Data Warehousing Assignment

This problem set consists of two data modeling scenarios. You will be asked to analyze the strengths and weak nesses of some design alternatives for each scenario. Short answers are fine – one or two paragraphs per question would be an appropriate length.

Scenario I In this scenario, we are interested in modeling student enrollment in Stanford courses. We would like to answer questions such as:

• Which courses are most popular? Which instructors are most popular?

• Which courses are most popular among graduate students? Undergraduates? • Are there courses for which the assigned classrooms is too large or too small?

We are planning to have a course enrollment fact table with the grain of one row per student per course enrollment. In other words, if a student enrolls in 5 courses there will be 5 rows for that student in the fact table. We will use the following dimensions: Course, Department, Student, Term, Classroom, and Instructor. There will be a single fact measurement column, EnrollmentCount. Its value will always be equal to 1.

We are considering several options for dealing with the Instructor dimension. Interesting attributes of instructors include FirstName, LastName, Title (e.g. Assistant Professor), Department, and TenuredFlag. The difficulty is that a few courses (less than 5%) have multiple instructors. Thus it appears we cannot include the Instructor dimension in the fact table because it doesn't match the intended grain. Here are the options under consideration:

OptionA

Option B

Option C

ModifytheInstructordimensionbyaddingspecialrowsrepresentinginstructorteams.Forexample,CS276ais taught by Manning and Raghavan, so there will be an Instructor row representing "Manning/Raghavan" (as well as separate rows for Manning and Raghavan, assuming that they sometimes teach courses as sole instructors). In this way, the Instructor dimension becomes true to the grain and we can include it in the fact table.

Change the grain of the fact table to be one row per student enrollment per course per instructor. For example, there will be two fact rows for each student enrolled in CS 276a, one that points to Manning as an instructor and one that points to Raghavan. However, each of the two rows will have a value of 0.5 in the EnrollmentCount field instead of a value of 1, in order to allow the fact to aggregate properly. (Enrollments are "allocated" equally among the multiple instructors.)

Create two fact tables. The first has the grain of one row per student enrollment per course and doesn't include the Instructor dimension. The second has the grain of one row per student enrollment per course per instructor and includes the Instructor dimension (as well as all the other dimensions). Unlike Option B, the value of EnrollmentCount will be 1 for all rows in the second fact. Tell warehouse users to use the second fact table for queries involving attributes of the instructor dimension and the first fact table for all other queries.

Please answer the following questions.

Question 1. What are the strengths and weaknesses of each option?

Answer1: Case A:

Strength:

- Adding simple team row is good for read operations as it will make dimensions also quite simple and clear and If we have a very little portion of multiple instructors, it will be the best approach

Weakness:

- When needed to use aggregation we will not be able to track multiple instructors a no values have been specified for team division & they can not be directly used in aggregation with [instructor1/instructor2] syntax.

Case B:

Strength:

- Though Processing will be slow but calculator of a lot of questions will become easier for courses where multiple instructors teach and If data does't have a high fraction of these multiple instructors processing would not be affected to big extent.

Weakness:

- Creating separate row for each instructor will increase the complexity a lot, which will result in high latency.As we already have separate row for rach student enrolled in different course.So if a course taugh by multiple instructors also gets multiple rows,Number of rows will increase exponentially as well doing aggregation functions will be quite difficult
- And the biggest problems if number of instructor of a particular course will be in odd numbers like 3 then equal division will not be possible at any cost a each instructor if takes values 0.33 *3 =0.99 sum will not be 1

Case C:

Strength:

- The problem of equal division in caseB will be sorted here & the complexity will also reduce.
- Queries which require only instructor table or only student table will be fast.

Weakness:

- Queries where both Instructors & Courses will be required we will use join operations which in heavy data analytics has high latency.

Question 2. Which option would you choose and why?

For the given scenario where Multiple Instructors are only 5% which is not a lot so I will prefer Case B as

number of rows will not increase that much.So latency effect would be negligible

While in case of CaseC two separate fact tables fro just 5% wound't be reasonable as using join operation would be more hectic here than increasing number of rows.

Question 3. Would your answer to Question 2 be different if the majority of classes had multiple instructors? How about if only one or two classes had multiple instructors? (Explain your answer.)

Yes, My answer to Question 2 would be different if majority classes had multiple instructors than in that case Number of rows will increase a lot so latency would be affected and queries will be very complex, so at that time I would prefer to chose CaseC as creating two different columns will make my querying job easier.

In case where whe would have only one or two classes for multiple instructors I would prefer CaseA as one tow classes could be handled manually so we dont need to increase number of rows or create two separate fact tables.

