

An Ontology of Tables: Part 2.

Dr. Tom Johnston
MindfulData.com

Now let's see how the ontological categories defined in Part 1 are reflected in different types of database tables.

Particulars, Individuals and Persistent Object Tables.

Persistent object tables are tables which represent particulars. Each of their rows represents an individual. Some typical examples include:

- Customers
- Salespersons
- Invoices
- Invoice Line Items
- Organizations
- Zip Codes

Some examples of tables which do not represent particulars include:

- Accounts Receivable
- Customer Status Codes
- A Customer Salesperson cross-reference (associative) table.

At the end of this essay, it should be clear why the latter three tables are not persistent object tables. In case it isn't, I'll briefly explain why.

Universals, Concepts and Reference Tables.

Reference tables are another type of table. They are sometimes called “lookup tables”, or “code tables” or even “domain tables”. In general, each row of a reference table has a code and a description of what it means. Sometimes it has an abbreviation instead of a code, and a full name instead of a description.

Color, and its instances red, blue and green, would be represented in a database as a reference table with three rows. Customer status code, and its instances VIP, in good standing, standard, poor and high risk, would be represented in a database as a reference table with five rows.

Instances of reference tables exist as properties (or relationships) of instances of things, either of individuals (instances of particulars) or of concepts (instances of universals).¹

In contrast to particulars, universals don’t exist through time, and so cannot change. Color, for example, doesn’t exist through time, although there certainly were colored things in the past, are colored things today, and will be colored things in the future. But things exist in time in the sense that they exist in their own right, and except for eternal things (if there are any), their instances begin to exist at a point in time, and cease to exist at a later point in time. This isn’t true of the instances of color, or of customer status, or of any other universals.

Here’s another way to look at the unchangeability of universals. Certainly the words we use to designate universals often change their meaning over time. “Naughtiness”, in the King James bible, meant “evil”. These days, it means something much more frivolous. But if we are willing to distinguish between universals and their representation in language, we can say that the universals did not change, but our labels for them did.

¹ Note that an individual can exist only once, while a concept can exist as many times as there are things that “instantiate” it. This is another important difference between particulars and universals.

So suppose we have a reference table of Customer Status Codes, with one row in that table being the code “VIP”. And let’s say that the business decides, on a certain date, to change the definition of “VIP” customer from “customer who has made over \$10,000 of invoiced purchases in the last twelve months” to “customer who has made over \$10,000 of invoiced purchases in the last twelve months and whose credit score is above 700”. Hasn’t the concept of VIP customer changed?

We could indeed say that. And if we did, we would be using the term “change” in a wider sense than I have been using it. There is nothing necessarily wrong with doing that. But I am creating a set of distinctions by using “change” to mean “change in one or more properties or relationships of particulars”.

This, of course, makes a tautology of the statement that universals do not change. My justification for using “change” in the sense I am using it is that it helps me explain my ontological categories in a way, hopefully, which is clear enough to allow others to use those categories in a way consistent with my intended use for them. Another justification is that I think that if we did extend the term “change” to include change in universals, we would have to admit that, in a sense admittedly difficult to explain, these are two different senses of the term “change”. In other words, I think that my use of the term “cuts nature at the joints”, i.e. is somehow natural rather than artificial.

Event Types, Events and Transaction Tables.

Transaction tables are a third type of table. Each transaction in a transaction table represents one event. The table indicates the type of event in question.

For example, sales are events. This kind of event usually has four main participants – buyer, seller, what was sold, and what was given in exchange. In non-barter sales transactions, goods or services are sold, and what is given in exchange is money, or the

promise to pay money at a later date (i.e. credit). All four participants in this type of event are particulars. All four participants in each instance of this type of event are instances of those particulars, i.e. individuals.

Unlike particulars, but like universals, events can't change because they don't endure through time. Events *happen*, and then they are over. For this reason, the only reason to update a transaction – the representation in a database table of an event – is to correct a mistake. A transaction is never updated because the event it represents changed. Again, events don't change. They happen, and then they are over with.

Particulars, by contrast, do change. So we might update a row in a Customer table, for example, because the customer represented by that row changed his address. We might also update a row in a Customer table because the original data entry was in error. But for transactions, only the latter reason can justify an update.²

Transactions may involve only one thing. For example, hospital staff may take a patient's temperature twice a day, and record it on a chart. The chart is a transaction table, and each entry is a transaction. But the patient is the only thing affected by the transaction. In these cases, transactions record the change of a property of a thing, in this case the temperature of a patient.

But many transactions involve two or more things, and what they reflect is a change in the relationship among those things. These kinds of transactions have a cumulative effect on that relationship, and the current state of that cumulative effect is recorded as a balance.

