Lab Number: 2

Title
Designing Various Data Warehouse Schemas (Star Schema, Snowflake Schema, Fact Constellation) Using MySQL Query

Objective
To gain practical understanding of different data warehouse schema designs—Star, Snowflake, and Fact Constellation by implementing them using SQL queries.

IDE/Tools Used
MySQL, Apache Server

Query Programming Language
SQL

Theory
A **Data Warehouse** is a centralized repository used for storing large volumes of structured data from multiple sources. It is specifically designed for query and analysis, rather than transaction processing. The primary goal of a data warehouse is to support decision-making processes by enabling data consolidation, historical analysis, and data mining.

A **Schema in data warehousing** defines the logical view of the entire database. It describes how the data is organized and how the relationships between data are maintained. There are three main types of data warehouse schemas:

1. Star Schema
The Star Schema is the simplest and most commonly used data warehouse schema. It consists of a central fact table that stores quantitative data (measures), and multiple dimension tables that contain descriptive attributes related to the facts. The structure resembles a star, with the fact table at the center and the dimension tables radiating outwards.

Advantages
- Simple to understand and design
- Efficient for querying and reporting

Example: A sales fact table connected to dimension tables like Customer, Product, Time, and Location

2. Snowflake Schema

The Snowflake Schema is a more complex version of the star schema where dimension tables are normalized into multiple related tables. This reduces data redundancy but increases the complexity of queries due to more joins.

Advantages

- Reduces data redundancy
- Better data integrity

Disadvantages

- More complex queries due to multiple table joins

Example: The Product dimension may be broken down into Product, Product Category, and Product Supplier tables

3. Fact Constellation (or Galaxy Schema)

A Fact Constellation Schema contains multiple fact tables that share many dimension tables. It is used in complex data warehouses where multiple business processes are modeled.

Advantages

- Enables integrated analysis of different processes
- Reusability of dimension tables

Example: A data warehouse with separate fact tables for Sales and Inventory, both sharing Time, Product, and Location dimensions

By designing these schemas using SQL queries in MySQL and integrating them with tools like Apache server, we gain practical insights into organizing and optimizing data for analytical processing. Understanding these schemas is essential for effective data warehousing and business intelligence solutions.

