Ontologies and Database Schema: What's the Difference?

Michael Uschold, PhD
Semantic Arts

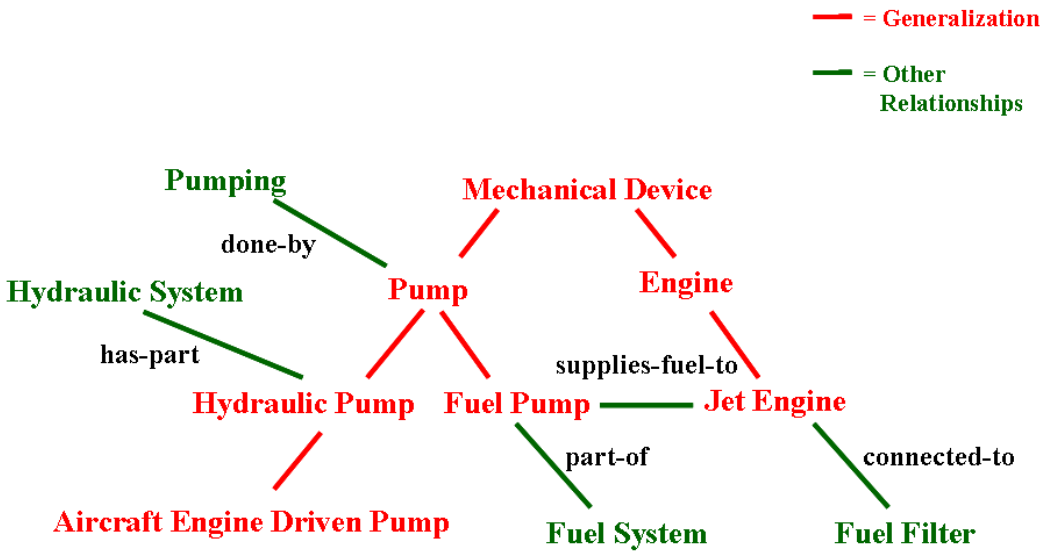
Objective

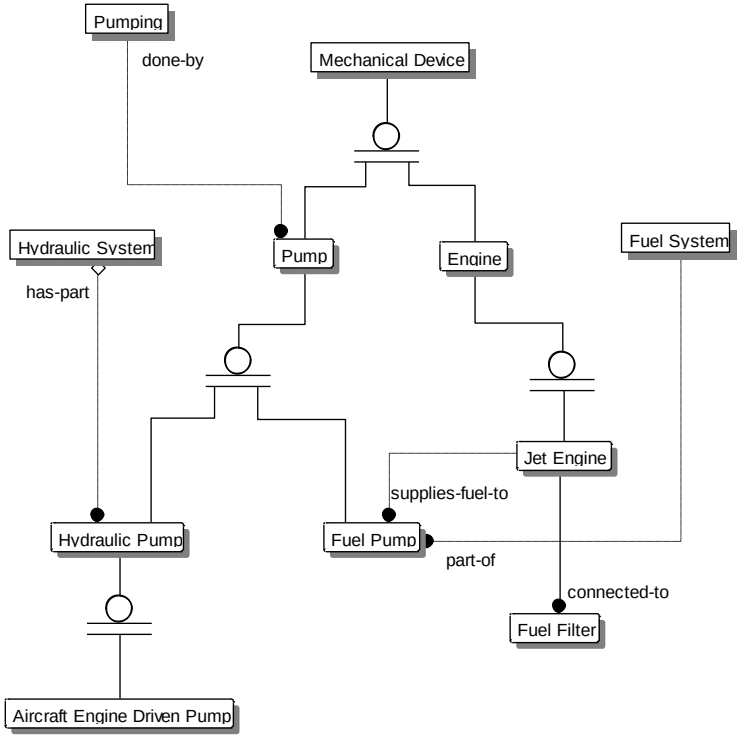
To settle once and for all the question:

What is the difference between an ontology and a database schema?

- Often asked.
- Never adequately answered.

Example: Hydraulics





Ontology **Logical DB Schema**: IDEF1X

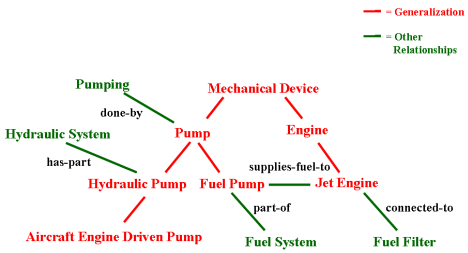
Is this similarity just superficial?

