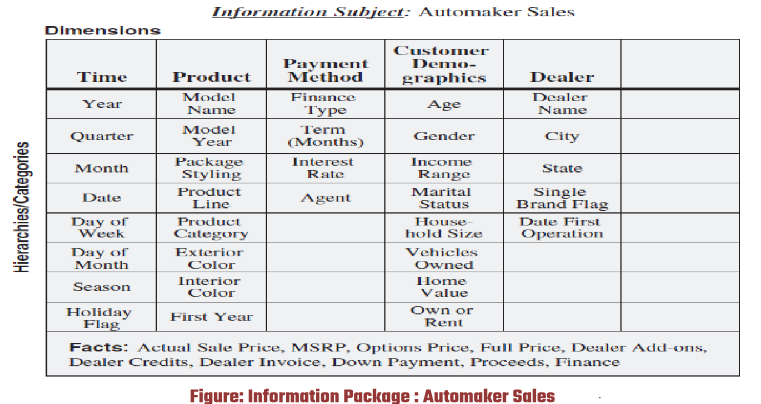
**Information Package:**

To determine and document information requirements for a DW. Useful because traditional methods may not fully capture evolving requirements.

*Information Packages focus on specific subject areas and enable:*

1. Defining common subject areas.
2. Designing key business metrics.
3. Deciding presentation of data.
4. Determining aggregation or roll-up methods.
5. Setting data quantity for analysis.
6. Defining data access methods.
7. Establishing data granularity.
8. Estimating data warehouse size.
9. Setting data refresh frequency.

**Dimensional modeling** organizes data based on business dimensions and metrics for easy analysis. It's built on a diagram called the multidimensional information package, which outlines how data should be structured.

1. *Design Decisions:*

Process Selection: Choosing subjects from information packages.

Grain Level Selection: Deciding on the level of detail of data.

Dimension Identification and Conformation: Selecting and aligning imp business dimensions.

Fact Selection: Choosing metrics or units of measurement.

Database Duration: Determining the time span of historical data required.

1. *Data Entities in Information Packages:*

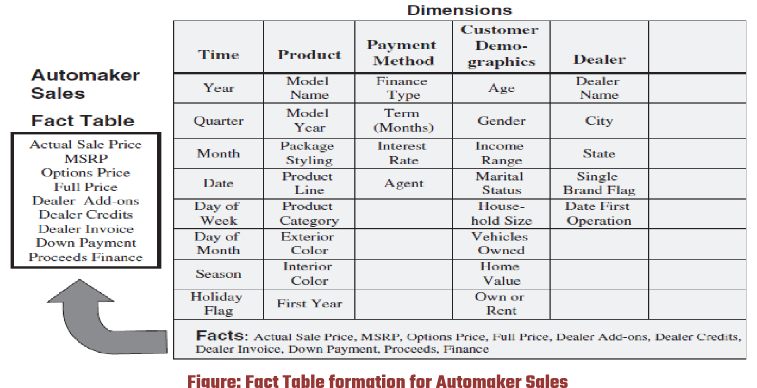
Measurements or Metrics: The numbers you're interested in analyzing.

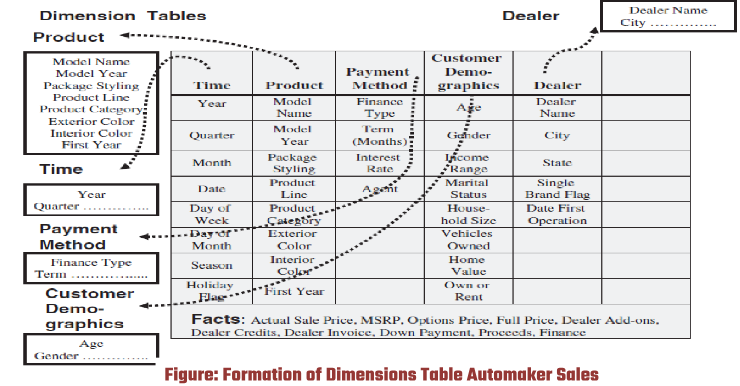
Business Dimensions: Categories that help categorize the data.

