

Ontology: Things, Stuff and Events.

Dr. Tom Johnston
MindfulData.com
Written sometime in 2006.

{
This is the amalgamation of three sets of notes. As a result, there may be some inconsistencies in terminology. I'll revise this material later on, and clear up any such issues. But for now, I think this essay still has value, even if there are some terminology problems.

What is that value? It is the value theory always has. It is a framework for thinking about things which organizes our experience by revealing patterns among things and events (or, as other philosophers would have it, by imposing patterns on things and events).

The patterns that ontology is concerned with are deep patterns. This term, "deep patterns" is both a description and a compliment. As a description, deep patterns are distinguished from what we might call "surface patterns" in several ways.

- 1. Surface patterns are pretty obvious; deep patterns are not.*
- 2. Deep patterns constitute the theory of something of interest; surface patterns comprise the engineering through which the theory is applied.*
- 3. Familiarity with surface patterns supports treatment of a difficulty. But only familiarity with deep patterns will support diagnosis.*

This last distinction is familiar to we IT professionals. Consider a software application that has been around for a long time. Multiple releases; countless bug fixes between releases. Often, an IT shop will have one person who is the guru for that system. And what distinguishes the guru from the other development and support personnel is not how well or how fast he can write code, or even how elegant his code may be. It's that the guy (consider "guy" a gender-neutral term) has an uncanny ability to figure out what's wrong when the system crashes, or when it produces crazy results.

I think most of us with a solid background in systems development and maintenance would agree with this description. There may be lots of very smart programmers who have worked on the system. But there's something almost mystical about the guru, and almost magical about how often his diagnostic hunches are exactly right.

So what does the guru know that the other smart guys don't? He knows the system. The other guys know a bunch of programs. Even if they are familiar with all the programs, and with what each one does, they don't have that gut-level instinct that the guru does.

Instinct? It's got to be something more than that because it reveals a knowledge of a logical structure (however convoluted) – a collection of algorithms being applied to instances of a data structure. The guru's ability may look like instinct; it may even feel to him like instinct. But it's clearly got to be something more than instinct. This something more is a grasp of the deep structure of the software application. It is a grasp of the underlying theory of the system.

Why, then, can't the guru clearly explain, step by step, how he thought through the problem and arrived at the correct diagnosis? Because to do that is a different skill. It's called pedagogy. It's the skill that makes a good teacher, or a good author.

The theory is a pattern. Patterns aren't grasped piece by piece. You either see a pattern, as a single, complex thing, or you don't see it. The guru sees the pattern "in his mind's eye". In very rare circumstances, the guru can also explain what he sees, but an inability to explain it does not imply a diminished ability to see it clearly.

So: this is why theory is important. What does this have to do with ontology and IT? Just this: in my view, there are a sufficient number of speakers and authors around who will explain the surface patterns of ontology as it applies to the management of data. They will explain the engineering of it – RDF triples, OWL, and the intricacies of ontology management tools.

This engineering knowledge is all that you need to be the kind of smart programmer I talked about. But it will not make you a guru. For that, you need the deep structure. You need to recognize the patterns for which the engineering knowledge provides a toolkit of tools that manipulate those patterns. In the hands of a journeyman, if overseen by a master craftsman, these tools will produce beneficial results. But that is because, aware of the deep structure, the master craftsman decides what needs to be done – how to build the most beautiful construct that will solve whatever practical problem is at hand, or how to repair an existing construct buffeted and broken by unexpected storms, or how to integrally create a completely new addition to the building.

The way to become a master craftsman of the application of ontologies to data management, to become a guru, lies first in an understanding of ontology itself, a subject that reaches back almost three millennia, and whose gurus came, originally, from the Aegean islands (especially Miletus) or the Greek mainland (especially Athens).

I'm working on providing this historical introduction to ontology elsewhere on this website. Here, I take another approach – to jump in and start doing some ontology. I'll focus on what computer science ontologists would call an "upper ontology". I prefer the opposite metaphor, and call it a "foundational ontology".

}

Things and Events: a Foundational Ontology.