The Basic Idea for Each

For shared understanding.

Ontology:

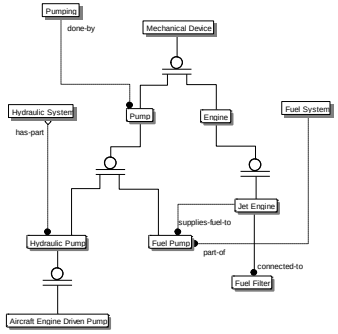
- defines a set of concepts and relationships
- that represent the content and structure
- of some subject matter
- in a formal language.



For a database.

Database Schema:

- defines the structure of a database in a formal language.
- loosely refers to any of: conceptual, logical, physical



Five Questions to get a Better Understanding

1. **What is it for?**
2. **What does it look like?**
3. **How do you build one?**
4. **How is it implemented and used?**
5. **Where are the semantics?**

"**it**": ontology *or* database schema

Quick Poll: more alike?

Are they more alike or different?

Quick Poll: more different?

What is it for?

	DB Schema	Ontology
Focus	**Data**	**Meaning** **Shared Understanding**
Core Purpose(s)	Structure instances for efficient storage and querying.	Human communication, interoperability, search, software engineering, …
Thus	Meaning lost.	Instances optional.

Single purpose.

Also for structuring instances.

What does it look like? Notation

	DB Schema	Ontology
Notation: syntax	ER diagrams; no standard serialization syntax.	Logic; no standard diagram notation syntax.
Notation: semantics	Minimal focus on formal semantics.	Strong focus on formal semantics.

What does it look like? Expressivity

	DB Schema	Ontology
Expressivity overlap	Entities	Classes
	Attributes, Relations	Properties
	Constraints	Axioms
Expressivity differences	No taxonomy.	Taxonomy is backbone.
	Constraints for integrity, foreign key, delete.	Constraints for meaning, consistency & integrity.

Cardinality constraints.

How to Build One?

	DB Schema	Ontology
Starting point:	Scratch, rarely reuse.	Reuse if possible.
Normalization:	Standard rules in natural language, little tool support.	No standard rules or guidelines.
Optimization:	Fundamental step. Manual, geared to specific queries for specific DB.	Ontology independent; Inference engine developers.

OntoClean

Some tuning may be required. A black art.

Toss expensive constraints. Dirty data, lost meaning.

How is it Implemented and Used?

	DB Schema	Ontology
Change Management, Agility, Flexibility	Locked into specific set of queries per DB.	No query lock-in. Queries usable on other systems.
	Tight coupling.	Looser coupling.
	Lost meaning.	Semantics explicit.
	Hard to evolve and maintain. ETL tools to help.	Potentially easier to evolve & maintain. Few tools.

Semantics hardwired in procedural code.

Still no picnic!

How is it Implemented and Used?

	DB Schema	Ontology
Processing Engines	SQL Engines	Theorem Provers
Primary Focus:	Queries	Derive new information from existing information.
	Reasoning with Views	
	Data integrity	Consistency and integrity
	Standardized on SQL	Less standardization

Both handle complex logical expressions.

How is it Implemented and Used?

	DB Schema	Ontology
Performance	Highly tuned for performance and scale.	Full inferencing: much smaller scale.
	Not work well with too many joins.	Reduced inferencing: reaching large scale.

Tradeoff: Performance vs. Function & Flexibility

RDB and Triple Stores both have a Niche.

Building Databases & Applications

- **Gather Requirements**
- **Build Conceptual Schema**
  (e.g., ER or UML model)
- **Refine to Logical Schema**
  (e.g., normalize, still ER or UML)
- **Refine to Physical Schema…**

Refine to Physical Schema

- **Define tables, columns, keys.**
- **Optimize to Specific Kinds of Queries.**
- **Create Data Dictionary** (semantics for humans).
- **Integrity Constraints**
  - Domain, Referential & Semantic integrity
  - Do the best you can, little automation.
- **Where are the Semantics?**

Five Questions to get a Better Understanding

1. What is it for?
2. What does it look like?
3. How do you build one?
4. How is it implemented and used?
5. **Where are the semantics?**

"**it**": ontology *or* database schema

Where are the Semantics for Database Schema?

- **Mainly in Conceptual Schema and Data Dictionary**
- **Designed for Humans**
- **Semantics don't evolve as DB and applications change**
- **Conceptual Schema semantics thrown away when building Physical Schema.**
- **Integrity constraints hardwired in procedural code**

Semantics are hardwired, lost, tossed, or out of date.

You cannot maintain what you do not understand!