Attributes: Details about each category.

**Fact Table:** Contains metrics or facts from the information package diagram.

**Dimensional Table:** Combines individual columns into one data structure, representing attributes for each corresponding dimension table.



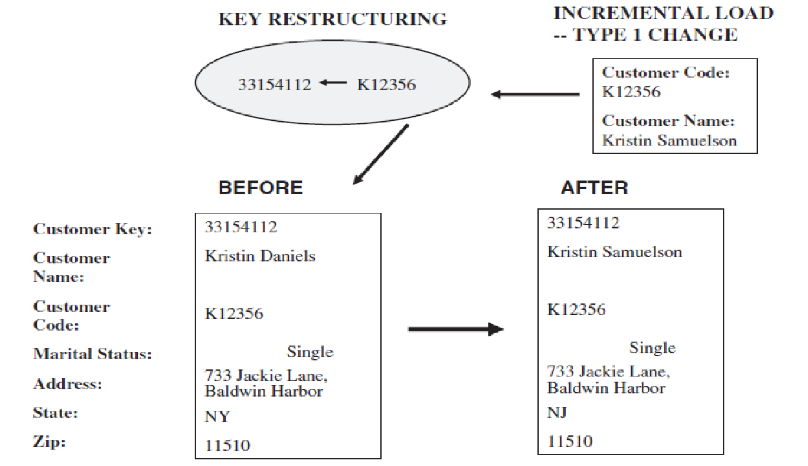


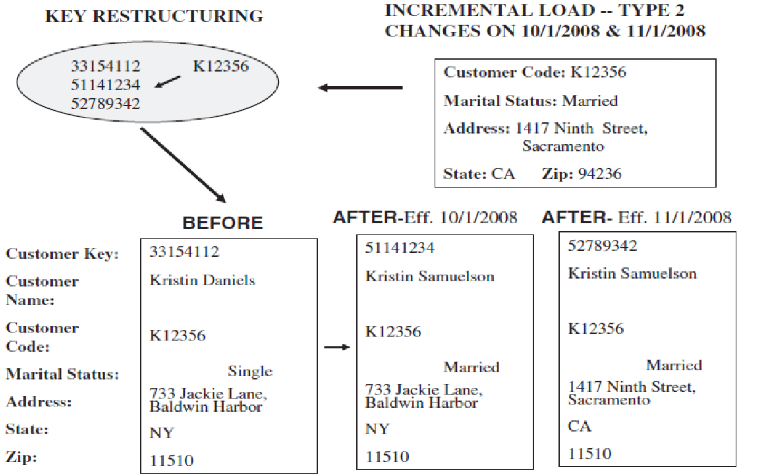
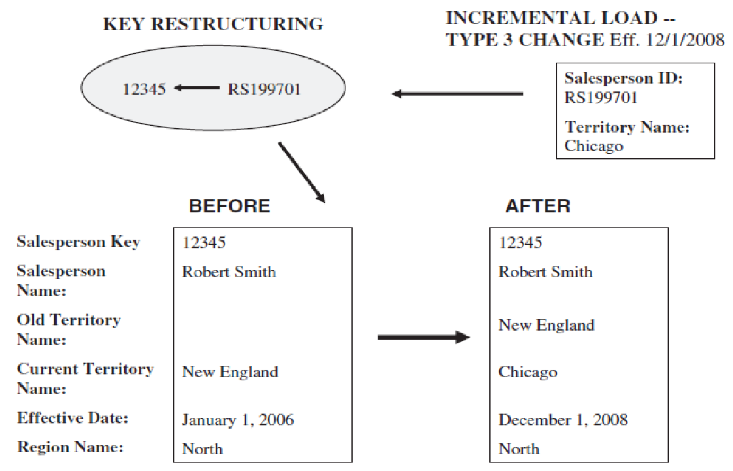
**Arranging Fact and Dimensional Tables:**

Optimization Criteria:

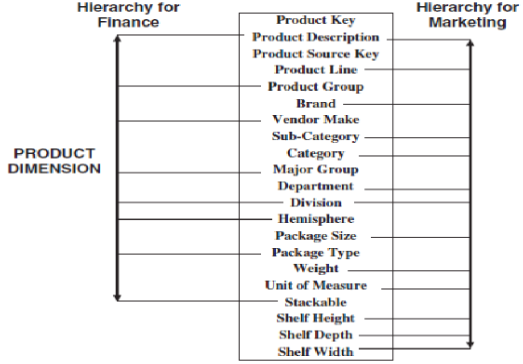
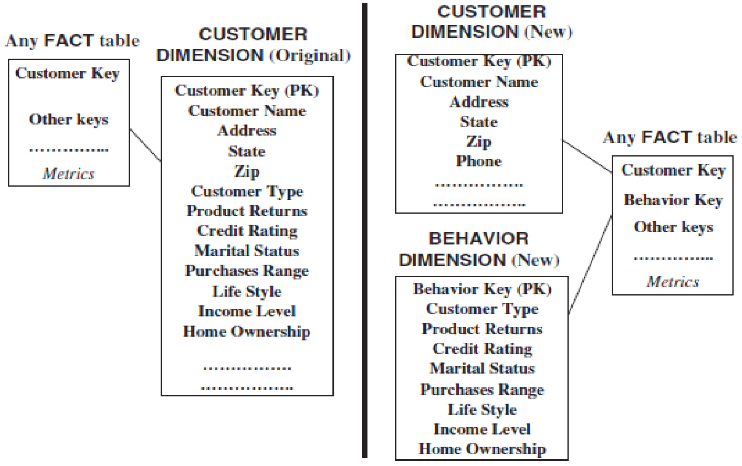
1. Best data access.
2. Query-centric structure.
3. Optimization for queries and analyses.
4. Interaction between dimension and fact tables.
5. Equitable interaction of dimensions with the fact table.
6. Support for drilling down or rolling up along dimension hierarchies.

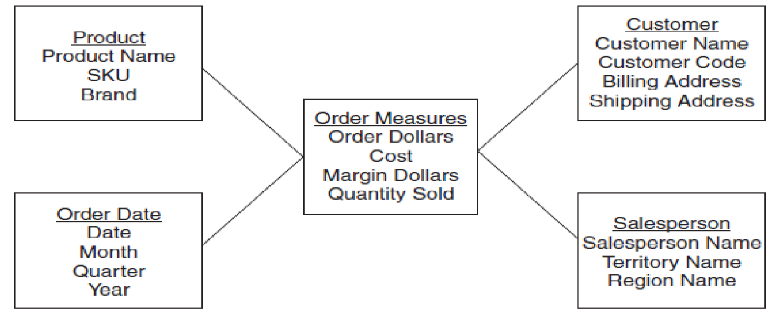
**Updates to Dimension Tables:**

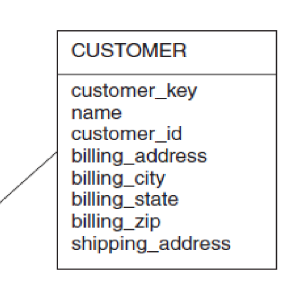
In a data warehouse, fact tables store transactional data and keep growing as new transactions happen. On the other hand, dimension tables hold descriptive details about the data and are more stable but still get updated occasionally, like adding new attributes or updating existing ones. Slowly Changing Dimensions (SCDs) are dimensions that don't change much over time. However, they still undergo changes, which are sorted into different types to help manage and analyze historical data effectively.

1. *Type 1 changes* involve correcting errors or discrepancies in the source systems, such as fixing typos or inaccuracies in attribute values. When a Type 1 change occurs, the existing attribute value in the dimension table is directly overwritten with the corrected value. This approach is straightforward and suitable for minor corrections that do not require preserving historical data.
2. *Type 2 changes* are implemented to preserve the historical context (keep track) of important changes in the source systems. When a change occurs, a new version of the affected dimension record is created, along with a date showing when the change happened. Original record stays as it is, ensuring that historical data is preserved for analysis & to maintain integrity over time.
3. *Type 3 changes* are used to track both old and new attribute values for a temporary period, typically for experimental changes. Unlike Type 2 changes, which preserve a complete history of changes, Type 3 changes focus on capturing transitional data states. Queries need adjustments to smoothly switch between old and new values, allowing analysis across various stages of change.

