

Errors and Events.

Dr. Tom Johnston
MindfulData.com
August, 2004.

Correction vs. Compensation.

There are only two ways to deal with errors. We can either correct them, or we can compensate for them. Let us say that these are the two ways in which we can "fix" errors.

Consider transactions which are entered during the day, but which may later be found to have been incorrect. An example are the transactions recording the amount of time consumed during an operation in a manufacturing process, or the amount of material consumed during such an operation.

In the case of a manufacturing company I know of, thousands of such transactions are generated every day, and perhaps 10 percent of them will be found to have been in error. Many of these errors are fixed within a day or two. However, the process of fixing errors in transactions can continue until the close of the current accounting month.

In general, accountants prefer to compensate for errors in transactions rather than to correct those transactions. The reason for this is easy to see. If we correct transactions, the record of their incorrect entry is erased. But if we create compensating transactions, then we leave a complete audit trail. This means that compensating for erroneous transactions leaves us with more information than correcting those transactions. It retains the information about the period of time during which the incorrect data was being provided. It also retains the information about what the incorrect data was.

An example of the need to retain this kind of information is when the transactions provide data which is reported to an outside party. For example, if the transaction incorrectly cancels a clause of an insurance policy effective on a certain date, when it should have cancelled that clause of the policy ten days later, then we may not need to simply correct the error, but also keep track of the period of time during which the clause was reported as having been canceled on the earlier date.

However, when such special requirements are not relevant, it sometimes is more straightforward to simply correct an erroneous transaction, particularly when there are no business requirements to reconstruct the chronological history of the transaction.

In many cases, a mixed strategy is appropriate. For a brief period of time, transactions are regarded as unaudited and eligible to be corrected, i.e. to be updated with data that replaces the incorrect data. However, after that period of time, transactions are regarded

as audited. Any errors detected in audited transactions must be compensated for with offsetting transactions.

Some Important Distinctions.

We need to distinguish between changing transactions, and changing non-transactional data. A transaction is the record of an event or a duration. Once the event has occurred, or the duration has ended, that temporal object can no longer change. Therefore any change to the transactions which represent them are changes made to correct errors in those transactions. They are not changes that reflect changes in the real world objects to which the transactions correspond.

Consider, by contrast, a change to a row in a customer table, in which we change customer Sally Smith's last name to Jones. This change might be to correct an error in data entry, but let us suppose that the change reflects a change in Sally's name due to the fact that Sally got married. Let us call this kind of change to a row of a table an update. By contrast let us call a change to a transaction a correction. So any kind of change to a row of any table which does not correspond to a change in the real world object which that row describes is a correction. Since transactions describe real world objects which do not change, the only changes we can make to transactions are corrections.

There is also an ambiguity in the word transaction itself. On the one hand, it is used to refer to any add, change, or delete against any table in the database. On the other hand, it is used to refer to a row in a transaction table. Let us examine the difference between these two uses of the term transaction.

In the first case, the transaction is the action of adding a row to a table, updating a row in a table, or deleting a row from a table. In the second case the transaction is a row in a special kind of table, a table whose instances represent on changing objects, which are either events or durations.

In the first case, the transaction record which is used for the update is not preserved. In the latter case, the transaction record which is used for the update is preserved. Transactions which update tables are preserved when it is important to preserve a history of the changes made to those tables. Usually when this is the case, the tables being updated are balances. An archetypal example is an account table in which the transactions debit or credit the account.

Note that transactions, in the sense of rows in a transaction table, are also updates. However they are not updates which add or delete a row to another table, at least not usually. In general they are updates to another special kind of table which I will call a balance table. (Not always. A table of meter readings is a table of transactions, but they don't necessarily update any other table.)

To distinguish between these two uses of the word transaction, we will refer to the first kind of transaction as a nonrecorded transaction and to the second kind of transaction as a recorded transaction. However, we will often let context make the distinction without using the adjective.

Ontologies.

The work we're doing here is ontology, in both the classical and the computer science sense of the term. Historically, ontologies have been developed by people interested in such things by simply thinking as deeply as they can about what the basic categories of things are. In our case, however, we are attempting to build an ontology by classifying the different kinds of structures in which we store data. These different kinds of structures have evolved over the last half a century in which computers have been used. They thus reflect the information needs of their myriad user communities. An ontology based on this study is more than an idiosyncratic attempt to understand the most basic categories of what there is. It is an attempt based on a wide range of human experience.

Finally, there is one note I want to make about Quine's criterion of ontological commitment. This criterion I regard as the bridge between classical ontology and computer science ontology. The criterion is that to be is to be the value of a variable. In computer science ontology, this means that to be is to be the object of an SQL statement. There is an irony in the application of Quine's criterion to computer science ontology, and the irony is this. Quine himself has strongly nominalistic tendencies. Yet the application of this criterion to computer science ontology commits us to the existence, as far as a particular database is concerned, of whatever we take each of the tables of that database to represent. This, obviously, is a hyper-realist ontology.