Quick Summary

Database Schema

Focus on DATA

DB Constraints
- to ensure integrity
- may hint at meaning

No ISA hierarchy

SQL Engines
- querying, views
- data integrity

Instances Central

Data Dictionary
- separate artifact

Ontology

FOCUS on Meaning

Ontology Axioms
- to specify meaning
- maybe for integrity

ISA Hierarchy is Backbone

Theorem Provers
- infer new information
- ensure consistency

Instances Optional

'Comments'
- part of the ontology

More Detailed Summary: 24 Features/Aspects

	Core for DB Schema	Secondary for DB Schema	Unimportant for DB Schema
Core for Ontology	1. Represented using a formal language. 2. Expressivity: types, properties, constraints. 3. Constraints for consistency checking.	4. Shared meaning of some subject matter. 5. Taxonomy. 6. Multi-purpose. 7. Embedded natural language definitions. 8. Constraints for meaning. 9. Constraints for ensuring self-consistency (not data).	16. Abstract types w/no instances. 17. Reused to build new ones. 18. Reused in unexpected ways. 19. Formal model-theoretic semantics.
Secondary for Ontology	10. Efficient querying and storage for data. 11. Standardized diagram notation. 12. Separate natural language definitions (data dictionary). 13. Constraints for data integrity. 14. Industry-wide construction guidelines (normalization). 15. Scale to huge sizes.	**More Alike?** • **Over 60% of features are common to both.** • **The 3 features core to both are the most important: what is expressed and how.**	
Unimportant for Ontology	20. Cardinality constraints for getting foreign keys right and ensuring tables created for many-to-many relationships. 21. Toss semantics after conceptual modeling. 22. Optimization for specific set of queries. 23. Sophisticated tool support for migrating data when schema evolve (ETL). 24. SQL for querying data.	**More Different?** **Only 12% of features are core to both.** ***Can you convert one into the other?***	

semantic arts

More Alike? or More Different?

Conversion?

- Conceptual Schema & Ontology
  No harder than between two different ontology languages

- Ontology & Logical Schema
  Some loss of information, a design artifact

- Ontology & Physical Schema
  Much loss of information, an implementation artifact

My Big Fat Greek Wedding Toast:

… even though we are apples and oranges, we are all fruit.

Conclusion

- They are similar beasts.
- They evolved in different communities.

Two cultures divided by a common idea?

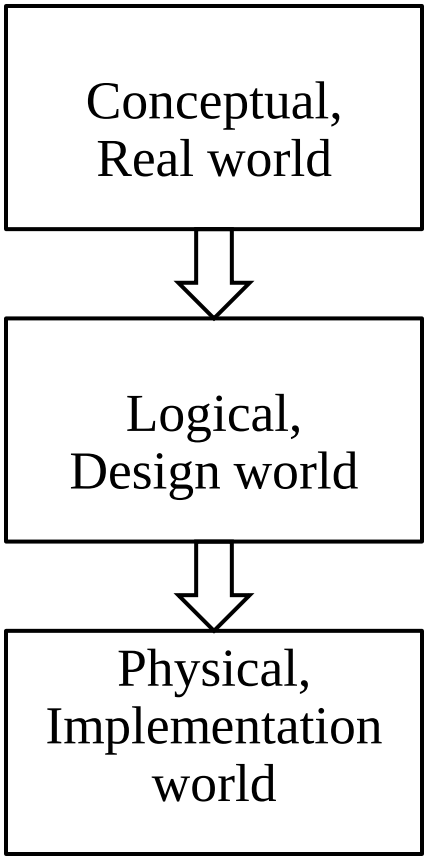
Speaking of weddings… What do you get if you cross:

- DB schema technology/community?
- Ontology technology/community?

Ontology/Model-Driven Architecture & Development

Basic Idea:

- Explicitly capture the semantics as formal ontologies.
- Base design- and implementation-level artifacts on the ontologies.
- Ontologies drive the applications, directly or indirectly.



Benefits:

- Looser coupling.
- Semantics is explicit.
- Integration/Interoperability by design.
- Inference gives automated consistency checking.
- Easier to evolve and maintain:
  - Less hardwiring of semantics means easier to understand and change
  - Ontologies evolve with the DB and applications.

Acknowledgements

For answering countless questions as a cube-mate:
- John Thompson

For reviews of a companion unpublished paper:
- Phil Bernstein,
- Tim Wilmering,
- Jun Yuan,
- Anhai Doan,
- Bill Andersen,
- Amit Sheth.

For ideas on model-driven development:
- Simon Robe.