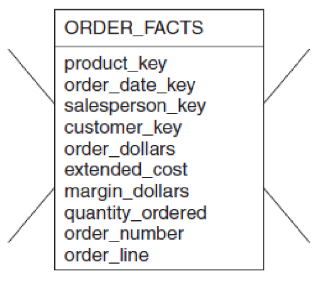
**Miscellaneous Dimensions:**

1. *Large dimensions*, often characterized by an extensive number of rows and columns, present unique challenges in DW. Dealing with dimensions containing millions of rows and numerous attributes like customer and product dimensions requires special considerations. These dimensions may feature multiple hierarchies, adding complexity to their structure. Challenges like slow data population, browsing performance issues, and the need for effective indexing and optimization techniques to ensure efficient data warehouse operations.
2. *Rapidly changing dimensions*, undergo frequent alterations over time. The Type 2 approach, which creates new rows for each attribute change, can lead to an explosion of rows in dimension tables for dimensions that change frequently. This may work well for dimensions with infrequent changes, it can result in performance issues for rapidly changing dimensions. Strategies to address this include breaking large dimension tables → simpler ones to manage frequent changes effectively while maintaining query performance & data integrity.
3. *Junk dimensions* encompass miscellaneous flags and textual fields from legacy systems that may not fit neatly into major dimensions. Managing these miscellaneous fields involves making decisions about their inclusion and relevance in DW. Options for handling junk dimensions include excluding irrelevant fields, incorporating them into the fact table, creating separate dimension tables for each field (though this may increase complexity), or consolidating useful fields into a single "junk" dimension. Each approach has implications for query performance, data management, and system complexity, requiring careful consideration based on the specific requirements of the data warehouse environment.

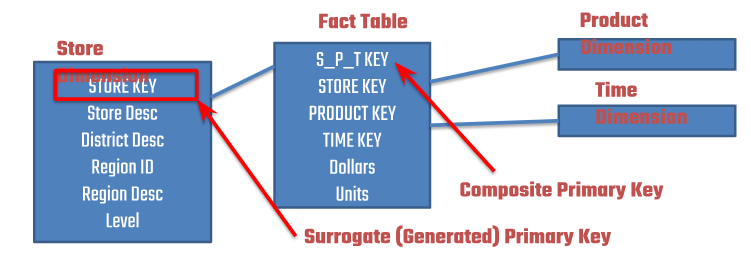
**Star Schema** is a widely used design in DW, consisting of a fact table surrounded by dimension tables. Drill Down takes place.

*Dimension Table Keys:*

Primary keys are crucial in dimension tables because they make sure each row can be identified uniquely. Selecting the right primary key is important for keeping data accurate and making queries fast. Although operational system keys might seem like good options, they often have specific meanings or aren't universal enough for data warehousing. That's why surrogate keys—numbers generated by the system—are often used instead. These surrogate keys help identify dimension table rows reliably without adding any extra meaning.

*Foreign Keys in Fact Tables:*

To establish relationships between the fact table and dimension tables, foreign keys are utilized in the former. Since each dimension table corresponds to multiple records in the fact table, foreign keys play a crucial role in linking these tables together. By referencing the primary keys of dimension tables, foreign keys facilitate seamless integration of dimensional attributes with the measures or facts stored in fact table.

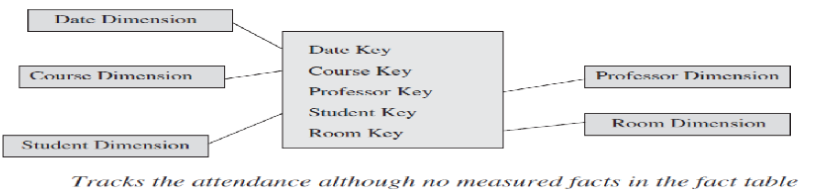


*Primary Keys in Fact Tables:* They represent the intersections of various dimensions, defining unique combinations of dimensional attributes. Depending on the design choices, the fact table may employ different types of primary keys, each with its implications for table structure and query performance.

