Srushti Shah
191071902
Final Year B. Tech. CE
Data Mining and Warehouse

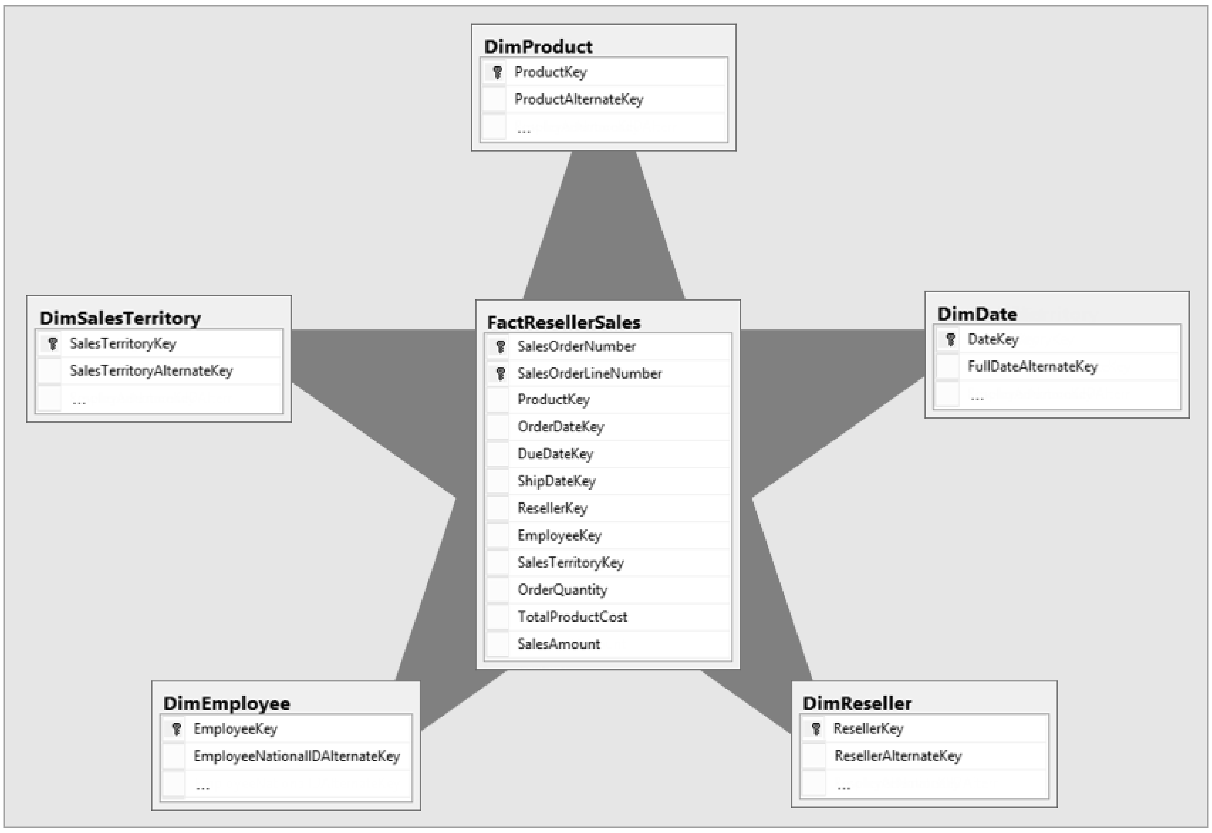
Experiment No. 1

AIM: To perform a multidimensional data model using SQL queries. e.g., snowflake, star and fact constellation schema

THEORY:

Star Schema

- **Star schema** is a mature modelling approach widely adopted by relational data warehouses. It requires modelers to classify their model tables as either *dimension* or *fact*.
- **Dimension tables** describe business entities—the *things* you model. Entities can include products, people, places, and concepts including time itself. The most consistent table you'll find in a star schema is a date dimension table. A dimension table contains a key column (or columns) that acts as a unique identifier, and descriptive columns.
- **Fact tables** store observations or events, and can be sales orders, stock balances, exchange rates, temperatures, etc. A fact table contains dimension key columns that relate to dimension tables, and numeric measure columns. The dimension key columns determine the *dimensionality* of a fact table, while the dimension key values determine the *granularity* of a fact table.
- For example, consider a fact table designed to store sale targets that has two-dimension key columns **Date** and **ProductKey**. It's easy to understand that the table has two dimensions. The granularity, however, can't be determined without considering the dimension key values. In this example, consider that the values stored in the **Date** column are the first day of each month. In this case, the granularity is at month-product level.
- Generally, dimension tables contain a relatively small number of rows. Fact tables, on the other hand, can contain a very large number of rows and continue to grow over time.



- The star schema is intensely suitable for data warehouse database design because of the following features:
  - It creates a DE-normalized database that can quickly provide query responses.
  - It provides a flexible design that can be changed easily or added to throughout the development cycle, and as the database grows.
  - It provides a parallel in design to how end-users typically think of and use the data.
  - It reduces the complexity of metadata for both developers and end-users.

Advantages of Star Schema

1. **Query Performance**: A star schema database has a limited number of table and clear join paths, the query run faster than they do against OLTP systems. Small single-table queries, frequently of a dimension table, are almost instantaneous. Large join queries that contain multiple tables takes only seconds or minutes to run. In a star schema database design, the dimension is connected only through the central fact table. When the two-dimension table is used in a query, only one join path, intersecting the fact tables, exist between those two tables. This design feature enforces authentic and consistent query results.
2. **Load performance and administration**: Structural simplicity also decreases the time required to load large batches of record into a star schema

