

## ***Semantic Technology 2007 Conference: Speaker Notes.***

Dr. Tom Johnston  
MindfulData.com

### ***Slide 1. Semantics and Ontology in IT Data Management.***

Last revised 5/27/2007.

#### **Introduction.**

In the ecosphere of IT Data Management, a new ecological niche has opened up. It is the application of logic and ontology to the management of data. The expertise required to flourish in this niche comes from three areas: Philosophy, Computer Science and Experience!

#### **Philosophy.**

The central concerns of Philosophy are ontology (theory of being) and epistemology (theory of knowledge). Although the term "epistemology" has not gained the currency among computer scientists that the term "ontology" has, both logic and psychology (especially in the form of psycholinguistics) have strong epistemological presuppositions and entailments. The term is certain to make its way into the vernacular, and soon.

Additionally, since the time that predicate logic was invented by Gottlieb Frege at the end of the 19<sup>th</sup> century, and the "linguistic turn" was taken by Anglo-American philosophers at the start of the 20<sup>th</sup> century, logic and semantics have been central to the concerns of such philosophers.

Notable among them is the logician W. V. Quine, whose "principle of ontological commitment" is oft-quoted in this new ecology, though not so often understood. For those managing production databases in the IT world, the tables in those databases represent our ontological commitments, the kinds of things we take to exist.

Thirty years ago, I was finishing philosophy PhD. Ontology, logic and semantics (philosophy of language) were central to my studies. My doctoral

dissertation attempted to solve a problem in ethical theory by applying a central idea of Quine's.

### **Computer Science, Artificial Intelligence and Knowledge Representation.**

Combining mathematical (usually predicate) logic and a rigorous approach to categorization (ontology), they developed formal taxonomies and ontologies.

Where “formal” means that software can do the heavy lifting of inferencing/reasoning/theorem proving.

I have followed the work of these pioneers, as a student.

I was also a voting member and active participant in the IEEE1600.1 SUMO workgroup.

### **The Management of Business Data, Primarily in Relational Databases.**

The management of data began as a seat of the pants art form.

Codd put the management of large collections of structured data on a mathematical foundation.

I have been working in business IT for thirty years now.

The most important thing this experience emphasizes is this: real-world data is dirty. It is incredibly dirty! And yet, we must, somehow, manage it. Philosophers don't know what “dirty data” means. Computer scientists may understand the expression, but they have little experience with such data. The dirtiness of real-world data only obscures the search for elegant algorithms and structures to make our inferencing engines (such as DBMSs) increasingly powerful.

Dirty data is more than just dust on the surface of FOPL-powered engines. Dirty data is an eruption of chaos in a world of data nowhere as clean as most academics seem to assume it is.

The introduction of ontologies into IT Data Management is at least an order of magnitude more complex than the introduction of ontologies into a world of clean data. (my guess, but a carefully considered one.)

Dirty data: bad primary keys (entity integrity violations); imperfect match keys; reference chains broken (referential integrity violations);

### **Aristotle.**

Ontology, as comp scientists have been developing it, is a combination of ontology, and formal logic.

Aristotle (with Plato) is the father of ontology.

Aristotle, single-handed, is the father of formal logic.

But that's a story for another time.

### **Logic.**

Those "other assertions" are the data in our databases.

The assertions cranked out are guaranteed to be true, provided the premises they are based on are true.

Some of the assertions cranked out will be ones we didn't realize were true. We didn't realize the full implications – literally – of what we already knew. This is what the machinery of logic can do. This is the power that created the man-made world.

### ***Assumptions about Audience.***

I don't assume a background in Philosophy.

I don't assume familiarity with formal logic.

Focus of this talk. “us”, “we”: those who create and manage our companies’ relational databases.

So: data modelers, DBAs and their managers, all the way up to the CIO.

If any of this presentation still fails to communicate to you, in terms you are comfortable with, I’ll be here through the 24th, and will be happy to meet one-on-one with any of you.

## *Slide 2. Agenda.*

Formal ontology (ontology + logic) has arrived at the front door of business IT in the form of tools to manage data not in relational databases. Primarily text – emails, documents, pdf files, websites.

But we've been doing formal ontology ever since Codd. So although we custodians of our companies' relational databases are somewhat overlooked as this new, shiny stuff garners all the attention, we are where the action is, and where it will remain.

But if we've been doing formal ontology all along (and just not calling it that), what's really new?

Increased inferencing power for DBMSs. Ability to extract more information from data. Ability to extract information from more data. But it's not here yet. So what should we do to prepare? That's the topic of my talk tonight.