Implementation

A. Star Schema Queries

```
-- Create a database
CREATE DATABASE autoxyz;

-- Create a source table for data warehouse.
CREATE TABLE company (
  id INT auto_increment PRIMARY KEY,
  item_name VARCHAR(255),
  brand VARCHAR(255),
  sold_by VARCHAR(255),
  category VARCHAR(255),
  day INT,
  month VARCHAR(255),
  quarter VARCHAR(255),
  years INT,
  location_name VARCHAR(255),
  state VARCHAR(255),
  pin_code INT,
  branch_name VARCHAR(255),
  branch_manager VARCHAR(255),
  qty_sold INT,
  amt_sold INT
);

INSERT INTO company (
  item_name, brand, sold_by, category,
  day, month, quarter, years, location_name,
  state, pin_code, branch_name, branch_manager,
  qty_sold, amt_sold
)
VALUES
  (
    "car", "model x", "tesla", "four wheeler",
    13, "june", "q2", 2021, "new baneshwor",
    "bagmati", "123", "baneshwor 1",
    "ashish", 2, 15000
  ),
  (
    "car", "model y", "tesla", "four wheeler",
    15, "october", "q4", 2022, "old baneshwor",
    "bagmati", "123", "baneshwor 3",
    "manoj", 1, 5000
  );
```

```
-- Create time dimension table for data warehouse.
CREATE TABLE timedim (
  t_id INT auto_increment PRIMARY KEY,
  day INT,
  month VARCHAR(255),
  quarter VARCHAR(255),
  years INT
);

INSERT INTO timedim (day, month, quarter, years)
SELECT
  day,
  month,
  quarter,
  years
FROM
  company;

-- Create item dimension table for data warehouse.
CREATE TABLE itemdim (
  i_id INT auto_increment PRIMARY KEY,
  item_name VARCHAR(255),
  brand VARCHAR(255),
  sold_by VARCHAR(255),
  category VARCHAR(255)
);

INSERT INTO itemdim (
  item_name, brand, sold_by, category
)
SELECT
  item_name,
  brand,
  sold_by,
  category
FROM
  company;

-- Create location dimension table for data warehouse.
CREATE TABLE locationdim (
  l_id INT auto_increment PRIMARY KEY,
  location_name VARCHAR(255),
  state VARCHAR(255),
  pin_code INT
);
```

```
INSERT INTO locationdim (location_name, state, pin_code)
SELECT
  location_name,
  state,
  pin_code
FROM
  company;

-- Create branch dimension table for data warehouse.
CREATE TABLE branchdim (
  b_id INT auto_increment PRIMARY KEY,
  branch_name VARCHAR(255),
  branch_manager VARCHAR(255)
);

INSERT INTO branchdim (branch_name, branch_manager)
SELECT
  branch_name,
  branch_manager
FROM
  company;

-- Create sales fact table for data warehouse using Foreign Key.
CREATE TABLE salesfact (
  t_id INT,
  i_id INT,
  l_id INT,
  b_id INT,
  qty_sold INT,
  amt_sold INT,
  FOREIGN KEY(t_id) REFERENCES timedim(t_id),
  FOREIGN KEY(i_id) REFERENCES itemdim(i_id),
  FOREIGN KEY(l_id) REFERENCES locationdim(l_id),
  FOREIGN KEY(b_id) REFERENCES branchdim(b_id)
);

INSERT INTO salesfact (
  t_id, i_id, l_id, b_id, qty_sold, amt_sold
)
SELECT
  t_id,
  i_id,
  l_id,
  b_id,
  qty_sold,
  amt_sold
FROM
  company c
  LEFT OUTER JOIN timedim t ON t.day = c.day
  AND t.month = c.month
  AND t.quarter = c.quarter
  AND t.years = c.years
  LEFT OUTER JOIN itemdim i ON i.item_name = c.item_name
  AND i.brand = c.brand
  AND i.sold_by = c.sold_by
  AND i.category = c.category
  LEFT OUTER JOIN locationdim l ON l.location_name = c.location_name
  AND l.state = c.state
  AND l.pin_code = c.pin_code
  LEFT OUTER JOIN branchdim b ON b.branch_name = c.branch_name
  AND b.branch_manager = c.branch_manager;

-- Query to select the records where years = 2022
SELECT
  *
FROM
  salesfact s
  LEFT OUTER JOIN timedim t ON t.t_id = s.t_id
WHERE
  years = 2022;
```