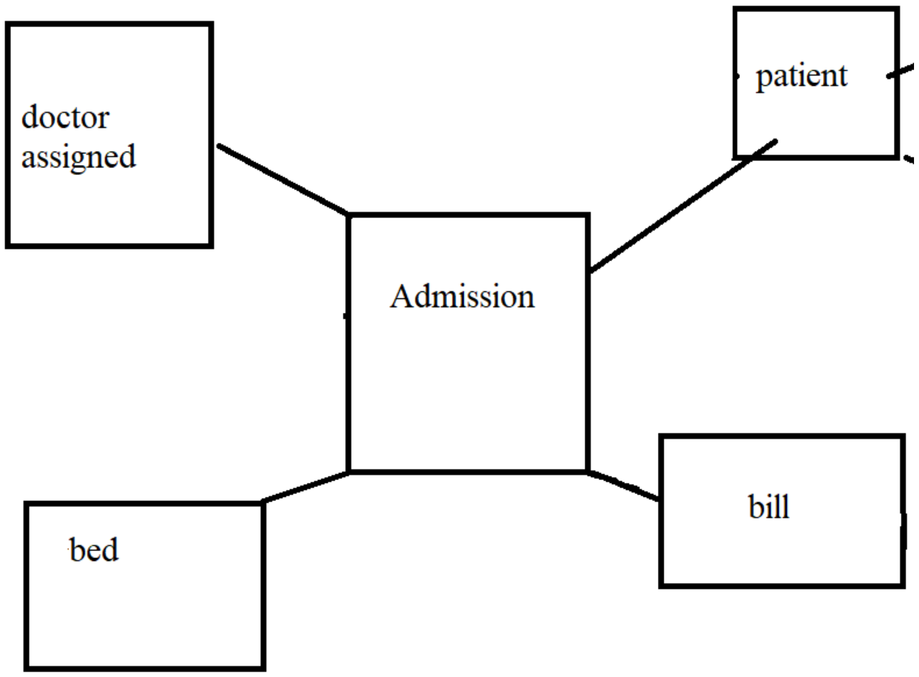
database. By describing facts and dimensions and separating them into the various table, the impact of a load structure is reduced. Dimension table can be populated once and occasionally refreshed. We can add new facts regularly and selectively by appending records to a fact table.

3. **Built-in referential integrity**: A star schema has referential integrity built-in when information is loaded. Referential integrity is enforced because each data in dimensional tables has a unique primary key, and all keys in the fact table are legitimate foreign keys drawn from the dimension table. A record in the fact table which is not related correctly to a dimension cannot be given the correct key value to be retrieved.
4. **Easily Understood**: A star schema is simple to understand and navigate, with dimensions joined only through the fact table. These joins are more significant to the end-user because they represent the fundamental relationship between parts of the underlying business. Customer can also browse dimension table attributes before constructing a query.

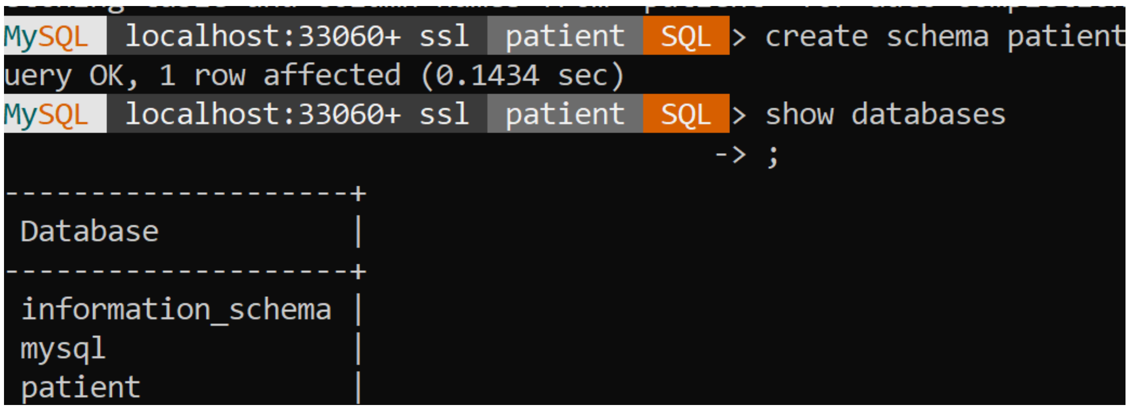
Disadvantage of Star Schema

1. There is some condition which cannot be meet by star schemas like the relationship between the user, and bank account cannot describe as star schema as the relationship between them is many to many.

<u>IMPLEMENTATION OF STAR SCHEMA USING SQL QUERY:</u>



1. create schema patient;



```
MySQL  localhost:33060+ ssl  patient  SQL > create schema patient
uery OK, 1 row affected (0.1434 sec)
MySQL  localhost:33060+ ssl  patient  SQL > show databases
                                          -> ;
--------------------+
 Database           |
--------------------+
 information_schema |
 mysql              |
 patient            |
```

2. Create fact table:
    a. create table admission (admitID int PRIMARY KEY, patientID int, doctorID int, billNo int, bedNo int, entryDate date, exitDate date)
        0 row(s) affected 14.984 sec
3. Create Dimension tables with the foreign key:

a. CREATE TABLE \`patient\`.\`patient\` (
    \`patientID\` INT NOT NULL, \`insuranceID\` INT NULL,
    \`patientName\` VARCHAR(45) NULL, \`dob\` DATE NULL,
    \`phoneNo\` VARCHAR(45) NULL,
    \`gender\` VARCHAR(45) NULL,
    \`address\` VARCHAR(45) NULL,
    PRIMARY KEY (\`patientID\`)
  );

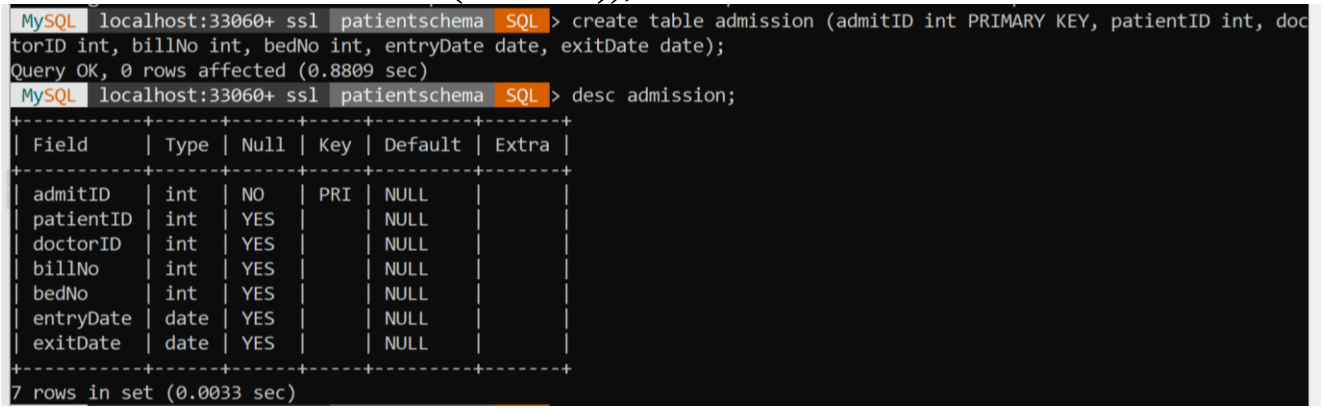
b. CREATE TABLE \`patient\`.\`doctor\` (
    \`doctorID\` INT NOT NULL,
    \`docName\` VARCHAR(45) NULL,
    \`degree\` VARCHAR(45) NULL,
    \`phoneNo\` VARCHAR(45) NULL,
    \`address\` VARCHAR(45) NULL,
    \`dob\` DATE NULL,
    \`joiningDate\` DATE NULL,
    \`gender\` VARCHAR(45) NULL,
    PRIMARY KEY (\`doctorID\`));

