Author: Madhurima Rawat

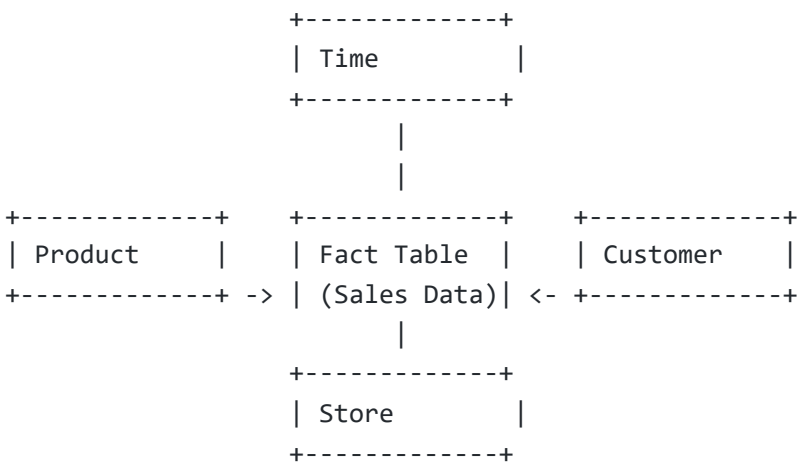
Creating a Star Schema in Data Warehouse

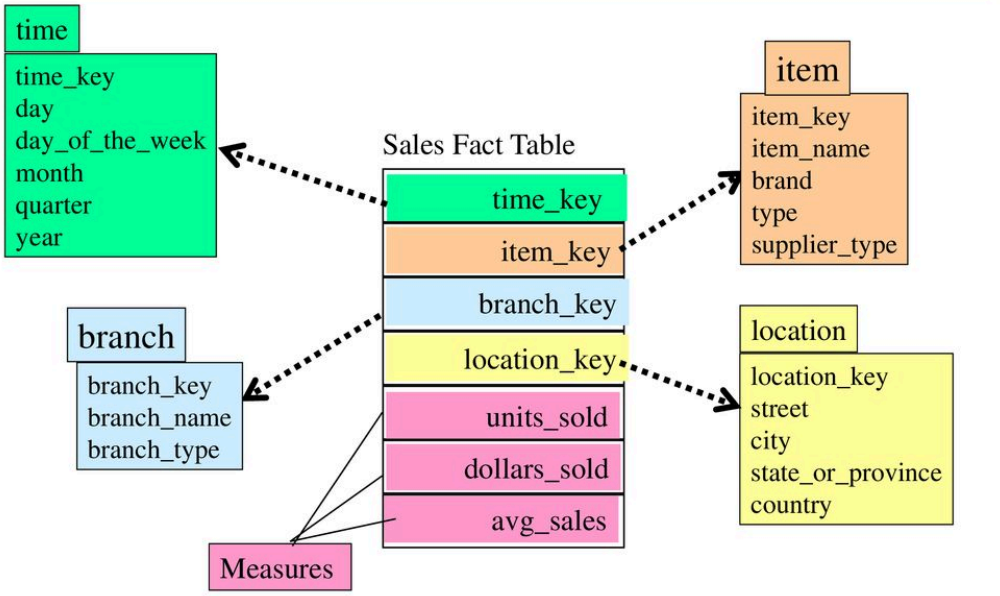
This experiment involves designing and implementing a star schema for a retail data warehouse. The schema includes a central fact table (sales_fact) and multiple dimension tables (product_dim, customer_dim, store_dim, time_dim) to store descriptive attributes. It enables efficient data querying for business analytics, such as analyzing sales trends by product, customer, store, or time.

What is a Star Schema?

A star schema is a data modeling technique commonly used in data warehousing. It consists of:

Star Schema for Data Modeling





1. **Fact Table**: The central table that contains quantitative data (measures) for analysis. It is typically large and has foreign keys linking to dimension tables.
2. **Dimension Tables**: Smaller tables that store descriptive attributes (context) about the measures in the fact table. These provide insights like "who," "what," "where," and "when."

The star schema is called so because its structure resembles a star, with the fact table at the center and the dimension tables radiating outwards.