B. Snowflake Schema Queries

```
-- Create a database named "snowcompany".
CREATE DATABASE snowcompany;

-- Create a source table for data warehouse.
CREATE TABLE company (
  id int AUTO_INCREMENT PRIMARY KEY,
  item_name varchar(255),
  brand varchar(255),
  sold_by varchar(255),
  category varchar(255),
  day int,
  month varchar(255),
  quarter varchar(255),
  years int,
  location_name varchar(255),
  state varchar(255),
  pin_code int,
  branch_name varchar(255),
  branch_manager varchar(255),
  department_name varchar(255),
  department_code int,
  supplier_name varchar(255),
  supplier_address varchar(255),
  supplier_type varchar(255),
  qty_sold int,
  amt_sold int
);

INSERT INTO company(
  item_name, brand, sold_by, category,
  day, month, quarter, years, location_name,
  state, pin_code, branch_name, branch_manager,
  department_name, department_code,
  supplier_name, supplier_address,
  supplier_type, qty_sold, amt_sold
)
VALUES
  (
    "Car", "Model X", "Tesla", "Four wheeler",
    13, "June", "Q2", 2021, "New
Baneshwor",
    "Bagmati", "123", "Baneshwor 1",
    "Ashish", "sales", 013, "C&C
Auto",
    "Koteshwor", "Auto Four Wheeler",
    2, 15000
  ),
  (
    "Car", "Model Y", "Tesla", "Four wheeler",
    15, "October", "Q4", 2022, "Old
Baneshwor",
    "Bagmati", "123", "Baneshwor 3",
    "Manoj", "finance", 420, "T&T
Auto",
    "Tripureshwor", "Auto", 1, 5000
  );

-- Create time dimension table for data warehouse.
CREATE TABLE timedim (
  t_id int AUTO_INCREMENT PRIMARY KEY,
  day int,
  month varchar(255),
  quarter varchar(255),
  years int
);

INSERT INTO timedim(day, month, quarter, years)
SELECT
  day,
  month,
  quarter,
  years
FROM
  company;

-- Create location dimension table for data warehouse.
CREATE TABLE locationdim (
  l_id int AUTO_INCREMENT PRIMARY KEY,
  location_name varchar(255),
  state varchar(255),
  pin_code int
);

INSERT INTO locationdim(location_name, state, pin_code)
SELECT
  location_name,
  state,
  pin_code
FROM
  company;

-- Create department dimension table for data warehouse.
CREATE TABLE departmentdim(
  dept_id int AUTO_INCREMENT PRIMARY KEY,
  dept_name varchar(255),
  dept_code int
);
```

```sql
INSERT INTO departmentdim(dept_name, dept_code)
SELECT
  department_name,
  department_code
FROM
  company;

-- Create branch dimension table for data warehouse.
CREATE TABLE branchdim (
  b_id int AUTO_INCREMENT PRIMARY KEY,
  branch_name varchar(255),
  branch_manager varchar(255),
  depart_id int,
  FOREIGN KEY(depart_id) REFERENCES departmentdim(dept_id)
);

INSERT INTO branchdim(
  branch_name, branch_manager, depart_id
)
SELECT
  branch_name,
  branch_manager,
  dept_id
FROM
  company c
  JOIN departmentdim d ON c.department_name = d.dept_name
  AND c.department_code = d.dept_code;

-- Create Supplier dimension table for data warehouse.
CREATE TABLE supplierdim(
  supp_id int AUTO_INCREMENT PRIMARY KEY,
  supp_name varchar(255),
  supp_address varchar(255),
  supp_type varchar(255)
);

INSERT INTO supplierdim(
  supp_name, supp_address, supp_type
)
SELECT
  supplier_name,
  supplier_address,
  supplier_type
FROM
  company;

-- Create item dimension table for data warehouse.
CREATE TABLE itemdim (
  i_id int AUTO_INCREMENT PRIMARY KEY,
  item_name varchar(255),
  brand varchar(255),
  sold_by varchar(255),
  category varchar(255),
  supplier_id int,
  FOREIGN KEY(supplier_id) REFERENCES supplierdim(supp_id)
);

INSERT INTO itemdim(
  item_name, brand, sold_by, category,
  supplier_id
)
SELECT
  item_name,
  brand,
  sold_by,
  category,
  supp_id
FROM
  company c
  LEFT OUTER JOIN supplierdim s ON c.supplier_name = s.supp_name
  AND c.supplier_address = s.supp_address
  AND c.supplier_type = s.supp_type;



-- Create sales fact table for data warehouse using Foreign Key.
CREATE TABLE salesFact (
  t_id int,
  i_id int,
  l_id int,
  b_id int,
  qty_sold int,
  amt_sold int,
  FOREIGN Key(t_id) REFERENCES timedim(t_id),
  FOREIGN Key(i_id) REFERENCES itemdim(i_id),
  FOREIGN Key(l_id) REFERENCES locationdim(l_id),
  FOREIGN Key(b_id) REFERENCES branchdim(b_id)
);

INSERT INTO salesFact(
  t_id, i_id, l_id, b_id, qty_sold, amt_sold
)
SELECT
  t_id,
  i_id,
  l_id,
  b_id,
  qty_sold,
  amt_sold
FROM
  company c
  LEFT OUTER JOIN timedim t ON t.day = c.day
  AND t.month = c.month
  AND t.quarter = c.quarter
  AND t.years = c.years
  LEFT OUTER JOIN itemdim i ON i.item_name = c.item_name
  AND i.brand = c.brand
  AND i.sold_by = c.sold_by
  AND i.category = c.category
  LEFT OUTER JOIN locationdim l ON l.location_name = c.location_name
  AND l.state = c.state
  AND l.pin_code = c.pin_code
  LEFT OUTER JOIN branchdim b ON b.branch_name = c.branch_name
  AND b.branch_manager = c.branch_manager;
SELECT
  *
FROM
  salesFact;

-- Query to show the records of sales fact table with the department dimension
table
SELECT
  *
FROM
  salesFact s
  LEFT OUTER JOIN branchdim b ON b.b_id = s.b_id
  LEFT OUTER JOIN departmentdim d ON b.depart_id = d.dept_id;
```