This presentation is pre-timed; the slides will change automatically. So please hold questions until the discussion part of the presentation.

*Slide 3. Part I.*

***Slide 4. Controlled Vocabulary: Definition.***

***Controlled Vocabulary.***

Terms which are minimally “fuzzy” around the edges.

As free as possible of semantic anomalies: synonyms, homonyms, ambiguity, vagueness.

Semantic relations made as explicit as possible: presupposition, inference, entailment.

Wittgensteinian “family resemblances” vs. Aristotelian definition by genus and difference. A big issue.

## ***Slide 5. Controlled Vocabularies and Data Dictionaries.***

The definitions in our data dictionaries aren't anything to be proud of. Most of them are sloppy. They are no foundation for automated inferencing by software, and a poor foundation for communication among persons.

Why are they so bad? Because they make no difference to the DBMS, or to the applications that access it. They're just documents that no one refers to. (Talk about self-fulfilling prophecies!)

### ***How to make the definitions rigorous?***

Base the definition on controlled vocabulary entries and references to other data dictionary entries. (Use Barron's prop.) The discipline of making your definitions conform will create the rigorous semantics we need. It will force implicit semantics out into the open.

But a major new discipline within IT must emerge: that of the semanticist/linguist/ontologist.

We can't just define every phrase that comes to mind. Because the result would include semantic anomalies such as:

- Ambiguity and vagueness
- Synonyms and homonyms
- Wittgensteinian family resemblances masquerading as Aristotelian definitions by genus and specific difference.
- IS-A and PART-OF are often confused.

Deciding on the "dictionaries", on what should be defined, is hard work. Creating good definitions is hard work!

## *Slide 6. Taxonomy: Definition.*

DBMSs don't make full use of their ability to express semantics. For example, with hierarchies:

- Little support for inheritance of attributes.
- No support for inheritance of methods.

And DBMSs, of course, only manage structured data, i.e. data in databases. They don't manage unstructured data, e.g. docs, emails, pdf files.

***Slide 7. Using a Thesaurus: an Example.***

We are searching for a concept, which may be specified by any number of different text strings. The concept is the semantics; the text strings are the data.

Without a formal thesaurus:

Some of us will search for all those text strings, others for one or more of them.

Inconsistent results will be returned for what all of us intended as a search for the same thing – the concept of SOA, across all the various ways it is alluded to.

## ***Slide 8. Ontology: Definition.***

While taxonomies are based on IS-A and INSTANCE-OF relationships, ontologies include other relationships such as “part of” and “belongs to” (meronymy relationships), and additional relationships specific to the subject matter being represented.

Relational data models are ontologies.

Quine’s criterion of ontological commitment: to be is to be the value of a variable.

With relational databases, this becomes: To be is to be the value of a typed variable.

Tables are the types. Rows are the variables. So the entities/tables in a relational data model / database are our ontological commitments to types. The rows are our ontological commitments to individuals, instances of those types.

This is the link between philosophical ontology and computer science ontology.

The link was presaged in Aristotle, the father of modern logic and (with Plato) of ontology.

Computer science ontology just combines logic and ontology.

It is impossible to exaggerate how, two and a half millennia ago, one man could have done all this, could have looked as much like a computer scientist at Stanford or Carnegie-Mellon as you care to imagine.

We follow. But he created, ex nihilo!

***Slide 9. Thesaurus: Definition.***

Informal (human-comprehensible only) thesauri collect terms that are semantically related, no matter how loosely. Their original purpose was when someone was “searching for just the right word”.

Formal thesauri require more strict semantics. Generally, synonyms and antonyms.

Synonyms as just string substitutions in searches.

Antonyms as substitutes for “Not {string}”.

## *Slide 10. Ontologies and Relational Databases.*

### **Extensional database.**

Contains the instances, i.e. rows in tables. What we think of as the database.

### **Intensional database.**

At its core, the logical data model, but now interpretable by software (DBMSs, other inference engines), not just by data modelers (people).

- But the DBMS doesn't know about customers, products, invoices, etc. The only ontology the DBMS understands is a mathematical one – physical structures representing mathematical sets and relations and links among those relations.
- We are the ones who know that one table is a customer table, another a product table, and so on. To the DBMS, “Customer”, “Product”, etc. are just character strings functioning as tags for different physical tables.

## *Slide 11. Inference Engine: Definition.*

Mathematical properties of relationships:

SQL DDL can declaratively express the minimum and maximum cardinalities of relationships, and the DBMS “understands” those declarations in the sense that it can enforce those constraints during update operations.

But SQL DDL cannot declare whether a relationship in a data model is (a) transitive or intransitive, (b) symmetric, antisymmetric or non-symmetric, or (c) reflexive or irreflexive. These are the basic mathematical properties of any relationship.

