#ontologies

shaurya agarwal

@

22 years gainfully employed: data cloud machine learning ai et. al.

https://www.linkedin.com
/in/**shauryashaurya**/

https://github.com/*shauryashaurya*/The-Silmaril

thank you tons!

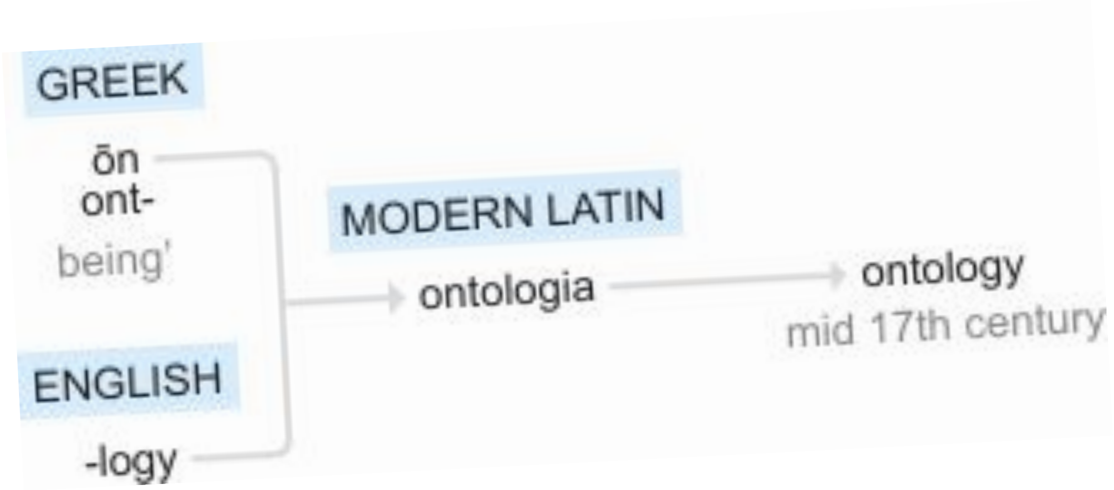
+

what do you mean

ontology?

ontology

...lmgtfy...



ontology

...ummm...

a set of concepts and categories in a subject area or domain that shows their properties and the relations between them.

ontology

...uh...

the branch of
metaphysics dealing
with the nature of being.

...you already know

let’s try to understand by building one

...you already know

movies

we all know a bit about ‘em

what words come to mind?

movies

“Scorsese’s a great movie director”

“Cruise is an incredible actor, so is Hanks”

“Back to the Future didn’t win awards but was popular!”

“Low budget indie horror is the genre with the best returns…”

what words come to mind?

movies

“Scorsese’s a great movie **director**”

“Cruise is an incredible **actor**, so is Hanks”

“Back to the Future didn’t win **awards** but was **popular**!”

“Low budget indie **horror** is the **genre** with the best returns…”

what words come to mind?

movies

“**director**”

“**actor**”

“**awards**”

“**genre**”

classes in the *movies ontology*

we kinda knew it right?

movies

“director”

“actor”

“awards”

“genre”

Core Concepts (Classes):

1. **Movie**: A cinematic film.
2. **Person**: Any individual involved in the movie. This can be specialized into:
   - **Actor**: A person who acts in movies.
   - **Director**: A person who directs movies.
3. **Character**: A fictional or real role portrayed in a movie.
4. **Genre**: A type/category of movies (e.g., Action, Drama, Comedy).

Data Properties (attributes):

- **Movie**: `title` (string), `releaseYear` (integer), `duration` (integer, in minutes), `rating` (float, e.g., IMDB rating).
- **Person**: name (string), `birthDate` (date).
- **Character**: name (string).
- **Genre**: name (string).

Relationships (Object Properties):

1. `hasActor` (Movie -> Person) : indicates who acted in the movie.
2. `hasDirector` (Movie -> Person) : indicates who directed the movie.
3. `playsCharacter` (Actor -> Character) : indicates which character an actor portrayed.
4. `belongsToGenre` (Movie -> Genre) : indicates which genre(s) the movie belongs to.

what words come to mind?

movies

“**director**”

“**actor**”

“**awards**”

“**genre**”

OBJECT oriented?

relational databases?

what words come to mind?

movies

“**director**”

“**actor**”

“**awards**”

“**genre**”

YES,

kinda, and no (kinda)

we just built one

what do you mean

ontology?