Star Schema Diagram

Imagine a star schema for a retail business that tracks sales data. The schema contains:

1. A **Fact Table**: `sales_fact`
2. Four **Dimension Tables**:
   - `product_dim`
   - `customer_dim`
   - `store_dim`
   - `time_dim`

```
          product_dim        customer_dim
               |                  |
               |                  |
store_dim ---- sales_fact ---- time_dim
```

MySQL Implementation

Create Database

```
CREATE DATABASE RetailDataWarehouse;
USE RetailDataWarehouse;
```

Create Dimension Tables

1. Product Dimension Table

```
CREATE TABLE product_dim (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10, 2)
);
```

- `product_id` : Unique identifier for each product.
- `product_name` : Name of the product.
- `category` : Product category (e.g., electronics, groceries).
- `price` : Price of the product.

2. Customer Dimension Table

```
CREATE TABLE customer_dim (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100) NOT NULL,
    gender ENUM('Male', 'Female', 'Other'),
    email VARCHAR(100),
    city VARCHAR(50)
);
```

- `customer_id` : Unique identifier for each customer.
- `customer_name` : Name of the customer.
- `gender` : Gender of the customer.
- `email` : Email address for communication.
- `city` : City where the customer resides.

3. Store Dimension Table

```
CREATE TABLE store_dim (
    store_id INT PRIMARY KEY AUTO_INCREMENT,
    store_name VARCHAR(100),
```

```
    location VARCHAR(100),
    manager_name VARCHAR(100)
);
```

- `store_id` : Unique identifier for each store.
- `store_name` : Name of the store.
- `location` : Physical address or city.
- `manager_name` : Name of the store manager.

4. Time Dimension Table

```
CREATE TABLE time_dim (
    time_id INT PRIMARY KEY AUTO_INCREMENT,
    date DATE NOT NULL,
    day_of_week VARCHAR(10),
    month VARCHAR(10),
    year INT,
    quarter VARCHAR(10)
);
```

- `time_id` : Unique identifier for each time entry.
- `date` : Specific date.
- `day_of_week` : Name of the day.
- `month` : Month name.
- `year` : Year of the date.
- `quarter` : Quarter of the year.

Create Fact Table

5. Sales Fact Table

```
CREATE TABLE sales_fact (
    sales_id INT PRIMARY KEY AUTO_INCREMENT,
    product_id INT NOT NULL,
    customer_id INT NOT NULL,
    store_id INT NOT NULL,
    time_id INT NOT NULL,
    quantity INT,
    total_sales DECIMAL(15, 2),
    FOREIGN KEY (product_id) REFERENCES product_dim(product_id),
    FOREIGN KEY (customer_id) REFERENCES customer_dim(customer_id),
    FOREIGN KEY (store_id) REFERENCES store_dim(store_id),
    FOREIGN KEY (time_id) REFERENCES time_dim(time_id)
);
```

- `sales_id` : Unique identifier for each sale.
- `product_id` : Links to the `product_dim` table.
- `customer_id` : Links to the `customer_dim` table.
- `store_id` : Links to the `store_dim` table.
- `time_id` : Links to the `time_dim` table.
- `quantity` : Number of items sold.
- `total_sales` : Total revenue generated from the sale.

Explanation of the Schema

1. **Fact Table (`sales_fact`)**: Stores measurable data like quantity sold and total sales. It connects to all dimension tables using foreign keys.
2. **Dimension Tables**:
   - Provide descriptive details.
   - Help in slicing and dicing the data for analysis (e.g., total sales by product, by customer, by store, or by time).

Key Benefits

- Simplified queries due to its denormalized structure.
- Efficient for read-heavy operations like reporting and analytics.

Inserting Sample Data

Insert Data into Dimension Tables

```
-- Insert into product_dim
INSERT INTO product_dim (product_name, category, price)
VALUES
('Laptop', 'Electronics', 800.00),
('Chair', 'Furniture', 120.00),
('Book', 'Stationery', 15.00);

-- Insert into customer_dim
INSERT INTO customer_dim (customer_name, gender, email, city)
VALUES
('John Doe', 'Male', 'john@example.com', 'New York'),
('Jane Smith', 'Female', 'jane@example.com', 'San Francisco'),
('Ali Khan', 'Male', 'ali@example.com', 'Chicago');

-- Insert into store_dim
INSERT INTO store_dim (store_name, location, manager_name)
```

```
VALUES
('TechStore', 'New York', 'Emily White'),
('HomeDecor', 'San Francisco', 'Robert Brown'),
('BookWorld', 'Chicago', 'Sophia Johnson');

-- Insert into time_dim
INSERT INTO time_dim (date, day_of_week, month, year, quarter)
VALUES
('2025-01-20', 'Monday', 'January', 2025, 'Q1'),
('2025-01-21', 'Tuesday', 'January', 2025, 'Q1'),
('2025-01-22', 'Wednesday', 'January', 2025, 'Q1');
```

