**Dimensional Modeling**

**1. Declare the Grain**

**Definition**: Declaring the grain means **defining exactly what a single row in the fact table represents**.

**Key Considerations**:

* Grain drives fact table design.
* Examples:
  + "One row per product sale per transaction"
  + "One row per daily snapshot of inventory"

**Benefits**:

* Prevents confusion during ETL and reporting.
* Establishes the level of detail for all facts and dimensions.
* Determines the type of fact table: transactional, periodic snapshot, accumulating snapshot.

**2. Star vs. Snowflake vs. Galaxy Schema**

| **Feature** | **Star Schema** | **Snowflake Schema** | **Galaxy Schema** |
| --- | --- | --- | --- |
| Structure | Central fact, denormalized dims | Central fact, normalized dims | Multiple fact tables, shared dimensions |
| Joins | Fewer, faster | More, slower | Moderate |
| Storage | Slightly larger | Smaller | Depends on model |
| Complexity | Simple | More complex | Most complex |
| Use Case | Reporting, dashboarding | OLAP cubes, normalized warehouses | Enterprise DW with multiple processes |

Explanation:

* Star Schema: Facts in center, dimensions flat and denormalized.
* Snowflake Schema: Dimensions are normalized into sub-dimensions.
* Galaxy Schema: Also called Fact Constellation; multiple fact tables share dimensions.

**3. Degenerate and Junk Dimensions**

**Degenerate Dimensions (DD):**

* Exist in **fact tables** but not in any dimension table.
* Typically come from **transaction identifiers**.
* Example: Invoice number, Order number.

**Junk Dimensions:**

* Combine **miscellaneous flags and indicators** into a single dimension.
* Avoids cluttering the fact table.
* Example:
  + Payment method (Cash, Card, UPI)
  + Shipping flag (Expedited Y/N)
  + Return status (Yes/No)

**4. Slowly Changing Dimensions (SCD) Types 0–6**

| **Type** | **Name** | **Description** |
| --- | --- | --- |
| 0 | Fixed Dimension | Never changes. No updates. |
| 1 | Overwrite | Update the record in-place. No history kept. |
| 2 | Historical Rows | Insert new row with versioning. History retained. |
| 3 | Previous Value | One column tracks old value (e.g., previous city) |
| 4 | History Table | Historical data is in separate table. |
| 5 | Hybrid 1+2 | Type 1 in dimension, Type 2 keys in fact |
| 6 | Hybrid 1+2+3 | Merge of Types 1, 2, and 3 (aka "Type 6") |

Notes:

* Type 1 is used when corrections are made (e.g., name typo).
* Type 2 is used when history must be preserved (e.g., customer moves city).
* Type 6 supports reporting both current and historical data efficiently.

**5. Surrogate vs. Natural Keys**

| **Key Type** | **Description** | **Use Case** |
| --- | --- | --- |
| Natural Key | Real-world business key (e.g., SSN, email, code) | Avoid if changes happen to real key |
| Surrogate Key | Synthetic DB-generated key (e.g., INT identity) | Preferred in DW for tracking changes |

Benefits of Surrogate Keys:

* Allow support for SCD Type 2 versioning.
* Decouple DW from source systems' key changes.
* Enable consistent joins and indexing.

**6. Bus Matrix Workshop**

**Definition**: A bus matrix defines business processes (facts) and associated conformed dimensions across the enterprise.

| **Process / Dimension** | **Date** | **Product** | **Customer** | **Store** | **Salesperson** |
| --- | --- | --- | --- | --- | --- |
| Sales | Yes | Yes | Yes | Yes | Yes |
| Inventory | Yes | Yes |  | Yes |  |
| Delivery | Yes | Yes | Yes | Yes |  |

Steps:

1. Identify business processes.
2. List relevant dimensions.
3. Mark where they intersect.
4. Identify conformed dimensions to reuse across fact tables.
5. Use as a blueprint for designing star schemas.

**7. Surrogate Key Generator Script**

**SQL (PostgreSQL)**

sql

CREATE SEQUENCE dim\_customer\_sk\_seq START WITH 1 INCREMENT BY 1;

INSERT INTO dim\_customer (customer\_sk, customer\_name, ...)

VALUES (nextval('dim\_customer\_sk\_seq'), 'John Doe', ...);

**T-SQL (SQL Server)**

sql

ALTER TABLE dim\_customer ADD customer\_sk INT IDENTITY(1,1) PRIMARY KEY;

**PySpark**

python

from pyspark.sql.functions import monotonically\_increasing\_id

dim\_df = source\_df.withColumn("customer\_sk", monotonically\_increasing\_id())

**Summary of Best Practices**

* Always **declare the grain** first—it shapes your fact model.
* **Star schema** is preferred for simplicity and performance.
* Use **SCD Type 2** to maintain history. Type 6 for hybrid needs.
* Implement **surrogate keys** for all dimensions.
* Build a **bus matrix** to unify your enterprise data warehouse design.