First, let's *do* some ontology.

- The world consists of **things, events** and **stuff**.
- **Things** endure over time, and because of that, they can change. They have properties and relationships, and these are what it is about things that change. Following Frege, logicians consolidate this distinction and speak of predicates. Properties are "one-place predicates", a relation between two things is a "two-place predicate", and so on. Things change because they take part in events, or, to put it another way, because events happen to them.
- **Events** are what, of interest to us, happens. Each event happens once, and then it is over. An event can be what happens all at once, at a given instant. Events can also be what happens over a given period of time. In either case, events do not change, even though they exist over time. Since, at any moment or duration of time, even at one place, a lot is happening, "of interest to us" is what picks out what makes up the event. But is it true that events do not change? What about, for example, all the changes going on at an event which is a two-hour rock concert? It is the things in or at that event that are changing, not the event itself.
- **Stuff** is what exists that can't be counted. As a result, nouns for stuff don't have a plural form. Boats are things; there are many of them. But we can't say "Water are things". We also can't say "Waters are things". And this is the way we handle stuff. We handle chunks of it, and chunks of stuff can be counted. Names of chunks of stuff do have a plural form. A lake is a thing. Also, lakes are things. A glass of water is a thing. Also, glasses of water are things. So while stuff isn't a thing, chunks of stuff are things.

Things.

{ See Part 1. }

Things: Properties and Relationships.

Things, as we have said, have properties, and have relationships with other things. It would be a very strange thing that had no properties and was not related to anything else. Perhaps that is what a no-thing, a nothing, is. But to say that a nothing is a special kind of thing, to say that it *is* at all, leads us into deep philosophical thicket. And our purposes, fortunately, do not lead us in that direction.

Consider a property (a one-place predicate), say color. Consider a relationship (an n-place predicate), say "being taller than" (a two-place predicate). How do we distinguish between things and their properties and relationships?

Grammar is a good guide to this distinction. If you can use “the”, “a”, “some” or “all” in front of the term, you are referring to things; otherwise not. For example, you can say “A customer who”, or “Some customers”, but not “A blue which” or “Some blues”. (You can, however, say "A color which" or "Some colors" . More on this later.)

Traditionally, things of the first type have been called “particulars”, while things of the second type have been called “universals”.¹ But because the word "particulars" has some connotations that may mislead us, we will reserve it as the cover term for both types and instances. There are types of particulars. I will call these types "natural kinds", more to avoid confusion with the ordinary use of the word "kind", but also following convention. Instances of natural kinds I will call "individuals".

What about universals? First, we can distinguish two kinds of universals – properties and relationships. For each, there are both types and instances. I'm not aware of a standard terminology to distinguish types of properties/relationships from instances of those types. So rather than making up some terms, I will rely on context to make it clear whether I am talking about types or instances of universals.

¹ The claim, made here, that particulars exist in their own right, while universals have only derived existence as properties or relationships of particulars, is both ancient and controversial. Those who make it are called nominalists; those who deny it are called realists. For the purposes of this article, however, it doesn't matter which position you take. It is worthy of note, however, that the realism vs. nominalism debate has an exact parallel in recent discussions of types of object-oriented systems. Those systems which treat classes “as first-class objects” are precisely realist, in the traditional philosophical sense. Those which do not are precisely nominalist. If you think that the difference between these two kinds of object-oriented systems is important, then you think that this ancient philosophical debate is important, too.

Let's take "blue" as a type. This is what we're talking about when we say "Blue is a pretty color". When we say "The getaway car was blue", we are talking about an instance of that type, an instance of that color.

Those who manage relational databases for a living should notice the following correlations, of which more will be said later.

1. Types of particulars, i.e. natural kinds, correspond to entities and tables.
2. Instances of those particulars correspond to entity instances and rows of tables.
3. Types of properties correspond to entity attributes and (non-foreign key) columns of tables.
4. Relationships as types correspond foreign key attributes and columns.
5. Instances of relationships correspond to foreign key instances, i.e. to a specific foreign key value in a specific table.