c. CREATE TABLE \`patient\`.\`bill\` (
    \`billNo\` INT NOT NULL,
    \`admitID\` INT NULL,
    \`insuranceID\` INT NULL,
    \`totalAmount\` INT NULL,
    \`AmtPaid\` INT NULL,
    \`AmtPending\` INT NULL,
    \`paymentMode\` VARCHAR(45) NULL,

```
            `Insured` TINYINT NULL,
            PRIMARY KEY (`billNo`))
          COMMENT = '                        ';
```

d. CREATE TABLE `patient`.`bed` (

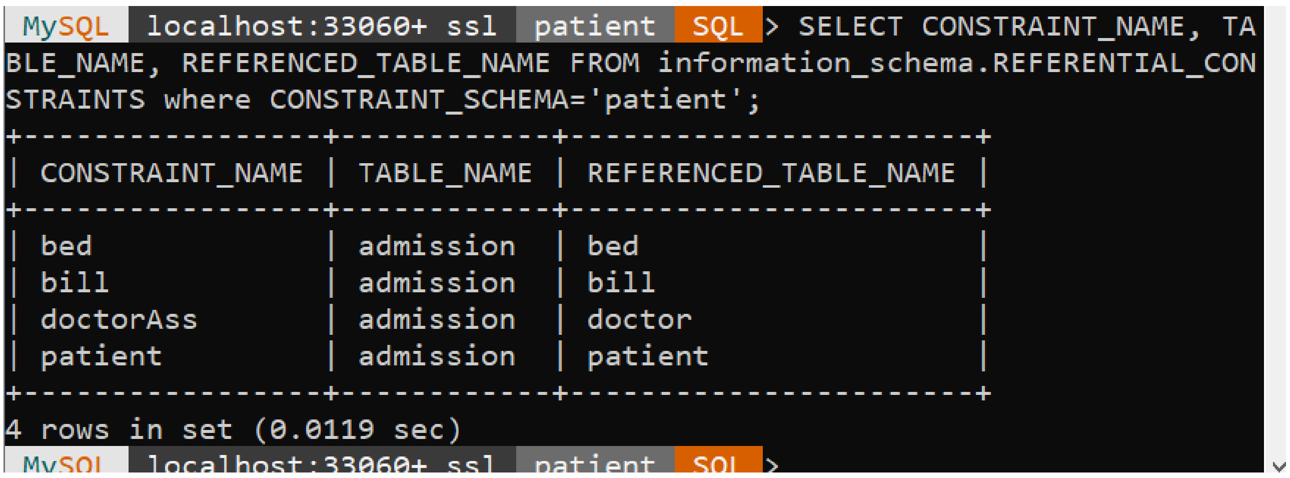
```
            `bedNo` INT NOT NULL,
            `wardID` INT NULL,
            `wardName` VARCHAR(45) NULL,
            `floor` INT NULL,
            `roomType` VARCHAR(45) NULL,
            `roomNo` INT NULL,
            PRIMARY KEY (`bedNo`));
```



```
MySQL  localhost:33060+ ssl  patientschema  SQL > create table admission (admitID int PRIMARY KEY, patientID int, doc
torID int, billNo int, bedNo int, entryDate date, exitDate date);
Query OK, 0 rows affected (0.8809 sec)
MySQL  localhost:33060+ ssl  patientschema  SQL > desc admission;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| admitID   | int  | NO   | PRI | NULL    |       |
| patientID | int  | YES  |     | NULL    |       |
| doctorID  | int  | YES  |     | NULL    |       |
| billNo    | int  | YES  |     | NULL    |       |
| bedNo     | int  | YES  |     | NULL    |       |
| entryDate | date | YES  |     | NULL    |       |
| exitDate  | date | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
7 rows in set (0.0033 sec)
```

e. Adding foreight key constraints to each table:

```
      ALTER TABLE `patient`.`admission`
      ADD INDEX `patient_idx` (`patientID` ASC) VISIBLE,
      ADD INDEX `bed_idx` (`bedNo` ASC) VISIBLE,
      ADD INDEX `bill_idx` (`billNo` ASC) VISIBLE,
      ADD INDEX `doctorAss_idx` (`doctorID` ASC) VISIBLE;;
      ALTER TABLE `patient`.`admission`
      ADD CONSTRAINT `patient` FOREIGN KEY
      (`patientID`)REFERENCES `patient`.`patient` (`patientID`),
      ADD CONSTRAINT `bed`
        FOREIGN KEY (`bedNo`)REFERENCES `patient`.`bed` (`bedNo`),
      ADD CONSTRAINT `bill` FOREIGN KEY (`billNo`)  REFERENCES
      `patient`.`bill` (`billNo`),
      ADD CONSTRAINT `doctorAss` FOREIGN KEY (`doctorID`)
      REFERENCES `patient`.`doctor` (`doctorID`);
```