C. Galaxy Schema Queries

```
-- Create the database
CREATE DATABASE galaxycompany;
USE galaxycompany;

-- Source Table for Data Warehouse (for initial data load)
CREATE TABLE company (
  id INT AUTO_INCREMENT PRIMARY KEY,
  item_name VARCHAR(255),
  brand VARCHAR(255),
  sold_by VARCHAR(255),
  category VARCHAR(255),
  day INT,
  month VARCHAR(255),
  quarter VARCHAR(255),
  years INT,
  location_name VARCHAR(255),
  state VARCHAR(255),
  pin_code INT,
  branch_name VARCHAR(255),
  branch_manager VARCHAR(255),
  department_name VARCHAR(255),
  department_code INT,
  supplier_name VARCHAR(255),
  supplier_address VARCHAR(255),
  supplier_type VARCHAR(255),
  qty_sold INT,
  amt_sold INT,
  qty_in_stock INT,
  reorder_level INT,
  last_updated DATE
);

INSERT INTO company (
  item_name, brand, sold_by, category,
  day, month, quarter, years, location_name,
  state, pin_code, branch_name, branch_manager,
  department_name, department_code,
  supplier_name, supplier_address,
  supplier_type, qty_sold, amt_sold,
  qty_in_stock, reorder_level, last_updated
)
VALUES
  (
    "Car", "Model X", "Tesla", "Four wheeler",
    13, "June", "Q2", 2021, "New Baneshwor",
    "Bagmati", 123, "Baneshwor 1", "Ashish",
    "Sales", 13, "C&C Auto", "Koteshwor",
    "Auto Four Wheeler", 2, 15000, 10,
    5, "2022-10-15"
  ),
  (
    "Car", "Model Y", "Tesla", "Four wheeler",
    15, "October", "Q4", 2022, "Old Baneshwor",
    "Bagmati", 123, "Baneshwor 3", "Manoj",
    "Finance", 420, "T&T Auto", "Tripureshwor",
    "Auto", 1, 5000, 5, 2, "2022-10-16"
  );

-- Time Dimension
CREATE TABLE timedim (
  t_id INT AUTO_INCREMENT PRIMARY KEY,
  day INT,
  month VARCHAR(255),
  quarter VARCHAR(255),
  years INT
);

INSERT INTO timedim (day, month, quarter, years)
SELECT
  DISTINCT day,
  month,
  quarter,
  years
FROM
  company;

-- Location Dimension
CREATE TABLE locationdim (
  l_id INT AUTO_INCREMENT PRIMARY KEY,
  location_name VARCHAR(255),
  state VARCHAR(255),
  pin_code INT
);

INSERT INTO locationdim (location_name, state, pin_code)
SELECT
  DISTINCT location_name,
  state,
  pin_code
FROM
  company;

-- Department Dimension
CREATE TABLE departmentdim (
  dept_id INT AUTO_INCREMENT PRIMARY KEY,
  dept_name VARCHAR(255),
  dept_code INT
```

```sql
);

INSERT INTO departmentdim (dept_name, dept_code)
SELECT
  DISTINCT department_name,
  department_code
FROM
  company;

-- Branch Dimension
CREATE TABLE branchdim (
  b_id INT AUTO_INCREMENT PRIMARY KEY,
  branch_name VARCHAR(255),
  branch_manager VARCHAR(255),
  depart_id INT,
  FOREIGN KEY (depart_id) REFERENCES departmentdim(dept_id)
);

INSERT INTO branchdim (
  branch_name, branch_manager, depart_id
)
SELECT
  DISTINCT c.branch_name,
  c.branch_manager,
  d.dept_id
FROM
  company c
  JOIN departmentdim d ON c.department_name = d.dept_name
  AND c.department_code = d.dept_code;

-- Supplier Dimension
CREATE TABLE supplierdim (
  supp_id INT AUTO_INCREMENT PRIMARY KEY,
  supp_name VARCHAR(255),
  supp_address VARCHAR(255),
  supp_type VARCHAR(255)
);

INSERT INTO supplierdim (