*Concatenated Primary Keys:* To form a composite primary key for the fact table. This concatenated primary key represents the unique combination of dimensional attributes associated with each record in the fact table. While this approach simplifies key management, it may result in larger key sizes and potential performance implications.

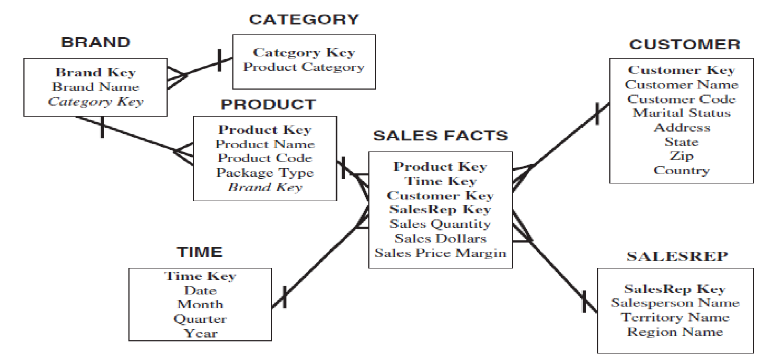
*Compound Primary Keys:* Alternatively, the fact table may utilize a compound primary key comprising multiple segments, each corresponding to the primary key of a dimension table. This approach allows for more granular control over key composition and facilitates efficient querying by minimizing key size. However, it requires additional management of foreign keys in the fact table to maintain referential integrity with dimension tables.

*Generated Primary Keys:* This approach simplifies key management and reduces the complexity of querying but may lead to increased storage overhead due to the inclusion of additional key attributes. Additionally, maintaining referential integrity between the fact table and dimension tables requires careful management of foreign key relationships.

*Factless fact tables* are a unique type of fact table in data warehousing that store data about events or activities where no measurable quantities are recorded. Unlike typical fact tables that contain numerical measures, factless fact tables capture the occurrence of certain events or relationships between dimensions without quantifying them.

**Advantages of Star Schema:**

1. User-Friendly: Star schema's intuitive structure simplifies user interaction, enabling direct query formulation without relying on complex interfaces.
2. *Reflects User Perspective:* Aligned with users' mental models, star schema facilitates data retrieval based on metrics and business dimensions, enhancing usability.
3. *Optimized Navigation:* Its straightforward join paths streamline navigation, ensuring faster data access and smoother user experience.
4. *Query Processing Efficiency:* Star schema's simplicity promotes efficient query execution, enabling swift retrieval of relevant data for analysis.
5. *Drill Down and Roll Up:* Supporting flexible data exploration, star schema allows users to delve into details or aggregate data for comprehensive analysis.
6. *STARjoin and STARindex:* Specialized operations like STARjoin (high-speed, parallelizable, multi-table join operation) and STARindex (specialized index created on foreign keys of the fact table) optimize query performance by facilitating high-speed, parallelizable joins and efficient index usage.

**Snowflake Schema**, achieved through the process of "snowflaking," involves normalizing dimension tables within a STAR schema. This normalization results in a structure resembling a snowflake due to the branching out of dimension tables.

**Advantages:**

1. *Reduced Redundancy:* Normalization minimizes data redundancy by removing repetitive attributes, leading to more efficient storage utilization.
2. *Improved Data Integrity:* By reducing redundancy, snowflake schema enhances data integrity, ensuring consistency across the database.
3. *Simplified Maintenance:* Updates and modifications are easier to manage, simplifying maintenance tasks and reducing the risk of errors.
4. *Flexible Scalability:* Snowflake schema allows for flexible scaling as additional data & dimensions can be incorporated without significantly altering the schema structure.
5. *Optimized Query Performance:* Although joins may be more complex in snowflake schema compared to star schema, query performance can be optimized through proper indexing and query optimization techniques.
6. *Support for Complex Relationships:* Snowflake schema accommodates complex relationships between data entities, facilitating more nuanced data analysis and reporting.