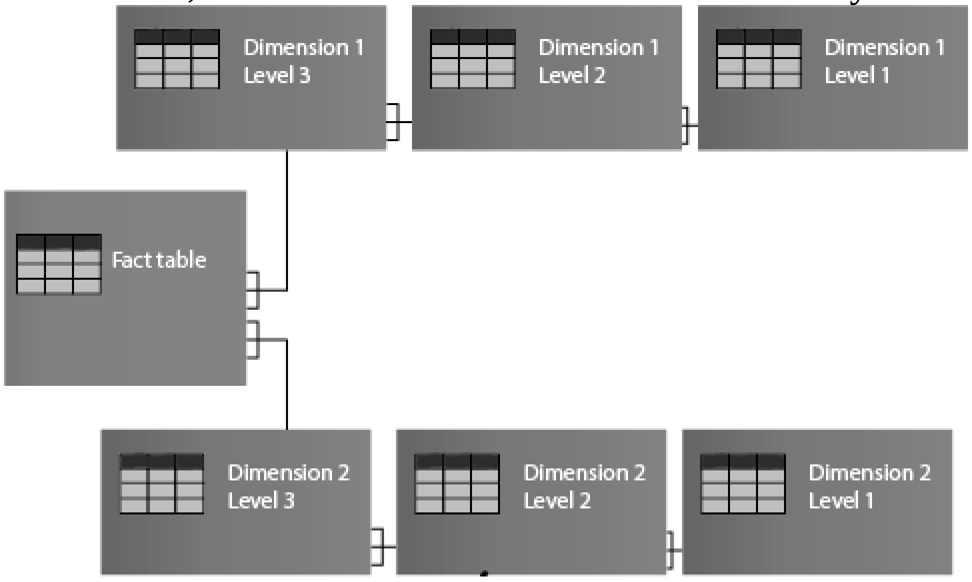


```
MySQL  localhost:33060+ ssl  patient  SQL > SELECT CONSTRAINT_NAME, TABLE_NAME, REFERENCED_TABLE_NAME FROM information_schema.REFERENTIAL_CONSTRAINTS where CONSTRAINT_SCHEMA='patient';
+-----------------+------------+-----------------------+
| CONSTRAINT_NAME | TABLE_NAME | REFERENCED_TABLE_NAME |
+-----------------+------------+-----------------------+
| bed             | admission  | bed                   |
| bill            | admission  | bill                  |
| doctorAss       | admission  | doctor                |
| patient         | admission  | patient               |
+-----------------+------------+-----------------------+
4 rows in set (0.0119 sec)
MySQL  localhost:33060+ ssl  patient  SQL >
```

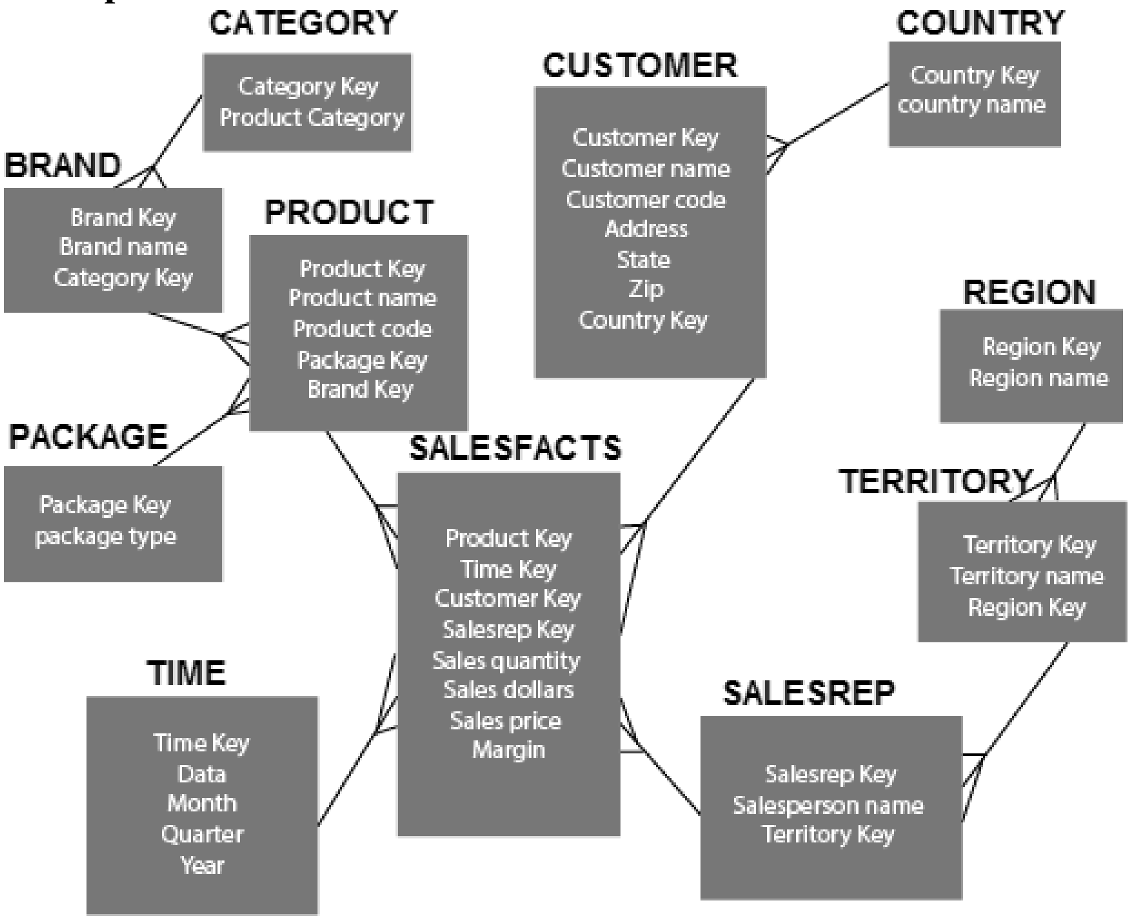
Snowflake schema

- **Snowflake schema** is equivalent to the star schema. "A schema is known as a snowflake if one or more-dimension tables do not connect directly to the fact table but must join through other dimension tables."
- The snowflake schema is an expansion of the star schema where each point of the star explodes into more points. It is called snowflake schema because the diagram of snowflake schema resembles a snowflake. Snowflaking is a method of normalizing the dimension tables in a STAR schema. When we normalize all the dimension tables entirely, the resultant structure resembles a snowflake with the fact table in the middle.
- Snowflaking is used to develop the performance of specific queries. The schema is diagrammed with each fact surrounded by its associated dimensions, and those dimensions are related to other dimensions, branching out into a snowflake pattern.
- The snowflake schema consists of one fact table which is linked to many dimension tables, which can be linked to other dimension tables through a many-to-one relationship. Tables in a snowflake schema are generally normalized to the third normal form. Each dimension table performs exactly one level in a hierarchy.
- The following diagram shows a snowflake schema with two dimensions, each having three levels. A snowflake schema can have any number of

dimensions, and each dimension can have any number of levels.



