Database Technologies Project – Phase 5

Overview

In the fifth phase of the project, we understand the concepts of dimensional modelling and data warehousing, and work on a real-world scenario for extracting insights from collected data. The main objective is to create a dimensional model to analyse user call records, and extract insights from it as per the provided specifications. The two main components of this are

1. Creation of the dimensional model to capture all the relevant information in the form of fact and dimension tables
2. Creation of appropriate OLAP queries to extract desired insights from the data

The entire project can be found at https://github.com/anihm136/DBMS_Project. The project uses SQL Server 2019 on a Docker container to perform the necessary SQL queries

A. Creation of dimensional model

In order to simplify data retrieval, a **star model** is used to create the dimensional model, which consists of a central fact table, and surrounding dimension tables.

The **fact table** for this application consists of call records for all numbers maintained by the company. Each record consists of a call ID, a user ID, an account ID, a phone number ID, an origin tower ID and a destination tower ID. These fields are foreign keys to the respective dimension tables, which are used to obtain further information about each subject. This model allows for each user to have multiple accounts, each account to maintain multiple phone numbers and multiple tower locations

The **dimension tables** contain information about each of the subjects in the model. The following dimension tables are used -

1. **User information:** Contains information about each user. It has a primary key of user ID, and stores the users' names and addresses
2. **Call details:** Contains information about each call. It has a primary key of call ID, and contains information about whether the call was successful or not, the start time and the end time of the call
3. **Account information:** Contains information about each user account. It has a primary key of account ID, and contains the account category (personal or corporate), the rate for outgoing and incoming calls and the limit for outgoing and incoming calls (if any). It also contains the start date for the account I.e, when the account was created
4. **Phone number information:** Contains information about each phone number. It has a primary key of phone number ID, and contains the phone number, extension and whether it is currently active
5. **Tower information:** Contains information about cell towers. It has a primary key of tower ID, and contains the area, city, state and country in which the tower is located

These dimensions will be used to obtain the necessary business insights required by the different users as well as the company

The SQL used to create the tables is uploaded to the submisison folder,as `create.sql`.

B. Performing OLTP on the data

Note: Due to difficulty in setting up SSAS on Linux, the actual OLTP actions have not been performed on the data. Instead, a detailed description of the procedures to retrieve the necessary data is provided below

Once the data model is ready, and the tables have been filled with data by transactions, the next step is to build a multi-dimensional model to perform OLTP on it. **SQL Server Analysis Services**(SSAS) is a suite of services by Microsoft, commonly used for OLTP. The following steps describe the procedure to perform OLTP using SSAS

First, to simplify data retrieval, a **data cube** is constructed. This can be simply visualised as a multi-dimensional view of the data, where one dimension (provided by the fact table) is time, representing transactions, and the other dimensions (provided by the dimension tables) represent information about specific subjects that are referenced by the fact table. To create the data cube,

1. Create a new Analysis Services project in Visual Studio, and import the existing database as a data source
2. Choose the necessary dimension tables and the attributes (or measures) required from them
3. Choose the fact table to provide the time dimension. If required, the software will prompt the user to import additional dimensions as referenced by the fact table
4. Check the logical diagram to ensure that all facts and dimensions are as expected, and build the project to create the cube
5. Deploy the cube on Azure Analytics Services to obtain OLTP capabilities on the model

Next, we can perform OLTP on the created data cube. The query language used for OLTP in SSAS is known as **Multidimensional Expressions** (MDX). It is similar in syntax to SQL, and allows users to retrieve various dimensions of the data cube and represent them on various axes to present them (also known as slicing and dicing the cube). The basic procedure is as follows -

1. Import the data cube created previously in SQL Server Management Studio by connecting to the SSAS instance where the model was deployed
2. Write MDX queries to retrieve information from the cube. Each MDX query has the following basic components -
   a) A list of dimensions to represent on the X axis (ROWS)
   b) A list of dimensions to represent on the Y axis (COLUMNS)

c) The data cube from which information is to be retrieved
d) Filtering conditions (similar to SQL)

SSAS currently supports up to 128 axes to represent dimensions, although generally only two are used to maintain simplicity of presentation. Dimensions can also be calculated at query runtime, using both existing dimensions and previously defined queries. These are useful in generating desired fields which may not be recorded in the underlying database.

Here, some queries are provided to obtain specific insights, as detailed by the problem statement. All queries assume that a data cube , named "CallRecords", was designed with the fact table and dimension tables described above

1. **Corporate companies want to know the mobile usage patterns of their employees so that they can explore opportunities to cut cost**

   One potential way to cut costs would be to restrict usage of the corporate mobile plan outside of business hours. To determine if this is a viable solution, it would be useful to identify the total talk time spent by employees, and how much it costed the company

```
WITH Measures.[Talktime] AS Measures.[CallDetails].
[Endtime]-Measures.[CallDetails].[Starttime]
SELECT Measures.[Talktime] ON COLUMNS, Measures.
[PhDetails].[PhNum] ON ROWS FROM CallRecords WHERE
Measures.[AccountDetails].AccountID=101
```

2. **Individuals want to know their mobile usage patterns so that they can move to different plans**

   One potential usage pattern could be that a user has high usage during a specific interval each month, and very low usage for the rest of the month. Identifying this pattern could help such customers move to plans that are suited for bursts of usage

```
WITH Measures.[Talktime] AS Measures.[CallDetails].
[Endtime]-Measures.[CallDetails].[Starttime]
SELECT Measures.[UserDetails].MEMBERS ON COLUMNS,
Measures.[Talktime] ON ROWS FROM CallRecords
```

   Note that this query returns usage details for all users. This can also be restricted to get the details for a specific user

3. **ABC Mobile Service Provider wants to retain customers and also figure out if they can cross sell and upsell to existing customers**

   Depending on the case, it may be useful to try and cross sell to customers who have more than one account, since they may have a need for multiple services and are also inclined to purchase them from the same company

```
SELECT Measures.[AccountDetails].[AccountID], Measures.
[UserDetails].[UserID] ON COLUMNS FROM CallRecords
```

4. **ABC Mobile Service Provider wants to know their best clients so that they can offer better services to them**

   One heuristic for deciding the best customers could be to identify the customers with most talk time spent each month. These customers form the most active part of the user base

   ```
   WITH Measures.[Talktime] AS Measures.[CallDetails].
   [Endtime]-Measures.[CallDetails].[Starttime]
   SELECT Measures.[Talktime] ON COLUMNS, ORDER(Measures.
   [UserDetails].MEMBERS, Measures.[Talktime], DESC) ON ROWS
   FROM CallRecords
   ```

   This sorts users in descending order of talk time, with the highest talk times being on top

5. **ABC Mobile Service Provider wants to find out geographical areas where they need to concentrate more to improve service quality**

   To identify regions with low network coverage, the geographic data of the origin and destination towers for each call, coupled with data on whether the call was successful or not, can be used to identify regions with high ratio of dropped calls. These regions require focus to improve quality of service

   ```
   SELECT Measures.[CallDetails].[CallDropped] ON COLUMNS,
   ORDER(Measures.[TowerDetails].[Location], Measures.
   [CallDetails].[CallDropped], DESC) ON ROWS FROM
   CallRecords
   ```

   This returns the locations with highest number of dropped calls on top of the list