² Note that I talk about updating a transaction, not about changing it. I do this because I am trying to draw parallels between ontology and databases. So to keep things clear, “change” will be an ontological term, referring to what happens in the world, while “update” will be a database term, referring to what happens in a database as a reflection of what happens in the world.

A Special Relationship: Balance Tables.

An inventory receipt, for example, increases the on-hand quantity for the type of product received. It also increases the total accounts payable for the receiving company. It is the equivalent of a sale, only one in which the company is the customer instead of the seller.

The transactions that most businesses are interested in are those that affect relationships that have quantitative measures. A payment is received, an event. The relationship between the payer and payee is altered by the amount of the payment. That relationship is recorded in a revolving credit balance record, or else an accounts receivable balance record. The payment is recorded as a credit, and the balance due is decreased by that amount.

These records are called balance records because, like a see-saw, the relationship they affect swings now one way, now back the other way. Each change to the relationship is triggered by a transaction, and the net effect of those changes is the current balance of the relationship. The transactions are so important, however, that we keep track of them, and usually provide the ability to “drill down” into those transactions to get to a history of how the current state of the relationship came about.

Balances aren't the only kind of relationship. There are other kinds. A Customer / Salesperson Cross-Reference table – an associative table, in relational terms – represents a relationship between customers and salespersons. This table might represent a business rule which states which salespersons are authorized to sell to which customers. This table can be updated by a transaction, but those updates are not important enough that we keep track of those transactions, or provide the ability to “drill down” into those transactions from the relationship.

To summarize: companies are all about ongoing relationships. Those relationships are affected by events, which are recorded as transactions. Rather than adding up all transactions that affect a relationship each time we want to see the current state of that

relationship, we keep a current state of the relationship record – a row in a balance table. Financial account tables are balance tables; each account number uniquely identifies a particular relationship, and the metrical properties of that account tell us the current status of the relationship.

Four Types of Tables.

So derived top-down from my background in the philosophical study of ontology, and derived bottom-up from my decades' long experience with business databases, I have aligned one with the other to come up with the following types of tables, and their correlation with ontological categories.

- **Persistent object tables.** These tables represent particulars. Their rows represent individuals. The principle of the identify of indiscernibles, and the distinction between essential and accidental properties, are reflected in the entity integrity constraint, that no two rows of such tables are identical, either in total, or in their primary key columns.
- **Reference tables.** These tables represent universals. Their rows represent concepts. The identity of indiscernibles and the essential vs. accidental property distinction apply here as well. If we think of reference tables as consisting of a code and a description for that code, the code is the essential property. But a change in the description, the non-essential property, is not a change which leaves the concept alone. It is a change which introduces a new concept that replaces the old one by “taking over” the code.
- **Transaction tables.** These tables represent types of event. Their rows represent specific events of that type. Events don't change. They happen, and then they are done with. Because of this, transactions are updated only if they were entered in

error. They are never updated because what they describe has changed. Again, events don't change.

- **Balance tables.** These tables represent relationships whose state is altered by a series of transactions and which are important enough to be persisted. Practical considerations aside, any balance could be recreated, as needed, “on the fly”, by summing up all the transactions that affected that relationship. But balance tables describe the web of ever-changing relationships that define the internal and external state of the company as it has been affected by that company's internal and external interactions with other things.

Ontology: Does It Matter?

What I'm trying to do is bring a high-level pattern, from another discipline (in this case, ontology), to bear on the practice of designing sets of tables to efficiently and effectively record and return information. Because this pattern is so high-level in comparison to collections of tables in individual databases, the value is not easy to see.

One value is that it is a repeating pattern that we can use to “chunk” our experience. But from this perspective, it is no better (or worse) than the pattern of three types of tables we mentioned above or, more generally, no better or worse than any other applicable pattern.

But I think it's a valuable pattern in a much stronger sense than this. And I will talk about this additional value in a later essay on ontology, the semantic web, and the IEEE's 1600.1 Standard Upper Ontology workgroup.

This set of ontological categories is, as far as I know, unique. So also is this set of four types of database tables.

Wrap-Up.

In the language of ontology: particulars endure. They have properties, and relationships. Events happen, and in happening they affect particulars, both their properties and their relationships. Ongoing relationships often have a current state that is continually being affected by certain kinds of events. For those relationships, we often want to keep track of and review the events that impacted them.

In the language of databases: some tables represent persistent objects. The foreign keys those tables have represent their relationships, and the non-foreign key columns represent their attributes. Transactions take place which update the values of certain attributes. Other transactions take place which update the state of certain relationships. We generally don't keep transaction tables for the former kind of transactions. But for the latter kind of transactions, when the relationships they affect are important enough to be represented by an account, we want to know the current state of the account – its balance – and often want to be able to review the succession of transactions that generated that balance.