- **Example:**



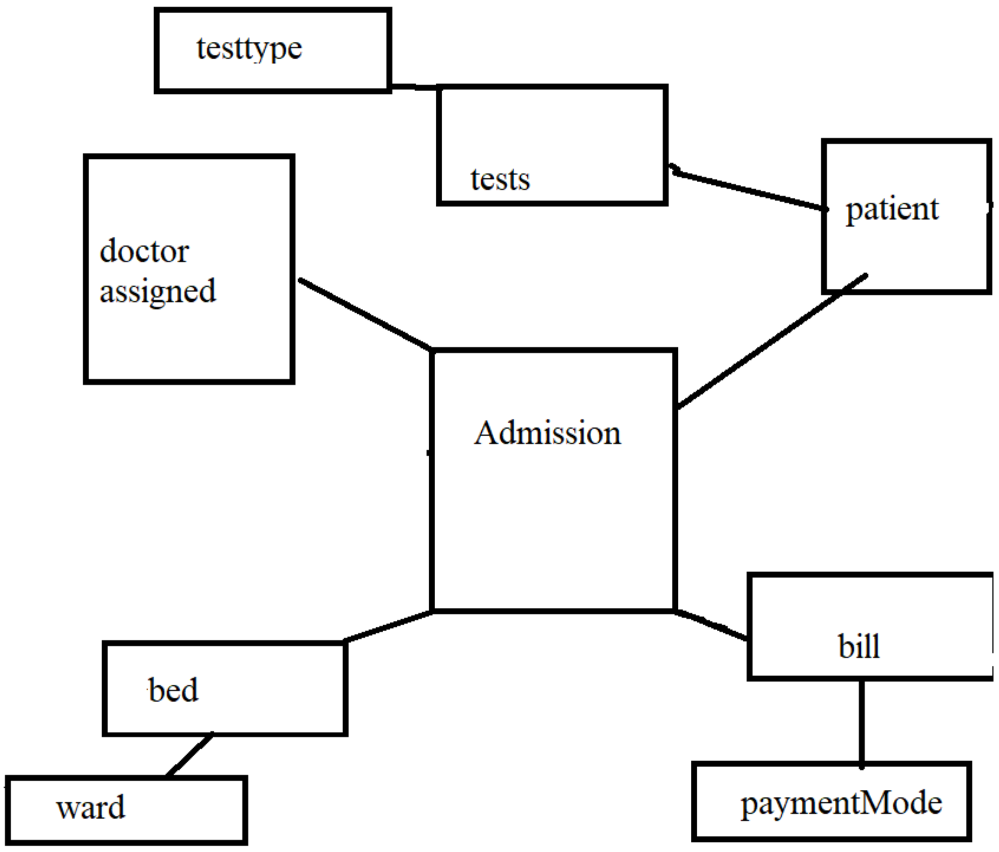
- A snowflake schema is designed for flexible querying across more complex dimensions and relationship. It is suitable for many to many and one to many relationships between dimension levels.
- **Advantage of Snowflake Schema**

1. The primary advantage of the snowflake schema is the development in query performance due to minimized disk storage requirements and joining smaller lookup tables.
2. It provides greater scalability in the interrelationship between dimension levels and components.
3. No redundancy, so it is easier to maintain.

- **Disadvantage of Snowflake Schema**

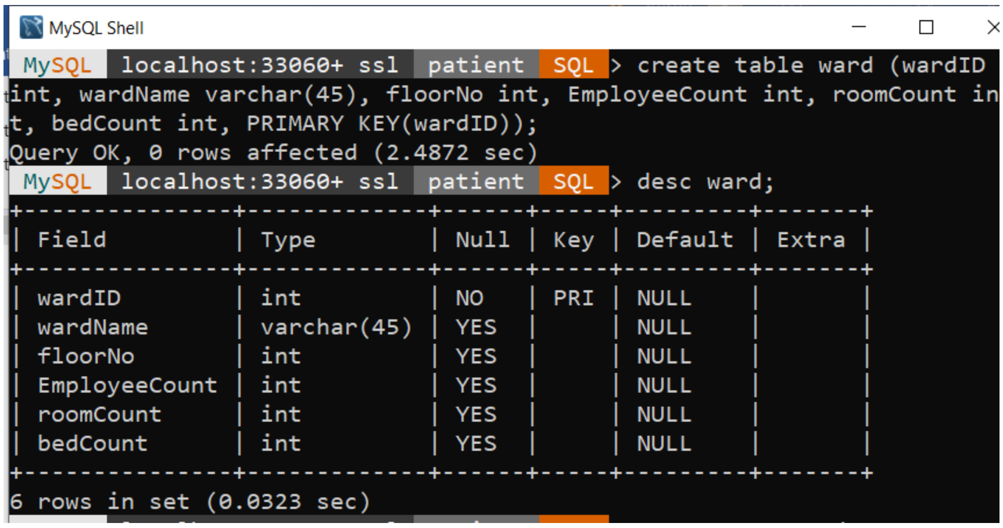
1. The primary disadvantage of the snowflake schema is the additional maintenance efforts required due to the increasing number of lookup tables. It is also known as a multi fact star schema.
2. There are more complex queries and hence, difficult to understand.
3. More tables more join so more query execution time.