  supp_name, supp_address, supp_type
)
SELECT
  DISTINCT supplier_name,
  supplier_address,
  supplier_type
FROM
  company;

-- Item Dimension
```

```sql
CREATE TABLE itemdim (
  i_id INT AUTO_INCREMENT PRIMARY KEY,
  item_name VARCHAR(255),
  brand VARCHAR(255),
  sold_by VARCHAR(255),
  category VARCHAR(255),
  supplier_id INT,
  FOREIGN KEY (supplier_id) REFERENCES supplierdim(supp_id)
);

INSERT INTO itemdim (
  item_name, brand, sold_by, category,
  supplier_id
)
SELECT
  DISTINCT c.item_name,
  c.brand,
  c.sold_by,
  c.category,
  s.supp_id
FROM
  company c
  LEFT JOIN supplierdim s ON c.supplier_name = s.supp_name
  AND c.supplier_address = s.supp_address
  AND c.supplier_type = s.supp_type;

-- Sales Fact Table
CREATE TABLE salesfact (
  t_id INT,
  i_id INT,
  l_id INT,
  b_id INT,
  qty_sold INT,
  amt_sold INT,
  FOREIGN KEY (t_id) REFERENCES timedim(t_id),
  FOREIGN KEY (i_id) REFERENCES itemdim(i_id),
  FOREIGN KEY (l_id) REFERENCES locationdim(l_id),
  FOREIGN KEY (b_id) REFERENCES branchdim(b_id)
);

INSERT INTO salesfact (
  t_id, i_id, l_id, b_id, qty_sold, amt_sold
)
SELECT
  t.t_id,
  i.i_id,
  l.l_id,
  b.b_id,
  c.qty_sold,
  c.amt_sold
FROM
  company c
  LEFT JOIN timedim t ON c.day = t.day
  AND c.month = t.month
  AND c.quarter = t.quarter
  AND c.years = t.years
  LEFT JOIN itemdim i ON c.item_name = i.item_name
  AND c.brand = i.brand
  AND c.sold_by = i.sold_by
  AND c.category = i.category
  LEFT JOIN locationdim l ON c.location_name = l.location_name
  AND c.state = l.state
  AND c.pin_code = l.pin_code
  LEFT JOIN branchdim b ON c.branch_name = b.branch_name
  AND c.branch_manager = b.branch_manager;

-- Inventory Fact Table
CREATE TABLE inventoryfact (
  t_id INT,
  i_id INT,
  l_id INT,
  b_id INT,
  qty_in_stock INT,
  reorder_level INT,
  last_updated DATE,
  FOREIGN KEY (t_id) REFERENCES timedim(t_id),
  FOREIGN KEY (i_id) REFERENCES itemdim(i_id),
  FOREIGN KEY (l_id) REFERENCES locationdim(l_id),
  FOREIGN KEY (b_id) REFERENCES branchdim(b_id)
);

INSERT INTO inventoryfact (
  t_id, i_id, l_id, b_id, qty_in_stock,
  reorder_level, last_updated
)
SELECT
  t.t_id,
  i.i_id,
  l.l_id,
  b.b_id,
  c.qty_in_stock,
  c.reorder_level,
  c.last_updated
FROM
  company c
  LEFT JOIN timedim t ON c.day = t.day
  AND c.month = t.month
  AND c.quarter = t.quarter
  AND c.years = t.years
  LEFT JOIN itemdim i ON c.item_name = i.item_name
  AND c.brand = i.brand
  AND c.sold_by = i.sold_by
  AND c.category = i.category
  LEFT JOIN locationdim l ON c.location_name = l.location_name
  AND c.state = l.state
  AND c.pin_code = l.pin_code
  LEFT JOIN branchdim b ON c.branch_name = b.branch_name
  AND c.branch_manager = b.branch_manager;

-- Example Query: Integrated Sales and Inventory Analysis
SELECT
  s.amt_sold,
  s.qty_sold,
  i.qty_in_stock,
  i.reorder_level,
  t.years,
  p.item_name
FROM
  salesfact s
  JOIN inventoryfact i ON s.t_id = i.t_id
  AND s.i_id = i.i_id
  JOIN timedim t ON s.t_id = t.t_id
  JOIN itemdim p ON s.i_id = p.i_id
WHERE
  t.years = 2022
  AND p.item_name = 'Car';
```

