Expt1.

Write problem definition:

Suppose that a data warehouse consists of the three dimensions *time, doctor*, and *patient*, and the two measures *count* and *charge*, where *charge* is the fee that a doctor charges a patient for a visit.

Dimension Modeling Diagram

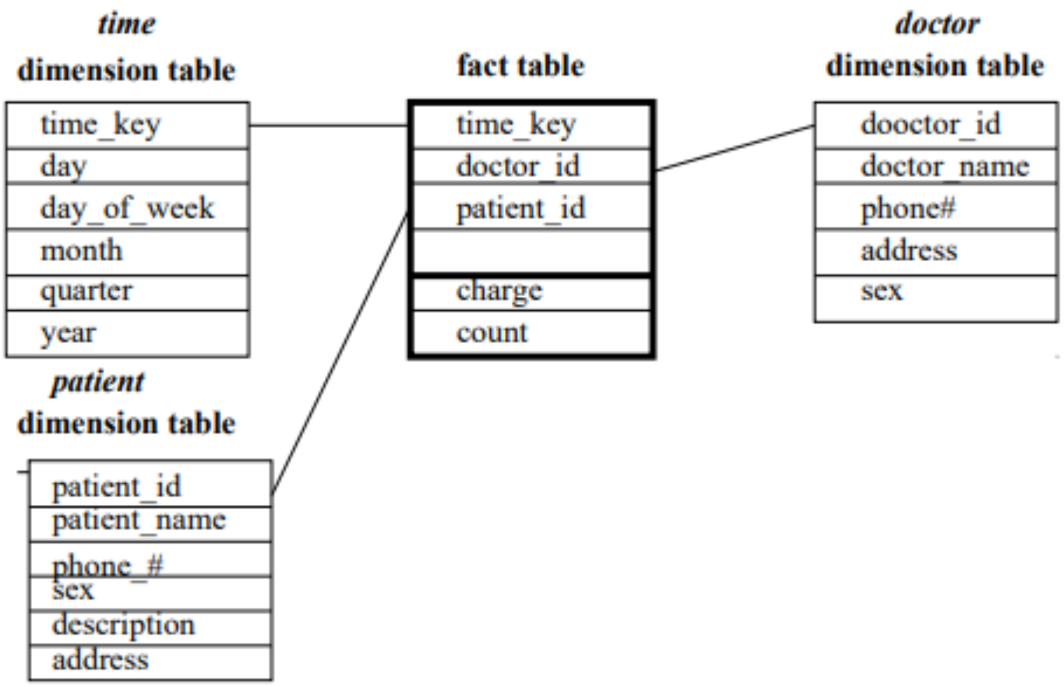

Figure 4.1: A star schema for data warehouse of Exercise 4.3.

DWM POSTLAB 1

Q1] Justify whether slowly changing dimension modelling is required for a problem selected?

Slowly Changing Dimensions (SCD) are the most commonly used advanced dimensional technique used in dimensional data warehouses. Slowly changing dimensions are used when you wish to capture the changing data within the dimension over time. There are three methodologies for slowly changing dimensions.

Type 1

For this type of slowly changing dimension you simply overwrite the existing data values with new data values. This makes the updating of the dimension easy and limits the growth of the dimension table to only new records. The drawback of this is you lose the historical value of the data because the dimension will always contain the current values for each attribute. For instance, you have a store dimension which has an attribute for a geographic region. If there is a redesign in the regional boundaries some stores may move from one region to another. This change will skew the historical reports and the reports run before the update will no longer match the reports run after the update for the same timeframe.

Type 2

This is the most commonly used type of slowly changing dimension. For this type of slowly changing dimension, add a new record encompassing the change and mark the old record as inactive. This allows the fact table to continue to use the old version of the data for historical reporting purposes leaving the changed data in the new record to only impact the fact data from that point forward. Several columns should be added to the dimension table (active record start/end dates and a current active record flag) to provide historical change management and ensure optimal use of the active record.

Type 3

This is a seldom used type of slowly changing dimension. In this type of slowly changing dimension you add a second column to store the most recent past value of the column(s) you wish to be able to report on. When the data is updated the existing value is "moved" to the column defined to store the previous past value and the new value is placed into the reportable column.

Conclusion

As you can see there are many ways to capture the changes in dimensions for current and historical reporting purposes. Because of the flexibility we recommend that you start all design solutions using SCD Type 2 as your default solution. Keep in mind that not all attributes need to be a captured as a Type 2 SCD.

Q2] Justify why fact-less fact table modelling is not required for the problem selected? Give the examples of fact less fact tables with its types

A2] A Data Warehouse fact-less fact table is a fact that does not have any measures stored in it. This table will only contain keys from different dimension tables. The fact-less fact is often used to resolve a many-to-many cardinality issue.

There are two types of fact-less fact tables

Event capturing fact-less fact

This type of fact table establishes the relationship among the various dimension members from various dimension tables without any measured value. For example, relation table capturing table is a fact-less fact table . Table will have entry into it whenever patient visits a doctor.
Following questions can be answered by the fact table:

1. Which doctor has maximum patients?
2. Which patients visit maximum doctors?
3. Which doctor earns the most?

All the above queries are based on the COUNT (), MAX () with GROUP BY.