IMPLEMENTATION AND OUTPUT:



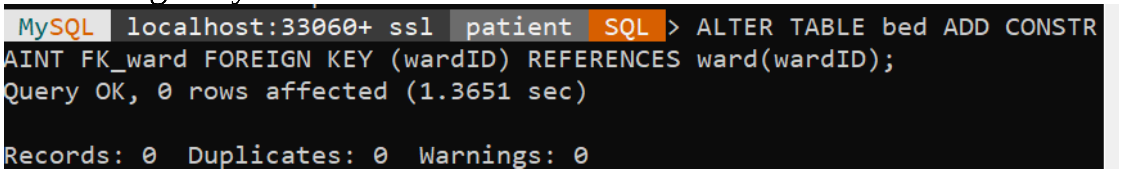
1. Add dimension table to bed

a. Create ward table



```
MySQL  localhost:33060+ ssl  patient  SQL > create table ward (wardID
int, wardName varchar(45), floorNo int, EmployeeCount int, roomCount in
t, bedCount int, PRIMARY KEY(wardID));
Query OK, 0 rows affected (2.4872 sec)
MySQL  localhost:33060+ ssl  patient  SQL > desc ward;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| wardID        | int         | NO   | PRI | NULL    |       |
| wardName      | varchar(45) | YES  |     | NULL    |       |
| floorNo       | int         | YES  |     | NULL    |       |
| EmployeeCount | int         | YES  |     | NULL    |       |
| roomCount     | int         | YES  |     | NULL    |       |
| bedCount      | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.0323 sec)
```

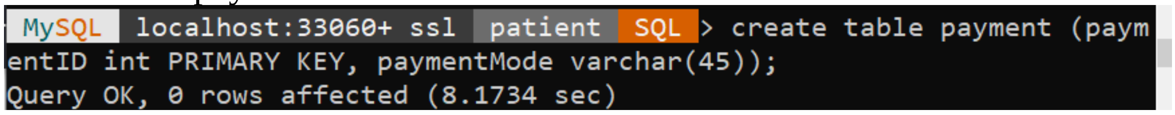
b. Add foreign key constraint to bed



```
MySQL  localhost:33060+ ssl  patient  SQL > ALTER TABLE bed ADD CONSTR
AINT FK_ward FOREIGN KEY (wardID) REFERENCES ward(wardID);
Query OK, 0 rows affected (1.3651 sec)

Records: 0  Duplicates: 0  Warnings: 0
```

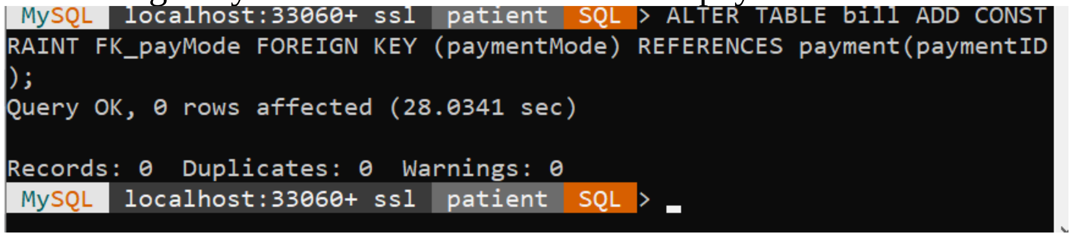
2. Add dimension table for payment mode

a. Create table payment



```
MySQL  localhost:33060+ ssl  patient  SQL > create table payment (paym
entID int PRIMARY KEY, paymentMode varchar(45));
Query OK, 0 rows affected (8.1734 sec)
```

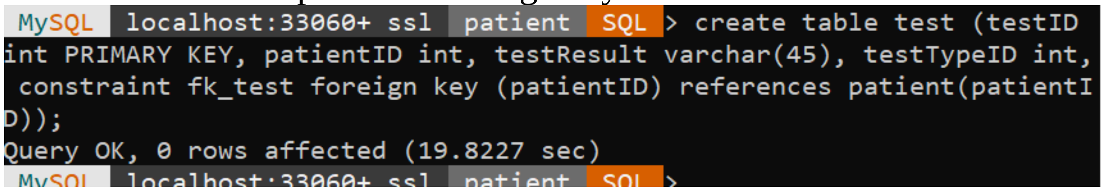
b. Add foreign key constraint to bill to reference paymentModeID



```
MySQL  localhost:33060+ ssl  patient  SQL > ALTER TABLE bill ADD CONST
RAINT FK_payMode FOREIGN KEY (paymentMode) REFERENCES payment(paymentID
);
Query OK, 0 rows affected (28.0341 sec)

Records: 0  Duplicates: 0  Warnings: 0
MySQL  localhost:33060+ ssl  patient  SQL >
```

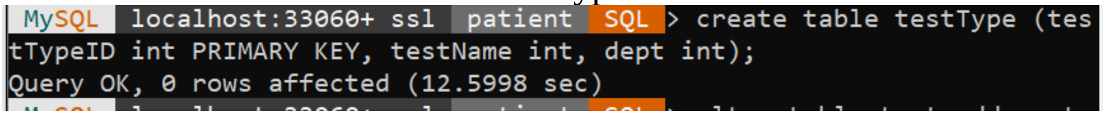
3. Add dimension test table

a. Add test table with patient as foreign key



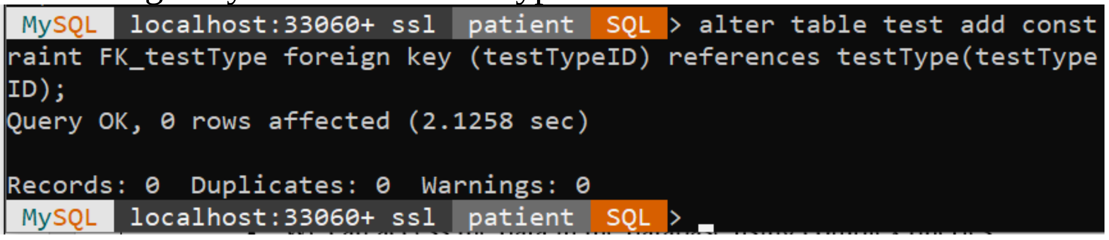
```
MySQL  localhost:33060+ ssl  patient  SQL > create table test (testID
int PRIMARY KEY, patientID int, testResult varchar(45), testTypeID int,
 constraint fk_test foreign key (patientID) references patient(patientI
D));
Query OK, 0 rows affected (19.8227 sec)
MySQL  localhost:33060+ ssl  patient  SQL >
```

b. Add dimension to test table with testTypeID



```
MySQL  localhost:33060+ ssl  patient  SQL > create table testType (tes
tTypeID int PRIMARY KEY, testName int, dept int);
Query OK, 0 rows affected (12.5998 sec)
```

