**Data Warehousing Assignment**

This problem set consists of two data modeling scenarios. You will be asked to analyze the strengths and weakOnesses 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:

**Option A**

Modify the Instructor dimension by adding special rows representing instructor teams.For example,CS276a is 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.

**Option B**

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.)

**Option C**

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?**

Option A: Modify the Instructor dimension by adding special rows representing instructor teams.

**Strengths:**

• The Instructor dimension matches the intended grain of the fact table.

• Allows capturing instructor teams for courses with multiple instructors.

• Maintains the integrity and consistency of the data model.

**Weaknesses:**

• May introduce additional complexity in the Instructor dimension.

• Requires identifying and adding special rows for instructor teams.

• May not be suitable if the number of instructor teams is large or dynamic.

Option B: Change the grain of the fact table to be one row per student enrollment per course per instructor.

**Strengths:**

• Accommodates courses with multiple instructors without modifying the Instructor dimension.

• Allows for accurate allocation of enrollments among instructors.

• Preserves the original grain of the fact table.

**Weaknesses:**

• Requires adjusting the EnrollmentCount field to fractional values.

• Aggregation and analysis of the EnrollmentCount may need to consider the fractional values.

Option C: Create two fact tables, one without the Instructor dimension and another with the Instructor dimension.

**Strengths:**

• Provides separate fact tables optimized for different types of queries.

• Allows efficient analysis of attributes related to the instructor dimension.

• Maintains a clean and consistent grain in each fact table.

**Weaknesses:**

• Requires managing and maintaining two separate fact tables.

• Increases the complexity of the data model.

• May require educating users about which fact table to use for different types of queries.

**Question 2. Which option would you choose and why?**

The choice of option depends on the specific requirements and priorities of the analysis. However, based on the given scenario and objectives, Option C seems like a reasonable choice. It provides the flexibility of having two separate fact tables optimized for different types of queries. The first fact table can be used for general enrollment analysis, such as popular courses and class size evaluation, while the second fact table can be used for instructor-specific analysis, such as popularity and performance. This approach allows for efficient querying and avoids the complexities introduced by modifying the Instructor dimension or adjusting fractional values in the EnrollmentCount field.

**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.)**

If the majority of classes had multiple instructors, Option B could be more suitable. Adjusting the fact table's grain to accommodate multiple instructors per course would allow for accurate representation of enrollment allocation among instructors. This would provide better granularity and accuracy in analyzing enrollments for courses with multiple instructors.

If only one or two classes had multiple instructors while the majority had a single instructor, Option A or Option C could still be viable. Option A would be more suitable if the instructor teams are well-defined and consistent. Option C could be a better choice if the number of classes with multiple instructors is relatively small and it's not necessary to optimize the fact table for instructor-specific analysis for these specific classes.

**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?**

One alternative design could be to create a separate InstructorTeam dimension that captures instructor teams for courses with multiple instructors. This dimension would be linked to the Instructor dimension and the fact table through appropriate relationships. The advantage of this design is that it allows for explicit representation and analysis of instructor teams while maintaining the original grain of the fact table. However, it adds complexity by introducing an additional dimension and requires managing the relationships between dimensions and the fact table accurately. The choice of this alternative design would depend on the specific requirements and the importance of analyzing instructor teams separately.

**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.

· Each customer is assigned a Customer Profitability Score based on the profits earned as a result of that customer’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**

The scores are attributes of the Customer dimension. When scores change ,the old score is overwritten with the new score (Type 1 Slowly Changing Dimension).

**Strengths:**

* Simple to implement
* Does not require additional storage space

**Weaknesses**:

* Can lead to data loss if the old score is overwritten with an incorrect value
* Can make it difficult to track changes in customer scores over time

**OptionB**

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).

**Strengths:**

* Tracks changes in customer scores over time
* Does not require additional storage space

**Weaknesses:**

* Can lead to duplicate rows in the Customer dimension
* Can make it difficult to join the Customer dimension to the Trades fact table

**OptionC**

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.

**Strengths**:

* Tracks changes in customer scores over time
* Does not require duplicate rows in the Customer dimension
* Can easily join the CustomerScores dimension to the Trades fact table

**Weaknesses**:

* Requires additional storage space

**OptionD**

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.

**Strengths**:

* Tracks changes in customer scores over time
* Does not require duplicate rows in the Customer dimension
* Does not require additional storage space
* Can easily join the CustomerScores outrigger table to the Trades fact table

**Weaknesses**:

* Requires an additional table

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

As mentioned above.

**Question 6. Which option would you choose and why?**

I would choose Option D. It tracks changes in customer scores over time, does not require duplicate rows in the Customer dimension, and does not require additional storage space. It also allows for easy joining of the CustomerScores outrigger table to the Trades fact table.

**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.)**

If the number of customers was much larger, I would still choose Option D. However, if the time interval between score recalculations was much larger, I might choose Option A. This is because Option A is simpler to implement and does not require additional storage space.

**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?**

One alternative design would be to create a separate CustomerScores table for each time period. This would allow for more granular tracking of changes in customer scores over time. However, it would also require more storage space.

Another alternative design would be to use a hybrid approach. For example, Option A could be used for the most recent time period, and Option D could be used for older time periods. This would allow for a balance between simplicity and flexibility.