Coverage table – Describing condition

This is another kind of fact-less fact. A fact-less-fact table can only answer 'optimistic' queries (positive query) but cannot answer a negative query. Coverage fact is used to support negative analysis reports.
For example, a doctor did not see any patients for given period of time.

If you consider the relation table, the event capturing fact table cannot answer 'which doctor did not see any patient?' Coverage fact attempts to answer this question by adding extra flag 0 for negative condition and 1 for positive condition.

Q3] Justify your approach to deal with large dimension tables for the problem selected

A3] An efficient data cube structure to support this preference would be to use partial materialisation, or selected computation of cuboids. By computing only, the proper subset of the whole set of possible cuboids, the total amount of storage space required would be minimised while maintaining a fast response time and avoiding redundant computation.
Since the user may want to drill through the cube for only one or two dimensions, this feature could be supported by computing the required cuboids on the fly. Since the user may only need this feature infrequently, the time required for computing aggregates on those one or two dimensions on the fly should be acceptable.
For each technique, explain how each of the following functions may be implemented:

i. The generation of a data warehouse (including aggregation)

ROLAP: Using a ROLAP server, the generation of a data warehouse can be implemented by a relational or extended-relational DBMS using summary fact tables. The fact tables can store aggregated data and the data at the abstraction levels indicated by the join keys in the schema for the given data cube.

MOLAP: In generating a data warehouse, the MOLAP technique uses multidimensional array structures to store data and multiway array aggregation to compute the data cubes.

HOLAP: The HOLAP technique typically uses a relational database to store the data and some low level aggregations, and then uses a MOLAP to store higher-level aggregations.

ii. Roll-up

ROLAP: To roll-up on a dimension using the summary fact table, we look for the record in the table that contains a generalisation on the desired dimension. For example, to roll-up the date
dimension from day to month, select the record for which the day field contains the special value all. The value of the measure field, dollars sold, for example, given in this record will contain the subtotal for the desired roll-up.

MOLAP: To perform a roll-up in a data cube, simply climb up the concept hierarchy for the desired dimension. For example, one could roll-up on the location dimension from city to country, which is more general.

HOLAP: The roll-up using the HOLAP technique will be similar to either ROLAP or MOLAP,
depending on the techniques used in the implementation of the corresponding dimensions.

Drill-down

ROLAP: To drill-down on a dimension using the summary fact table, we look for the record in the table that contains a generalisation on the desired dimension. For example, to drill-down on
the location dimension from country to province or state, select the record for which only the next
lowest field in the concept hierarchy for location contains the special value all. In this case, the city
field should contain the value all. The value of the measure field, dollars sold, for example, given in this record will contain the subtotal for the desired drill-down.

MOLAP: To perform a drill-down in a data cube, simply step down the concept hierarchy for the desired dimension. For example, one could drill-down on the date dimension from month to day in order to group the data by day rather than by month.

HOLAP: The drill-down using the HOLAP technique is similar either to ROLAP or MOLAP
depending on the techniques used in the implementation of the corresponding dimensions

Q4] Justify the use of junk dimension and surrogate key for the problem selected.
A4.] In data warehouse design, frequently we run into a situation where there are yes/no indicator fields in the source system. If we keep all those indicator fields in the fact table, not only do we need to build many small dimension tables, but the amount of information stored in the fact table also increases tremendously, leading to possible performance and management issues.

JUNK DIMENSION

Junk dimension is the way to solve this problem. In a junk dimension, we combine these indicator fields into a single dimension. Junk dimensions are used to reduce the number of dimensions in the dimensional model and reduce the number of columns in the fact table. A junk dimension combines two or more related low cardinality flags into a single dimension.

As you can see these are limited in number and, if created as single dimensions, the dimensions would be limited to a single attribute. In order to eliminate these small dimensions, we create a single "junk" dimension which cross joins all possible attributes into a single dimension which will be used in the fact table. These two types of dimensions are useful and powerful in creating better to use and understand data models.

SURROGATE KEY

A surrogate key is a key which does not have any contextual or business meaning. It is manufactured "artificially" and only for the purposes of data analysis. The most frequently used version of a surrogate key is an increasing sequential integer or "counter" value (i.e. 1, 2, 3). Surrogate keys can also include the current system date/time stamp, or a random alphanumeric string.

The Uses of Surrogate Keys are as per the following

- Surrogate keys are unique. Because surrogate keys are system-generated, it is impossible for the system to create and store a duplicate value.
- Surrogate keys apply uniform rules to all records. The surrogate key value is the result of a program, which creates the system-generated value. Any key created as a result of a program will apply uniform rules for each record.
- Surrogate keys stand the test of time. Because surrogate keys lack any context or business meaning, there will be no need to change the key in the future.
- Surrogate keys allow for unlimited values. Sequential, timestamp, and random keys have no practical limits to unique combinations.