Screenshots of the different generated schemas

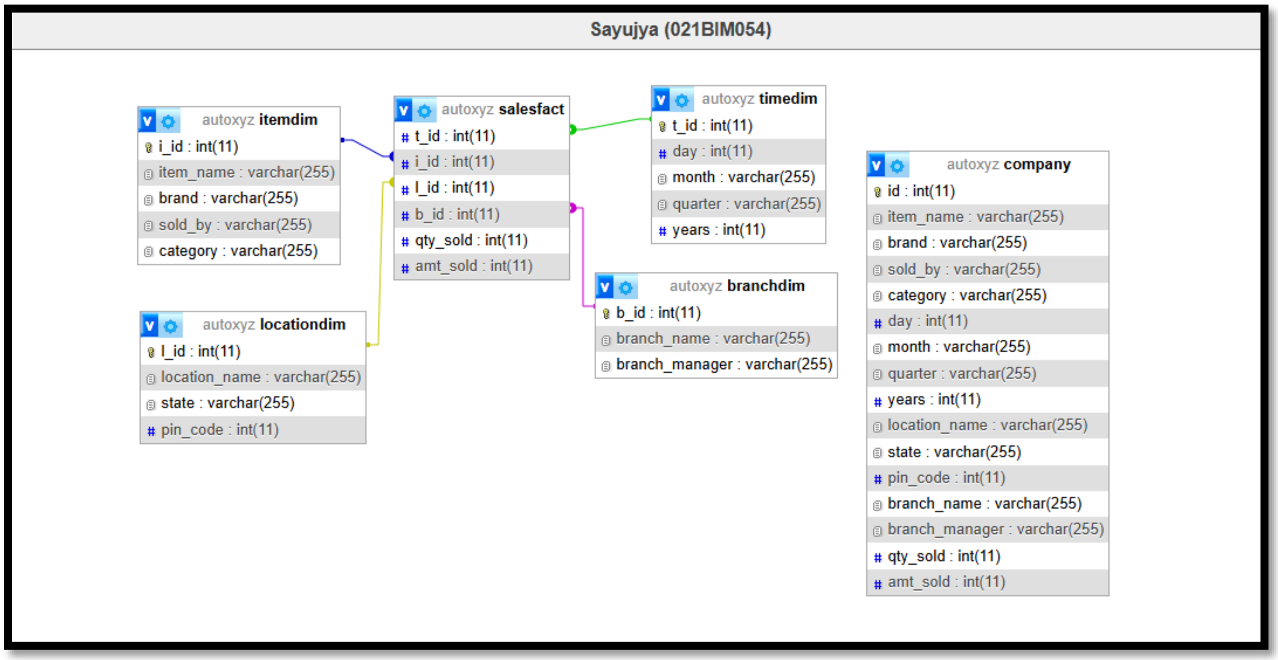

Figure 1: Star Schema

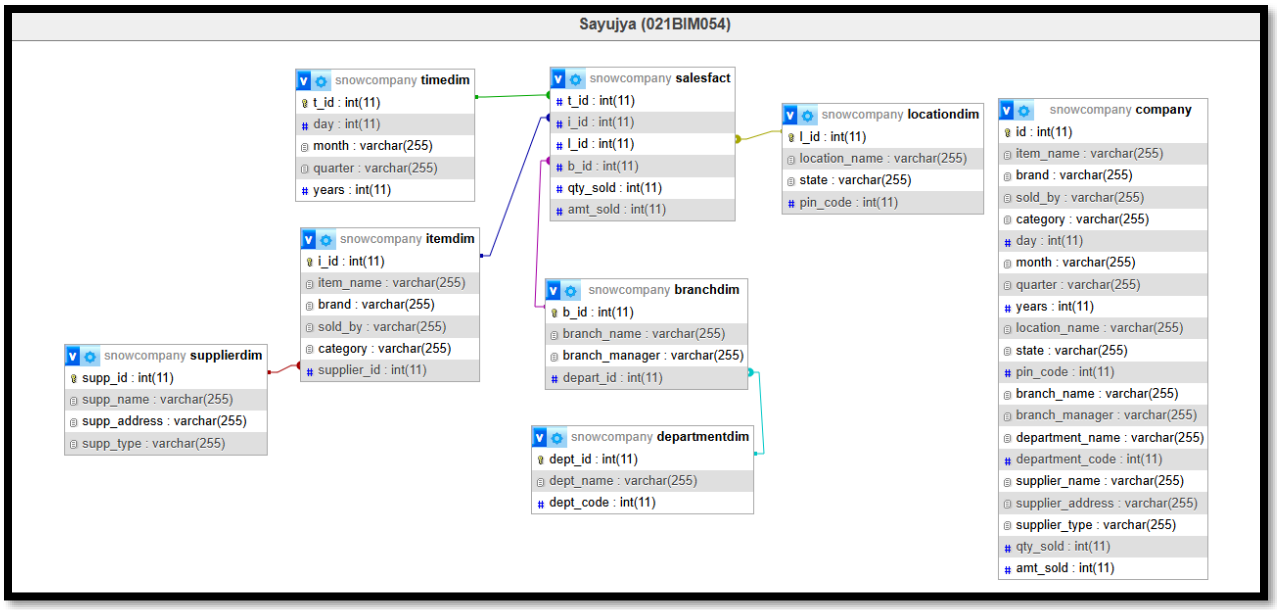

Figure 2: Snowflake Schema

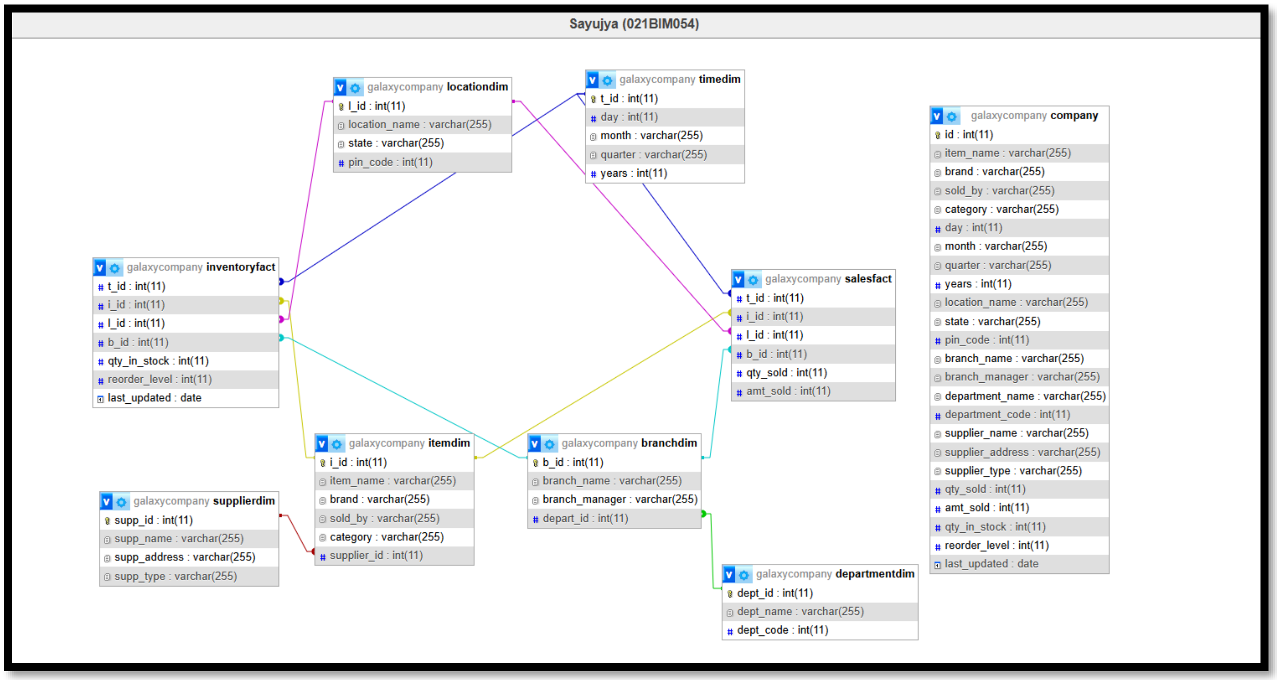

Figure 3: Galaxy Schema

Discussion

This lab explored the logical design of data warehouse schemas and their implementation using SQL in MySQL. Building the Star Schema demonstrated the simplest denormalized structure, where fact tables directly connect to dimension tables. This structure proved easy to query and visualize, highlighting why it is widely used in analytical systems.

The Snowflake Schema introduced normalization of dimensions, increasing structural complexity. Through additional tables such as supplier and department, the snowflake design showed how redundancy can be minimized, improving data integrity but requiring more joins in queries.

The Galaxy Schema (Fact Constellation) expanded the complexity further by introducing multiple fact tables, i.e. sales and inventory, sharing common dimensions. This schema reflected real world scenarios where organizations track several processes. Constructing these schemas in SQL helped clarify how dimensional modeling supports more flexible and scalable analytical queries.

Conclusion

This lab successfully illustrated the differences between Star, Snowflake, and Galaxy schemas. The SQL implementations showed how each schema type structures data, balances normalization, and affects query execution. By constructing dimension and fact tables and inserting data using joins, the lab demonstrated how data warehouses organize information for decision-making. The work reinforced the practical importance of schema selection in building efficient analytical systems.