Dear leader:

Recently I'm working with data related to Receipts, and I want to know the following questions. The first one is the total size of Receipts data, specifically how many pieces of data there are in total, and how many new pieces of data are added every day. Knowing this information will help me analyze the impact of issues in the data on data quality. (What questions do you have about the data?)

I mainly discover data quality issues by normalizing the data. In the process of data normalization, problems such as insertion exceptions and update exceptions can be eliminated. For example, the data of RewardReceiptItem in Receipts is extracted and used as a separate table. In the Receipt table, only Save a RewardReceiptItem's primary key as an index to reduce data duplication. (How did you discover the data quality issues?)

But before deciding on this solution, what needs to be understood is the requirements of the system when querying and using data. If all the fields of the RewardReceiptItem table are required for each query, then although some fields are separated to save storage space, but in Each time you query, you need to connect multiple tables, and the overhead here is even greater. (What do you need to know to resolve the data quality issues?)

When trying to create data assets, the other information you need to know is which attributes are required when analyzing data or displaying data on the front end, and the possible length of these fields. This information helps me to set the attributes of the field when creating the form, such as whether it can be NULL, if it is a character, it is more appropriate to choose varchar or char to set the length. (What other information would you need to help you optimize the data assets you're trying to create?)

In production, if the amount of data is too large, the query waiting time may be too long. If you must use a relational database, you must consider using an index to improve the query speed. If you use a non-relational database, such as mongodb Etc., can also improve the speed of the query. In addition, the concurrency capability of the database is also a test. If the pressure of concurrent access is high, it is recommended to use a non-relational database for storage, because non-relational databases can better support distributed deployment. (What performance and scaling concerns do you anticipate in production and how do you plan to address them?)

Best,

JINGYA WANG