**Lab Document: Dimensional Modeling Core Concepts**

**Lab 1: Declaring Grain in a Fact Table**

**Objective:**

Understand and define the level of granularity for a fact table.

**Steps:**

1. Open any SQL-based DWH (e.g., PostgreSQL, SQL Server, or Azure Synapse).
2. Create a Sales\_Fact table that records sales transactions.
3. Decide and declare the grain explicitly. For example:
   * One row per product per customer per day.
4. SQL Template:

sql

-- Declare the grain

-- 1 row = 1 product per customer per transaction

CREATE TABLE Sales\_Fact (

SalesID INT PRIMARY KEY,

CustomerID INT,

ProductID INT,

DateKey DATE,

Quantity INT,

TotalAmount DECIMAL(10, 2)

);

1. Document your grain assumption in comments.

**Lab 2: Star vs Snowflake vs Galaxy Schema**

**Objective:**

Create schemas to differentiate star, snowflake, and galaxy structures.

**Steps:**

1. **Star Schema:**
   * Create denormalized dimension tables.
   * Create a fact table with foreign keys to dimensions.

sql

CREATE TABLE DimProduct (

ProductID INT PRIMARY KEY,

Name VARCHAR(100),

Category VARCHAR(50)

);

CREATE TABLE FactSales (

SalesID INT PRIMARY KEY,

ProductID INT,

Quantity INT,

FOREIGN KEY (ProductID) REFERENCES DimProduct(ProductID)

);

1. **Snowflake Schema:**
   * Normalize dimensions by creating sub-dimensions (e.g., category as separate table).

sql

CREATE TABLE DimCategory (

CategoryID INT PRIMARY KEY,

CategoryName VARCHAR(50)

);

CREATE TABLE DimProduct (

ProductID INT PRIMARY KEY,

Name VARCHAR(100),

CategoryID INT,

FOREIGN KEY (CategoryID) REFERENCES DimCategory(CategoryID)

);

1. **Galaxy Schema (Fact Constellation):**
   * Create multiple fact tables (Sales, Returns) sharing conformed dimensions.

sql

CREATE TABLE FactReturns (

ReturnID INT PRIMARY KEY,

ProductID INT,

ReturnDate DATE,

Reason VARCHAR(100)

);

**Lab 3: Degenerate & Junk Dimensions**

**Objective:**

Implement degenerate and junk dimensions.

**Steps:**

1. **Degenerate Dimension:**
   * Store Order Number directly in the fact table without a corresponding dimension table.

sql

CREATE TABLE Sales\_Fact (

OrderID VARCHAR(20), -- Degenerate

...

);

1. **Junk Dimension:**
   * Combine low-cardinality attributes (e.g., PromoFlag, PaymentMethod).

sql

CREATE TABLE DimJunk (

JunkID INT PRIMARY KEY,

PromoFlag VARCHAR(10),

PaymentMethod VARCHAR(20)

);

1. Insert sample combinations (e.g., 'Y', 'Credit Card').

**Lab 4: Slowly Changing Dimensions (SCD Types 0 to 6)**

**Objective:**

Implement and observe changes in different SCD types.

**Steps:**

Create DimCustomer table with fields for SCD types.

1. **Type 1:** Overwrite the data

sql

UPDATE DimCustomer SET Address = 'New Address' WHERE CustomerID = 1;

1. **Type 2:** Add a new row with versioning

sql

INSERT INTO DimCustomer (CustomerID, Name, Address, StartDate, EndDate, IsCurrent)

VALUES (1, 'John Doe', 'New Address', '2024-01-01', NULL, 1);

1. **Type 3:** Maintain current and previous value

sql

ALTER TABLE DimCustomer ADD PreviousAddress VARCHAR(100);

UPDATE DimCustomer SET PreviousAddress = Address, Address = 'New Address' WHERE CustomerID = 1;

1. Practice Type 4 (history table), Type 6 (combo of 1+2+3).

**Lab 5: Surrogate vs Natural Key Demo**

**Objective:**

Demonstrate benefits of surrogate keys.

**Steps:**

1. Create DimEmployee using natural key:

sql

CREATE TABLE DimEmployee (

EmpCode VARCHAR(10) PRIMARY KEY,

Name VARCHAR(100),

Department VARCHAR(50)

);

1. Create another version using surrogate key:

sql

CREATE TABLE DimEmployee\_Surrogate (

EmpID INT PRIMARY KEY IDENTITY(1,1),

EmpCode VARCHAR(10),

Name VARCHAR(100),

Department VARCHAR(50)

);

1. Demonstrate how surrogate keys help with history tracking and updates.

**Lab 6: Bus Matrix Design Workshop**

**Objective:**

Design a bus matrix mapping business processes and dimensions.

**Steps:**

1. Identify business processes: e.g., Sales, Returns, Shipments.
2. List conformed dimensions: Date, Customer, Product, Region.
3. Create a matrix:

| **Business Process** | **Date** | **Customer** | **Product** | **Region** |
| --- | --- | --- | --- | --- |
| Sales | Yes | Yes | Yes | Yes |
| Returns | Yes | Yes | Yes | No |
| Shipments | Yes | Yes | No | Yes |

1. Use this matrix to guide schema creation.

**Lab 7: Surrogate Key Generator Script**

**Objective:**

Implement a reusable surrogate key generator in Python.

python

import hashlib

def generate\_surrogate\_key(\*args):

key\_string = '|'.join(map(str, args))

return int(hashlib.md5(key\_string.encode()).hexdigest()[:8], 16)

# Example

print(generate\_surrogate\_key("Customer", "Name", "2023-05-01"))

Alternatively, in SQL Server:

sql

-- SQL Server example using HASHBYTES

SELECT CONVERT(BIGINT, CONVERT(VARBINARY, HASHBYTES('MD5', 'Customer|Name|2023-05-01'))) AS SurrogateKey