It is through these correlations that classical ontology can begin to be applied to data management issues. And the value of the correlation goes the other way, too. Tables in a (normalized) relational database correspond to relations that have been given an interpretation. And relations are precise mathematical objects; specifically, a relation is a subset of the Cartesian Product of an unordered set of sets, each of which is defined on a specific domain, where a domain is an ordered or unordered set of values and operations defined on them. This kind of mathematical rigor, I believe, might be useful to those philosophers who still think that the problem of particulars and universals is important.

Things: Particulars.

So: particulars are types whose instances exist in their own right. “Existence in its own right” is a concept that philosophers have discussed for centuries – indeed, for millennia. But the use to which I will put the term does not, I believe, require any reference to those discussions. Let’s just note that what exists in its own right is what does not exist merely as a feature of other things. You and I are human beings, and not merely properties of other things. In the terminology introduced above, all instances of particulars are individuals, and all individuals are instances of particulars.

By contrast, color is not a particular, and are red, blue and green are not individuals. Some individuals which are material objects *have* color, such as cars and flowers. So instances of color exist, but not in their own right. They exist as properties of things. So while instances of particulars exist in their own right, instances of universals exist only derivatively, as instances of the concepts which are their subtypes, and as properties of the individuals which they characterize.

Another thing to notice about individuals is that because they exist through time, they are subject to change. This is very important. So particulars are things whose instances can change. These changes we understand and express as changes to either their properties or their relationships. Possibly, coming into existence and dropping out of existence are additional properties (which we usually record as start dates and end dates in database tables) – although philosophers have also discussed whether or not existence (let alone existence in its own right) is a property.

Things, Once Again.

Another point about individuals, already alluded to, is that individuals have properties and relationships. For example, physical objects have mass and density; some of them have color. Non-physical objects have their own properties and relationships. Prime

numbers, for example, have the property that they are divisible only by themselves and 1.²

Another important point about both particulars and universals is related to Aristotle's distinction between essential properties and accidental properties, and to Leibniz's principle of the identity of indiscernibles. As to Aristotle's distinction, essential properties are those which, if they change, make the individual a different individual. As to Leibniz's principle, if two things cannot be distinguished by any of their properties or relationships, they are not two things, but one and the same thing.³

In a table which is a relation, i.e. a set, no two rows can be identical. A relational DBMS will not allow the insertion of two rows that have the same values in each of their columns. But usually, relational tables have primary keys. A primary key is a set of one or more relationships (implemented as foreign keys) and/or properties (non-foreign keys) that distinguish every row in the table from every other row. This key usually consists of fewer than all the columns of the table in question. A relational DBMS will also not allow the insert of two rows that have the same values in their primary key columns.

In the first case, we have the relational DBMS implementation of Leibniz's principle. In the second case, primary key columns are Aristotelian essential properties, while the other columns are accidental properties. Values in the non-primary key columns can change without forcing us to delete the row and insert a new row. Values of Aristotelian accidental properties can change without making the instance a different instance. As for primary keys, values in primary key columns cannot change. Looked at another way, if

² Which implies that prime numbers, and numbers generally, are particulars, not concepts. But this, in turn, implies that individual prime numbers, such as the number one (represented by the numeral "1") are individuals, and can exist in their own right (although not physically, of course). Others might argue that the number one is simple a property shared by all and only those sets which have a single member. If this is true, then numbers are universals, not particulars, and individual numbers are concepts, not individuals.

³ These and other allusions in this essay to philosophical discussions of ontology make a short essay on the history of ontology desirable. I will provide that essay later on.

they do change, that means we have a different row. Aristotelian essential properties cannot change or, if they do, that means we have a different individual.⁴

(to be continued.)

⁴ To be more accurate, the Aristotelian position is that a change in essential properties doesn't make an individual a different individual. It changes an individual from one natural kind to another. In fact, this isn't just more accurate. It's completely different from the way I am trying to apply the Aristotelian distinction to databases. So, another promissory note here, I will have to say more about why I am using the Aristotelian distinction in this incorrect way, later on.