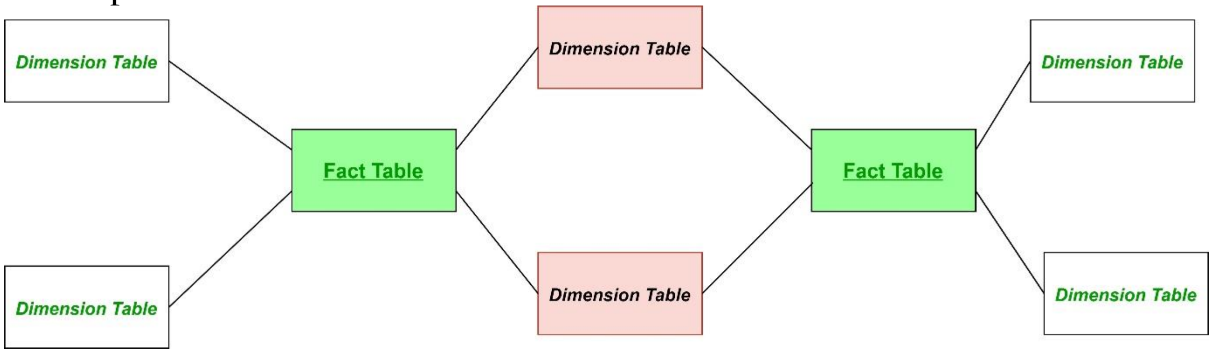
c. Add foreign key reference to testType



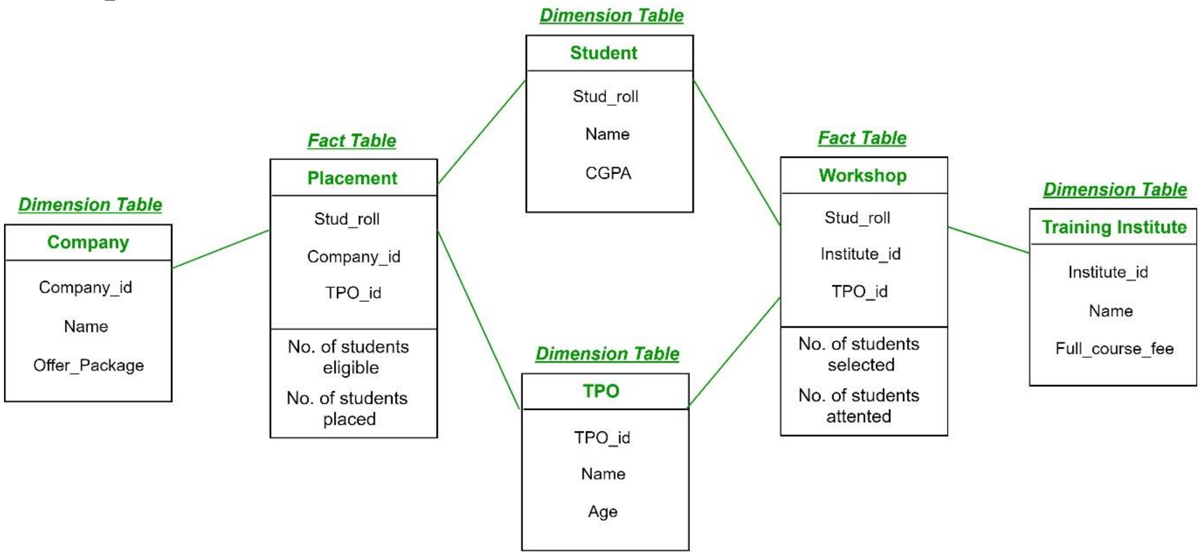
```
MySQL  localhost:33060+ ssl  patient  SQL > alter table test add constraint FK_testType foreign key (testTypeID) references testType(testTypeID);
Query OK, 0 rows affected (2.1258 sec)

Records: 0  Duplicates: 0  Warnings: 0
MySQL  localhost:33060+ ssl  patient  SQL >
```

Fact Constellation Schema

- Fact Constellation Schema is a schema that represents a multidimensional model of tables. This schema is a group of different fact tables that have few similar dimensional tables. It can be represented as a group of multiple star schemas and thus, it is also called a Galaxy schema. Fact schema is the most frequently used schema to design a Data warehouse and also, it is a little complicated than star and snowflake schema model.
- Fact constellation schema is a tool of analytical processing via online, which has a huge group of the number of fact tables that share dimensional tables, also aggregated as a group of stars. We can also call it as an extension of the star constellation model.
- A fact constellation schema can have multiple fact tables associated with it. It is commonly used a schema for designing data warehouses and it is much complicated than any other schema such as star and snowflake schema. We can create a fact constellation schema from a star schema by splitting them into one or more-star schemas. It is also said that a fact constellation schema can have many fact tables and a shared dimensional table.
- Architecture: Here the pink-coloured box is dimensional tables that are common to both the star schemas. Green coloured are the fact tables that are responsible for each of their star schemas.



- **Example:**



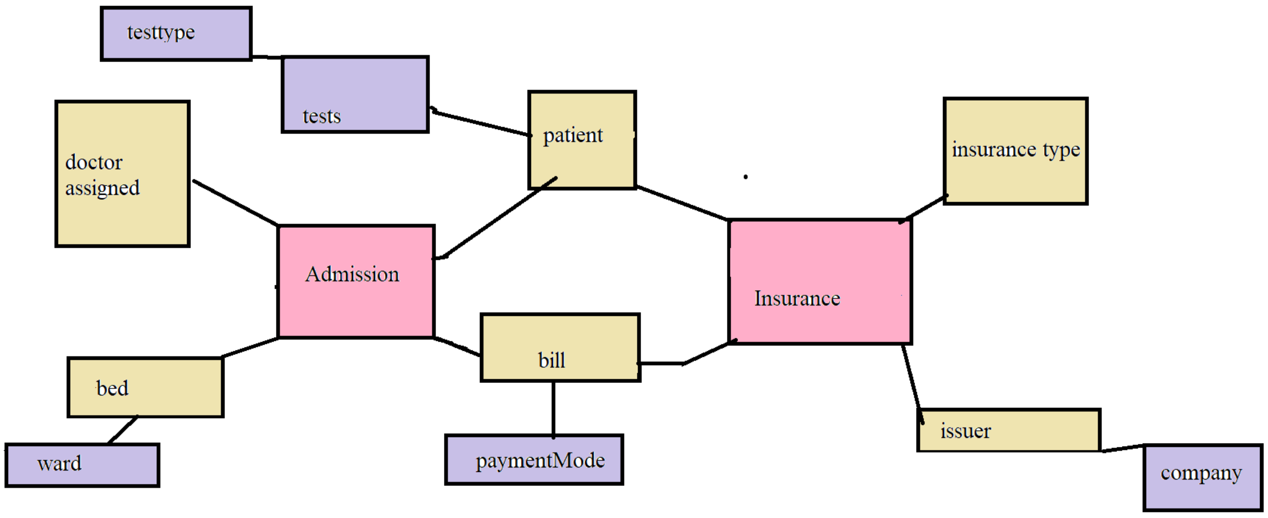
Advantages

- Tables are subdivided into fact and dimensional to understand the relationship between them.
- It is a flexible schema that makes users use it.
- Here dimensional tables are shared by the number of fact tables.
- It is a normalized form of snowflake and star schema.
- We can access the data in the database using complex queries.