Question 4. [OPTIONAL] Can you think of another reasonable alternative design besides Options A, B, and C? If so, what are the advantages and disadvantages of your alternative design?

- I would like to try creating two fact tables and they would be courses and students as separate row for a student for each course creates a lot number of extra rows and updating them would also be difficult though updating is not a case of Data Warehousing but these separate fact tables will solve the issue a lot
- And Also I would use the approach of caseB as well to create separate rows for multiple instructors so this would increase the number of rows but I have also decreased them by separating data in two fact tables So complex queries would also become easier
- The latency which was getting increased by CaseB approach is taken by two fact tables courses and students as using join operations on less rows is fast than A lot more number of rows without two fact tables.

Scenario II In this scenario, we are building a data warehouse for an online brokerage company. The company makes money by charging commissions when customers buy and sell stocks. We are planning to have a Trades fact table with the grain of one row per stock trade. We will use the following dimensions: Date, Customer, Account, Security (i.e. which stock was traded), and TradeType.

The company's data analysts have told us that they have developed two customer scoring techniques that are used extensively in their analyses.

· Each customer is placed into one of nine Customer Activity Segments based on their frequency of transactions, average transaction size, and recency of transactions.

· EachcustomerisassignedaCustomerProfitabilityScorebasedontheprofitsearnedasaresultofthatcustomer's trades. The score can be either 1,2,3,4, or 5, with 5 being the most profitable.

These two scores are frequently used as filters or grouping attributes in queries. For example:

· How many trades were placed in July by customers in each customer activity segment?

· What was the total commission earned in each quarter of 2003 on trades of IBM stock by customers with a profitability score of 4 or 5?

There are a total of 100,000 customers, and scores are recalculated every three months. The activity level or profitability level of some customers changes over time, and users are very interested in understanding how and why this occurs.

We are considering several options for dealing with the customer scores:

OptionA Option B Option C

Option D

ThescoresareattributesoftheCustomerdimension.Whenscoreschange,theoldscoreisoverwrittenwiththe new score (Type 1 Slowly Changing Dimension).
The scores are attributes of the Customer dimension. When scores change, new Customer dimension rows are created using the updated scores (Type 2 Slowly Changing Dimension).

The scores are stored in a separate CustomerScores dimension which contains 45 rows, one for each combi- nation of activity and profitability scores. The Trades fact table includes a foreign key to the CustomerScores dimension.

The scores are stored in a CustomerScores outrigger table which contains 45 rows. The Customer dimension includes a foreign key to the outrigger table (but the fact table does not). When scores change, the foreign key column in the Customer table is updated to point to the correct outrigger row.

Please answer the following questions.

Question 5. What are the strengths and weaknesses of each option?

OptionA:

Strength:

- Querying will be quite faster,reading and writing data also as we dont need to store data of previous scores

Weakness:

- Previous data could be used to track customer journey in a more better way

OptionB:

Strength:

-

Weakness:

- Read and write speed will be decreased as we are storing previous data also in the same column as separate rowPrivous data is also tracked which can be used to define customer journey and get more better customer

analytics.

OptionC:

Strength:

- Creating separate dimensiontable will solve the problem of OptionB where latency was the issue

Weakness

- But queries wherae both previous data and present data would be required queries will become quite complex

And can affect latency again

OpionD:

- Not able to find

Question 6. Which option would you choose and why?

I will Choose option C as we will get both previous record and current record as well here score recalculation is done at a large time so doing this job wont require a big separate dimension table and using join operations will also be not that much time consuming.

Question 7. Would your answer to Question 6 be different if the number of customers and/or the time interval between score recalculations was much larger or much smaller? (Explain your answer.)

Yes,if score calculation would be more frequent then OptionC wont work as those 45 rows would get filled quite early and we wont be able to track customer history for a long time

So I would use OptionB as I have to give up on long term history of customer as records are getting more frequently so using different table with just 45 rows wont be efficient instead just adding a single row in the same fact table the approach of OPtionB would be able to manage these frequent updations

If Score updation would take more time then also I would chose option B as creating separate table for stong just few history scores would not be efficient at the cost of latency which is affected by join operation while Option B here will do the work as score updation take long time so adding more rows in the fact table will help us track the previous records also and since score calculation takes long time rows would also be less and no separate dimension table means no joins s latency will also be decreased.

Question 8. [OPTIONAL] Can you think of another reasonable alternative design besides Options A, B, C, and D? If so, what are the advantages and disadvantages of your alternative design?