Listing these properties with every relationship definition will be an important step in preparing to use new semantic inferencing abilities that vendors will eventually supply.

***Slide 12. Semantics: Definition.***

### *Slide 13. Semantics: Commentary.*

#### **Transformational rules in program code.**

- This hides the semantics, so only the programmer knows what they are.
- This is inflexible because the semantics are expressed procedurally, and are consequently compile-time bound.
- But transformational rules can be expressed in data structures -- state transition tables. More sophisticated, colored Petri Nets.

#### **Structural Rules in database schemas.**

- These rules are not hidden, and they are not expressed procedurally.
- But they are nonetheless compile-time bound.
- And they are even more difficult to change than procedural code – because procedural code is affected by changing them (and not vice versa).

***Slide 14. Semantic Interoperability: Definition.***

Label: the text string plus its formal semantics.

Synonymity is the strongest of these relationships, but not the only one.

***Slide 15. Semantic Interoperability: Commentary.***

The past: managing data and, in the process, without quite being aware of how, managing the semantics -- kind of.

The future: managing the semantics, whether expressed in data or in functions, in schemas or in code.

The combined codebase and database now appear as one physical realization of our company's semantics.

Management of these physical artifacts is now consciously driven from directives arising from the management of semantics.

***Slide 16. Part II.***

***Slide 17. Objective 1. Develop a Rigorous Semantics: Data Dictionaries and Controlled Vocabularies.***

***Slide 18. Objective 1. Develop a Rigorous Semantics: Type Hierarchies and Taxonomies.***

Inventory the taxonomies in the enterprise's databases. These are the supertype/subtype hierarchies.

Integrate these taxonomies in the Enterprise Data Model.

Apply these taxonomies consistently across all enterprise databases:

- OLTP developed or purchased systems
- Operational Data Stores
- Data Warehouses
- Conformed Dimension Data Marts
- Extracts for downstream use (e.g. for SAS reporting)

Will be writing in the near future on how to do this. Children of non-leaf nodes must partition the parent node – be:

- Mutually exclusive; and
- Collectively exhaustive.

***Slide 19. Objective 2. Develop a Formalized Semantics: Database Schemas in FOPL.***

This is a good example because the vast majority of relationships in a relational model are one-to-many (aka parent-to-child), optional for the parent entity and required for the child.

Theorem-proving inferencing is essential for extracting information from unstructured data. This automation of logic depends on a formalized taxonomy and ontology.

Theorem-proving inferencing is not very important for querying tables of structured data, because what a query asks is very simple: “Find me all the rows that look like this .....” (described in the SQL WHERE clause).

But semantic interoperability among structured databases has the same requirements as theorem-proving inferencing. To determine semantic interoperability, formal taxonomies and ontologies must exist which define the meaning of entities, attributes and relationships.

So we in IT need formal taxonomies and ontologies, after all.

***Slide 20. Objective 3. Develop an Integrated Semantics.***

***Slide 21. Objective 4: Develop a Late-Bound Semantics.***

**Reasons:**

- In order to make them visible to business users.
- In order to consolidate them in one place.
- In order to reduce the cost of changing them.

**Hardcoding to be removed from programs:**

- Constants that might change.
- Formulas that might change.
- Decisions that might change.

**Hardcoding to be removed from schemas:**

- Ontological commitments that might change.

***Slide 22. Part III.***

*Slide 23. Part IV.*

## ***Slide 24. Final Thoughts – 1.***

Most companies make money by processing transactions – often very large volumes of transactions.

These transactions alter the balance of the relationship between the company and its suppliers, customers, regulators, etc. “alter the balance” both metaphorically and literally.

We’ve been doing a pretty good job of it so far. In which case, why should we care about ontologies, etc?

Not primarily because of enhanced inferencing. Primarily because of enhanced descriptions, and the semantic interoperability they make possible.

***Slide 25. Final Thoughts – 2.***

These are the four objectives, the agenda for Part II.

### ***Slide 26. Final Thoughts – 3.***

No semantic silos. No ontology silos.

And in big companies, silos pop up like mushrooms.  
You don't have to do anything to create them.

You have to work hard to prevent them.  
Because silos reflect "turf". And turf wars are ubiquitous in large corporations.

We used to be concerned that databases couldn't "talk" to one another.  
We've pretty much solved that problem at the level of data.  
We've only begun to address the issue of insuring that we all mean the same thing by the same data (or the same functions.)

***Slide 27. End of Presentation.***

Brochures describing my consulting services are available at the front table.