Disadvantages:

- It is difficult to understand as it is a very complex schema to implement.
- It uses more space in the database comparative to the star schema.
- It has many joins between dimensional and fact tables and thus it is difficult to understand.
- This is difficult to maintain and operate.

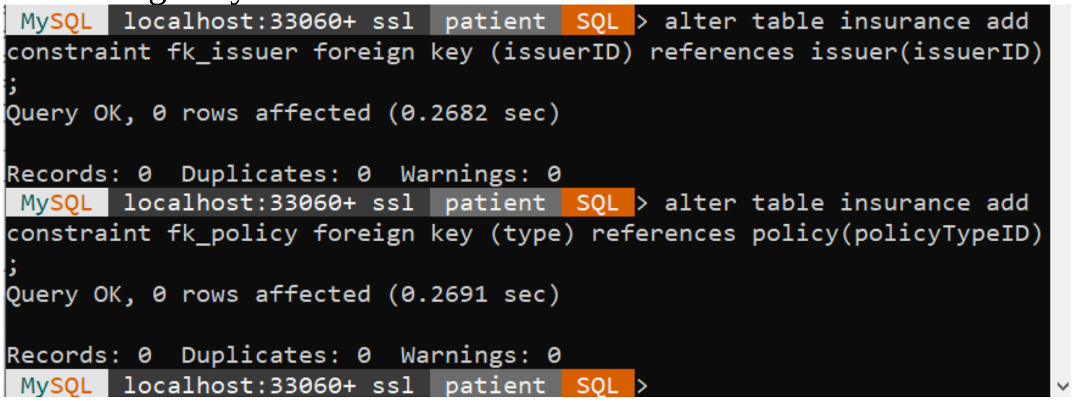
<u>IMPLEMENTATION AND OUTPUT:</u>



1. Create another fact table: Insurance

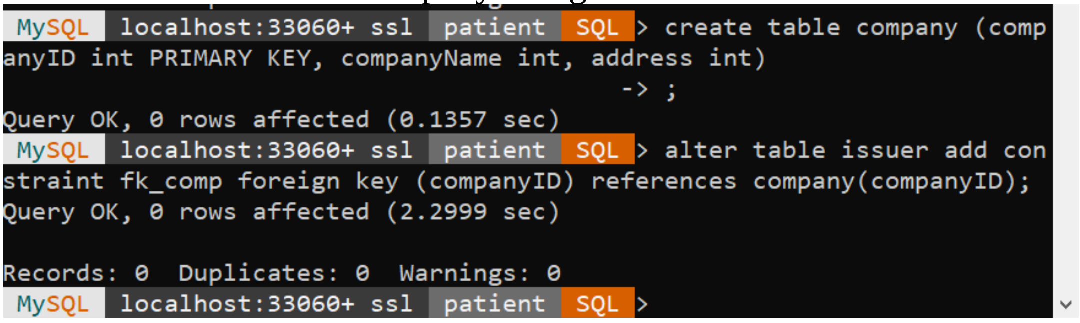
CREATE TABLE `patient`.`insurance` (
`insuranceID` INT NOT NULL, `type` INT NOT NULL,
`issuerID` VARCHAR(45) NULL, `price` INT NULL, `CoverageAmt` VARCHAR(45) NULL, `issueDate` VARCHAR(45) NULL, PRIMARY KEY (`insuranceID`));

2. Create Dimension Table
   a. Create issuer table
      CREATE TABLE `patient`.`issuer` (`issuerID` INT NOT NULL, `issuerName` VARCHAR(45) NULL, `companyID` INT NULL, `experience` INT NULL, PRIMARY KEY (`issuerID`));
   b. Create policyType
      CREATE TABLE `patient`.`policy` (
      `policyTypeID` INT NOT NULL, `policyName` VARCHAR(45) NULL, `Coverage` VARCHAR(45) NULL,
      `Preminum` VARCHAR(45) NULL, `Term` INT NULL,
      `Conditions` VARCHAR(45) NULL, PRIMARY KEY (`policyTypeID`));
3. Add foreign key constraints



```
 MySQL  localhost:33060+ ssl  patient  SQL > alter table insurance add
constraint fk_issuer foreign key (issuerID) references issuer(issuerID)
;
Query OK, 0 rows affected (0.2682 sec)

Records: 0  Duplicates: 0  Warnings: 0
 MySQL  localhost:33060+ ssl  patient  SQL > alter table insurance add
constraint fk_policy foreign key (type) references policy(policyTypeID)
;
Query OK, 0 rows affected (0.2691 sec)

Records: 0  Duplicates: 0  Warnings: 0
 MySQL  localhost:33060+ ssl  patient  SQL >
```

4. Create dimension table company to tag with issuer



```
 MySQL  localhost:33060+ ssl  patient  SQL > create table company (comp
anyID int PRIMARY KEY, companyName int, address int)
                                          -> ;
Query OK, 0 rows affected (0.1357 sec)
 MySQL  localhost:33060+ ssl  patient  SQL > alter table issuer add con
straint fk_comp foreign key (companyID) references company(companyID);
Query OK, 0 rows affected (2.2999 sec)

Records: 0  Duplicates: 0  Warnings: 0
 MySQL  localhost:33060+ ssl  patient  SQL >
```

CONCLUSION: Thus, from this experiment I executed the star schema with dimension and fact tables, snowflake schema and fast constellation schema using SQL Queries.