|  |  |
| --- | --- |
| Semester | T.E. Semester V – Computer Engineering |
| Subject | Data Warehousing and Mining |
| Subject Professor In-charge | Prof. Kavita Shirsat |
| Assisting Teachers | Prof. Kavita Shirsat |
| Laboratory | Lab 312 A |

|  |  |  |
| --- | --- | --- |
| Student Name | Sahil Pokharkar | |
| Roll Number | 21102A0006 | |
| Grade and Subject  Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| PBLE | Implementation of star schema | |
| Resources / Apparatus Required | Hardware:  Computer system | Software:  PostgreSQL |
| Description | In the field of data warehousing, two common schema designs are widely used: the star schema and the snowflake schema. These schema designs are crucial for organizing data in a data warehousing environment to facilitate efficient querying and analysis. Both schemas have their own advantages and are suitable for different types of data analysis and reporting.  **Star Schema:**   1. **Structure**: The star schema is a simple and denormalized design that consists of a single fact table referencing multiple dimension tables. The fact table contains the measures or metrics of interest, while the dimension tables contain the descriptive attributes associated with the measures. 2. **Components**:    * Fact Table: Contains the primary business metrics.    * Dimension Tables: Provide descriptive information related to the data in the fact table. 3. **Simplicity**: It is straightforward and easy to understand, making it a popular choice for data warehousing. 4. **Performance**: Query performance is usually better in a star schema because it involves fewer joins. 5. **Denormalization**: The data is partially denormalized, reducing the need for multiple joins and simplifying queries.   **Snowflake Schema:**   1. **Structure**: The snowflake schema is an extension of the star schema. It normalizes the dimension tables, which means the dimension tables are normalized into multiple related tables. 2. **Components**:    * Fact Table: Contains the primary business metrics.    * Dimension Tables: Each dimension table may be normalized into multiple related tables. 3. **Normalization**: The snowflake schema is a normalized form of the star schema, leading to reduced data redundancy. 4. **Complexity**: It is more complex than the star schema due to the presence of additional tables. 5. **Storage Efficiency**: Snowflake schema requires more storage space due to the normalization of dimension tables. 6. **Query Performance**: The snowflake schema might involve more complex joins, which can impact query performance compared to the star schema.   **Considerations:**   * **Usage**: The choice between star and snowflake schemas depends on the specific requirements of the business and the complexity of the data being analyzed. * **Performance vs. Normalization**: While the snowflake schema provides better normalization, the star schema typically offers better query performance. | |
| Program | CREATE TABLE sales\_fact (  sale\_id SERIAL PRIMARY KEY,  product\_id INT,  customer\_id INT,  time\_id INT,  location\_id INT,  quantity INT,  total\_amount DECIMAL(10, 2)  );  CREATE TABLE product\_dim (  product\_id SERIAL PRIMARY KEY,  product\_name VARCHAR(255),  brand VARCHAR(50),  category\_id INT,  price DECIMAL(10, 2)  );  CREATE TABLE customer\_dim (  customer\_id SERIAL PRIMARY KEY,  customer\_name VARCHAR(255),  email VARCHAR(255),  city VARCHAR(100),  state VARCHAR(50)  );  CREATE TABLE time\_dim (  time\_id SERIAL PRIMARY KEY,  order\_date DATE,  year INT,  month INT,  day INT  );  CREATE TABLE location\_dim (  location\_id SERIAL PRIMARY KEY,  city VARCHAR(100),  state VARCHAR(50),  country VARCHAR(50)  );  CREATE TABLE category\_dim (  category\_id SERIAL PRIMARY KEY,  category\_name VARCHAR(100)  );  -- Insert 50 entries into category\_dim table  INSERT INTO category\_dim (category\_name)  SELECT 'Category ' || generate\_series FROM generate\_series(1, 50);  -- Insert 50 entries into product\_dim table  INSERT INTO product\_dim (product\_name, brand, category\_id, price)  SELECT 'Product ' || generate\_series, 'Brand ' || generate\_series, (generate\_series % 50) + 1, (generate\_series \* 10)::DECIMAL(10, 2) FROM generate\_series(1, 50);  -- Insert 50 entries into customer\_dim table  INSERT INTO customer\_dim (customer\_name, email, city, state)  SELECT 'Customer ' || generate\_series, 'customer' || generate\_series || '@example.com', 'City ' || generate\_series, 'State ' || generate\_series FROM generate\_series(1, 50);  -- Insert 50 entries into time\_dim table  INSERT INTO time\_dim (order\_date, year, month, day)  SELECT '2023-01-01'::DATE + (generate\_series - 1), 2023, EXTRACT(MONTH FROM '2023-01-01'::DATE + (generate\_series - 1)), EXTRACT(DAY FROM '2023-01-01'::DATE + (generate\_series - 1)) FROM generate\_series(1, 50);  -- Insert 50 entries into location\_dim table  INSERT INTO location\_dim (city, state, country)  SELECT 'City ' || generate\_series, 'State ' || generate\_series, 'Country ' || generate\_series FROM generate\_series(1, 50);  -- Insert 50 entries into sales\_fact table  INSERT INTO sales\_fact (product\_id, customer\_id, time\_id, location\_id, quantity, total\_amount)  SELECT (generate\_series % 50) + 1, (generate\_series % 50) + 1, (generate\_series % 50) + 1, (generate\_series % 50) + 1, (generate\_series % 10) + 1, ((generate\_series % 50) + 1) \* 100 FROM generate\_series(1, 50);  --Query 1: Get a list of all products with their prices.  SELECT product\_name, price  FROM product\_dim;  --Query 2: Find the total sales for each category.  SELECT c.category\_name, SUM(sf.total\_amount) AS total\_sales  FROM sales\_fact sf  JOIN product\_dim p ON sf.product\_id = p.product\_id  JOIN category\_dim c ON p.category\_id = c.category\_id  GROUP BY c.category\_name;  --Query 3: Find the top-selling customer.  SELECT c.customer\_name, SUM(sf.quantity) AS total\_quantity  FROM sales\_fact sf  JOIN customer\_dim c ON sf.customer\_id = c.customer\_id  GROUP BY c.customer\_name  ORDER BY total\_quantity DESC  LIMIT 1;  --Query 4: List the sales for a specific day.  SELECT t.order\_date, SUM(sf.total\_amount) AS total\_sales  FROM sales\_fact sf  JOIN time\_dim t ON sf.time\_id = t.time\_id  WHERE t.order\_date = '2023-01-02'  GROUP BY t.order\_date;  --Query 5: Find the average price of products in each category.  SELECT c.category\_name, AVG(p.price) AS average\_price  FROM product\_dim p  JOIN category\_dim c ON p.category\_id = c.category\_id  GROUP BY c.category\_name;  SELECT\* FROM sales\_fact; | |
| Output | 1. Query 1   A screenshot of a computer  Description automatically generated   1. Query 2   A screenshot of a computer  Description automatically generated   1. Query 3   A screenshot of a computer  Description automatically generated   1. Query 4      1. Query 5 | |
| Conclusion: | In conclusion, the star schema is a simple, denormalized structure with a single fact table and several dimension tables, ensuring better query performance due to fewer joins. It is efficient for simple querying and analysis, suitable for less complex data. On the other hand, the snowflake schema is a normalized version of the star schema, with dimension tables further broken down into related tables, improving data integrity but increasing complexity and storage requirements. While it offers improved data consistency, the snowflake schema may lead to more complex querying and potentially slower performance compared to the star schema. The choice between these schemas depends on the complexity of the data and the specific requirements of the analytical processes. | |