Insert Data into Fact Table

```
-- Insert into sales_fact
INSERT INTO sales_fact (product_id, customer_id, store_id, time_id, quantity, total_sales)
VALUES
(1, 1, 1, 1, 2, 1600.00),
(2, 2, 2, 2, 4, 480.00),
(3, 3, 3, 3, 10, 150.00);
```

Querying and Displaying Data

Describe Tables

```
-- Describe all tables
DESCRIBE product_dim;
DESCRIBE customer_dim;
DESCRIBE store_dim;
DESCRIBE time_dim;
DESCRIBE sales_fact;
```

Display Data

```
-- Select all data from each table
SELECT * FROM product_dim;
SELECT * FROM customer_dim;
SELECT * FROM store_dim;
SELECT * FROM time_dim;
SELECT * FROM sales_fact;
```

Sample `product_dim` Data

product_id	product_name	category	price
1	Laptop	Electronics	800.00
2	Chair	Furniture	120.00
3	Book	Stationery	15.00

Aggregate and Analyze Data

The provided SQL queries help analyze the total sales data by different dimensions in the retail data warehouse.

1. Total Sales by Product

```
SELECT p.product_name, SUM(s.total_sales) AS total_revenue
FROM sales_fact s
JOIN product_dim p ON s.product_id = p.product_id
GROUP BY p.product_name;
```

- **Explanation**: This query calculates the total revenue for each product by summing up the `total_sales` from the `sales_fact` table, grouped by the `product_name` from the `product_dim` table.
- **Output**:

product_name	total_revenue
Laptop	1600.00
Chair	480.00
Book	150.00

  - **Laptop**: The total revenue from laptops is 1600.00.
  - **Chair**: The total revenue from chairs is 480.00.
  - **Book**: The total revenue from books is 150.00.

2. Sales by Customer

```
SELECT c.customer_name, SUM(s.total_sales) AS total_spent
FROM sales_fact s
JOIN customer_dim c ON s.customer_id = c.customer_id
GROUP BY c.customer_name;
```

- **Explanation**: This query calculates the total amount spent by each customer. It sums the `total_sales` for each customer, based on the `customer_id` in the `sales_fact` table and the `customer_name` in the `customer_dim` table.
- **Output**:

customer_name	total_spent
John Doe	1600.00
Jane Smith	480.00
Ali Khan	150.00

  - **John Doe**: John has spent a total of 1600.00.
  - **Jane Smith**: Jane has spent a total of 480.00.
  - **Ali Khan**: Ali has spent a total of 150.00.

3. Total Sales by Store

```sql
SELECT st.store_name, SUM(s.total_sales) AS total_store_sales
FROM sales_fact s
JOIN store_dim st ON s.store_id = st.store_id
GROUP BY st.store_name;
```

- **Explanation**: This query calculates the total sales for each store. It sums the `total_sales` for each store by joining the `sales_fact` table with the `store_dim` table using the `store_id`.
- **Output**:

store_name	total_store_sales
TechStore	1600.00
HomeDecor	480.00
BookWorld	150.00

  - **TechStore**: TechStore has generated 1600.00 in total sales.
  - **HomeDecor**: HomeDecor has generated 480.00 in total sales.
  - **BookWorld**: BookWorld has generated 150.00 in total sales.

These aggregate queries help summarize the sales performance by different aspects of the business, such as products, customers, and stores. This structured data aggregation is crucial for analyzing trends and making data-driven business decisions. The output data shows how sales are distributed among products, customers, and stores, helping businesses identify high-performing products, top customers, and best-selling stores.

The star schema provides a streamlined and efficient structure for organizing business data, enabling easy and fast queries for analytical purposes. By separating fact and dimension tables, it enhances the clarity of data relationships and optimizes reporting and decision-making processes.