we just built one

…& we still don’t know
what it is…

what do you mean

ontology?

declarative

...& vocabularies

2 words are going to be key

ontology

declarative
is not imperative

...& vocabularies one word for one thing
and a collection of all the unique words

2 words are going to be

key

ontology

declarative
is not imperative

…& vocabularies one word for one thing
and a collection of all the unique words

2 words are going to be

key

ontology

…tell them about that PhD thing…

and

e·pis·te·mol·o·gy

/əˌpistəˈmäləjē,eˌpistəˈmäləjē/

declarative
is not imperative

…& vocabularies one word for one thing
and a collection of all the unique words

2 words are going to be

key

ontology *WHAT DO WE KNOW?*

…tell them about that PhD thing…

e·pis·te·mol·o·gy

/əˌpistəˈmäləjē,eˌpistəˈmäləjē/

knowing about knowing

HOW DO WE KNOW?

WHAT
not HOW

...& vocabularies one word for one thing
and a collection of all the unique words

2 words are going to be

key

ontology

WHAT
not HOW

...& vocabularies one word for one thing
and a collection of all the unique words

WHAT ==

ontology

WHAT
not HOW

...& vocabularies one word for one thing
and a collection of all the unique words

WHAT ==

ontology

whoa!

i blame the name

then why is the idea not more commonly understood?

ontology?

harry potter

the marauder’s map?

…we can do better

ontology?

douglas adams

hhgttg?

...too long..

ontology?

lotr

ah! how about...

...

ontology?

shaurya shaurya
The Silmaril
ontologies

...vocab

clearly state the concept

clearly

one world, one word, one concept

…declarative

SQL - you say WHAT, the database engine figures out HOW

Pandas - you say WHAT, the pandas engine figures out HOW

separating WHAT and HOW helps

… a lot of ‘reasoning’ and ‘logic’ becomes *imperative* and ontologies help us drag it back to being ***declarative***

comparing apples to oranges and other different animals

comparing apples to oranges and other different animals

Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS

comparing apples to oranges and other different animals

Abstract Syntax Trees - Syntactic structure without domain meaning, just grammar

TAXONOMIES- Simple semantic organization

THESAURUS- TAXONOMY + Synonyms and Antonyms

ONTOLOGIES- Rich semantic definitions (Rules, Restrictions etc.) over Taxonomies and Thesauri

KNOWLEDGE GRAPHS- Applied semantics with real-world entities

comparing apples to oranges and other different animals

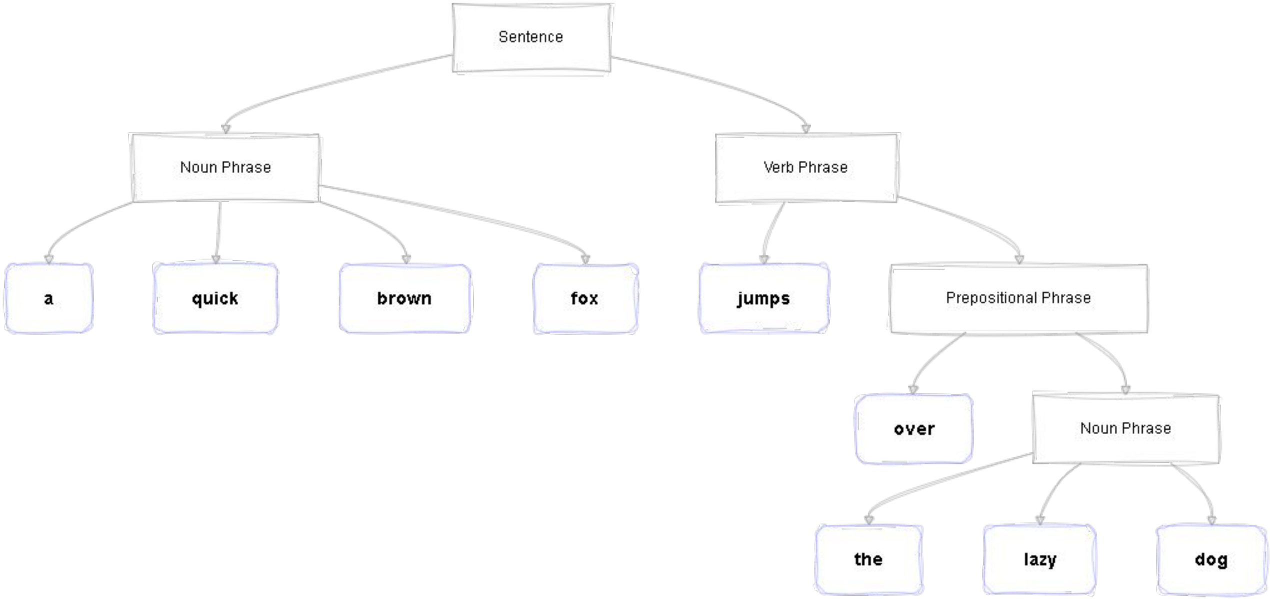
Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS



comparing apples to oranges and other different animals

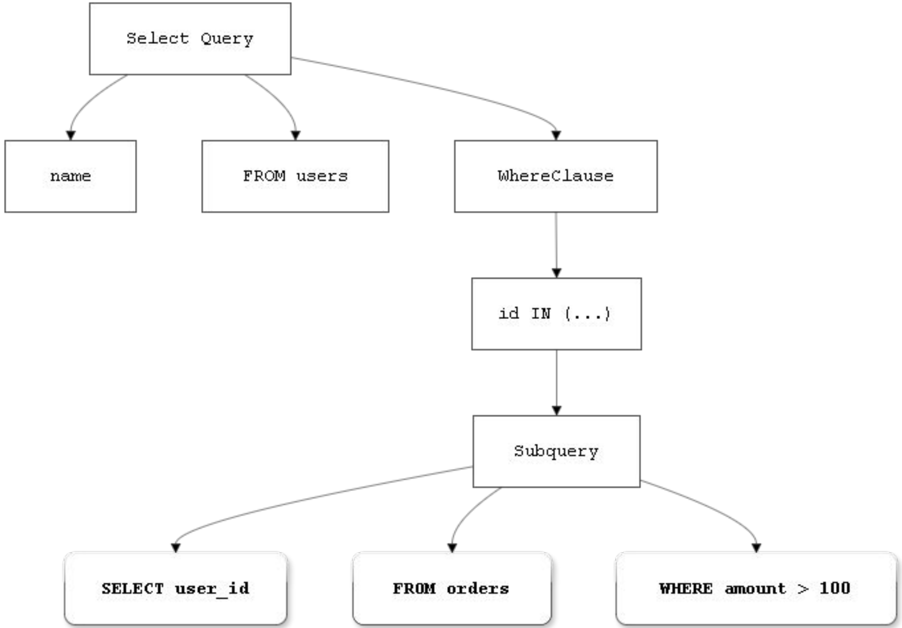
Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS



comparing apples to oranges and other different animals

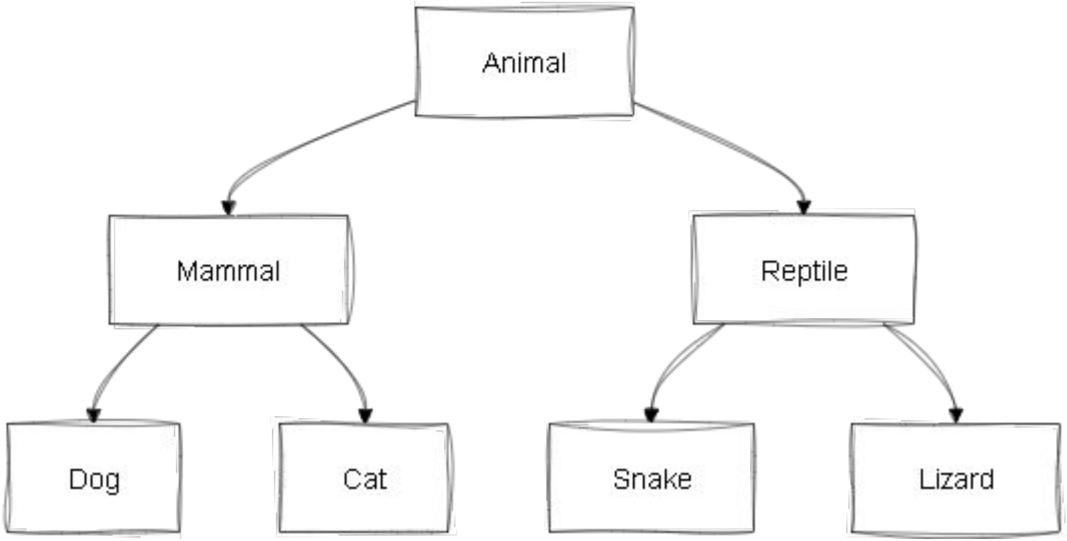
Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS



comparing apples to oranges and other different animals

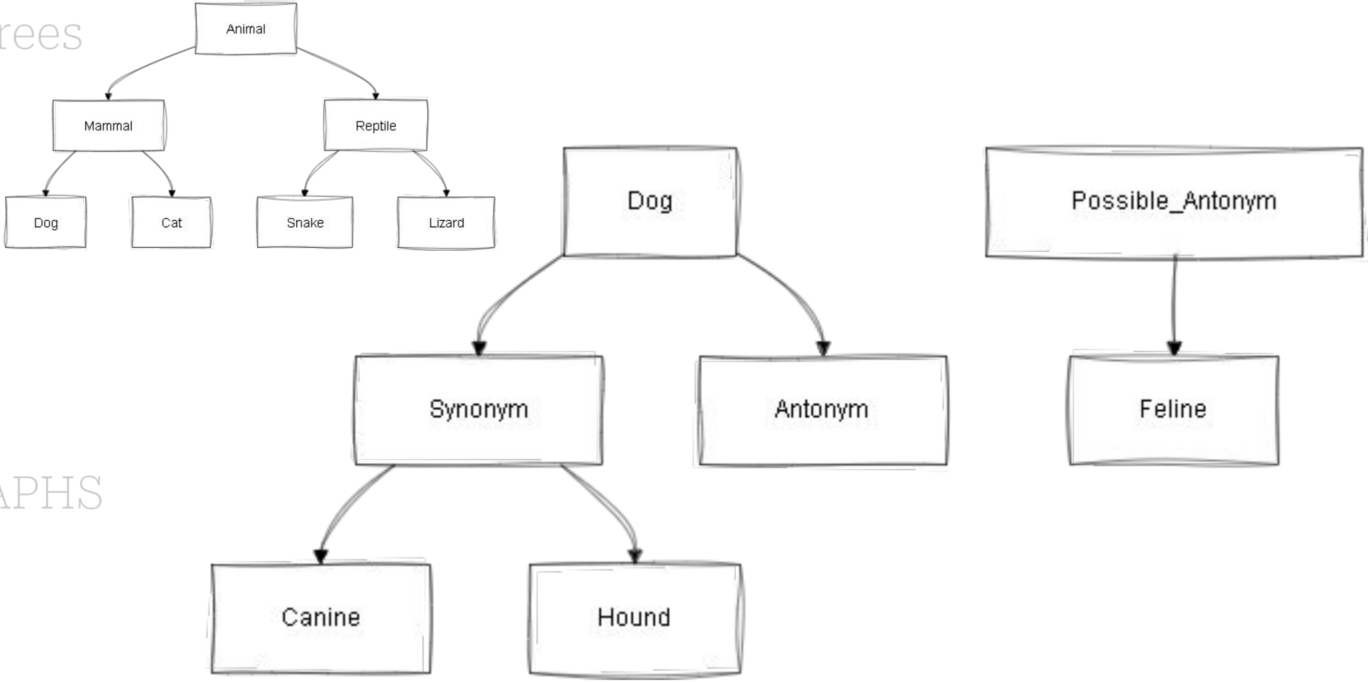
Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS



comparing apples to oranges and other different animals

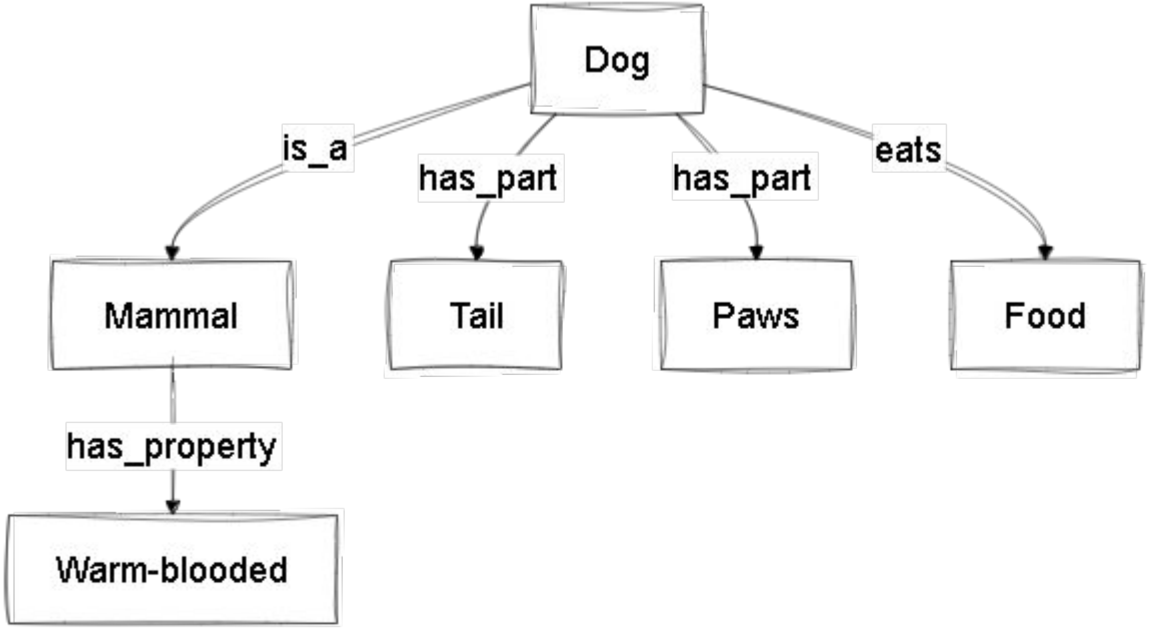
Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS



comparing apples to oranges and other different animals

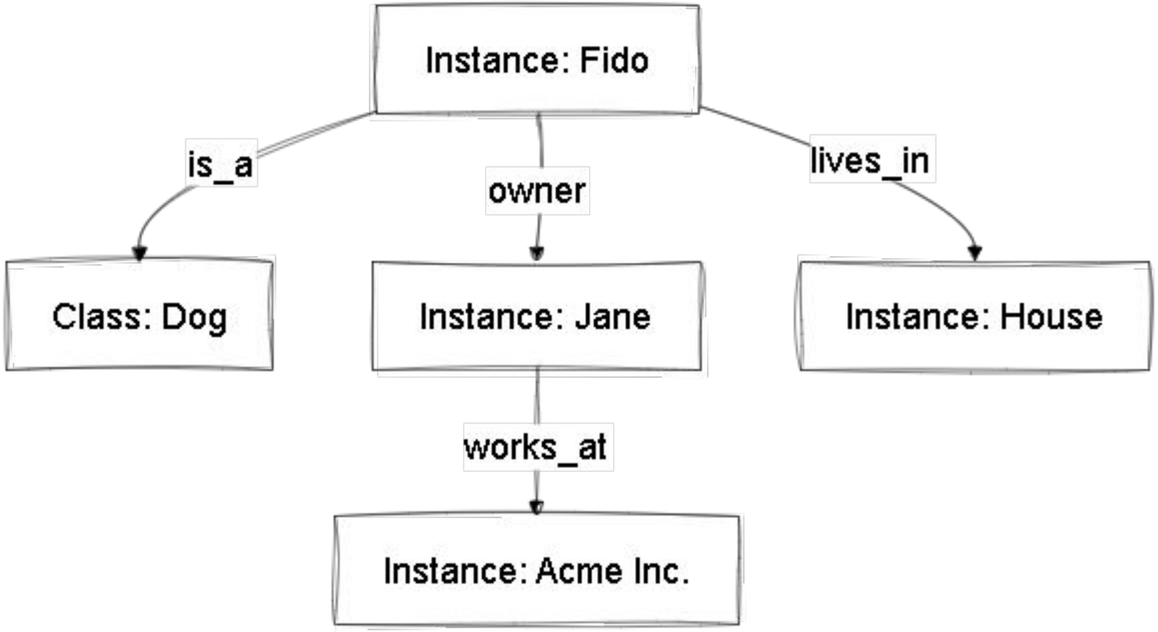
Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS



comparing apples to oranges and other different animals

Abstract Syntax Trees

TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS

Taxonomies and ontologies provide *schemas (rules, structure)*

Knowledge graphs populate these schemas with actual data instances

comparing apples to oranges and other different animals

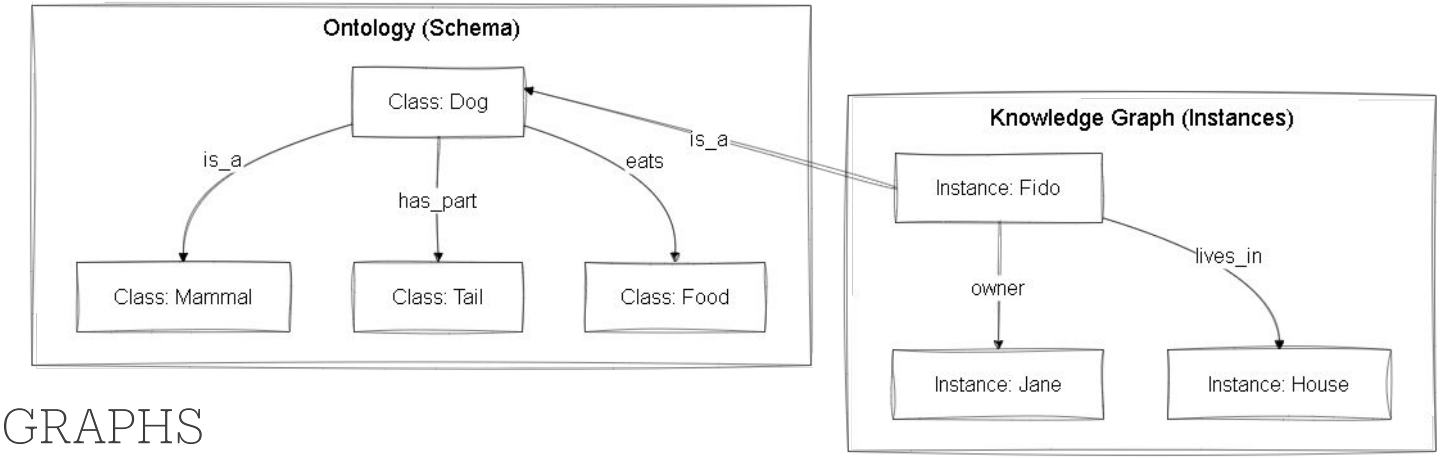
Abstract Syntax Trees

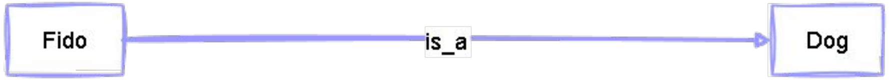
TAXONOMIES

THESAURUS

ONTOLOGIES

KNOWLEDGE GRAPHS





comparing apples to oranges and other different animals

Abstract Syntax Trees - Syntactic structure without domain meaning, just grammar

TAXONOMIES- Simple semantic organization

THESAURUS- TAXONOMY + Synonyms and Antonyms

ONTOLOGIES- Rich semantic definitions (Rules, Restrictions etc.) over Taxonomies and Thesauri

KNOWLEDGE GRAPHS- Applied semantics with real-world entities

comparing apples to oranges and other different animals

	AST	**Ontology**	**Knowledge Graph**	**Taxonomy**
Structure	Tree	Graph	Graph	Hierarchy / Tree
Focus	Syntax	Semantics	Instances	Classification
Relationship types	Grammar-based	Multiple semantic types	Multiple semantic types	Mainly "is-a"
Purpose	Code processing	Knowledge representation	Data integration	Organization
Flexibility	Fixed grammar	Extensible	Highly extensible	Limited

The Library Analogy

AST is like the grammatical structure of sentences in books

Taxonomy is like the Dewey Decimal System organizing books by subject

Ontology is like a detailed catalog that includes cross-references, related topics, and subject interconnections

Knowledge Graph is like the entire library with all its books, connections, and references implemented

The City Analogy

AST is like the blueprint of a building, showing its structural components

Taxonomy is like organizing city areas into districts, neighborhoods, and blocks

Ontology is like a city planning document defining what constitutes residential zones, commercial areas, how they relate, and rules governing them

Knowledge Graph is like the actual city with specific buildings, streets, and the relationships between them

reason with ontologies

discover and auto-generate classes

- **award winners** - the reasoner sees actors/directors with award relationships and creates `oscarwinner`, `awardnominee` classes automatically.
- **frequent collaborators** - spots patterns like scorsese + deniro appearing together repeatedly and creates a `frequentcollaborators` class.
- **veterans vs newcomers** - counts movie relationships and birth dates to automatically classify people as `veteranactor` or `risingtalent`.
- **movie eras** - groups films by release year into `classiccinema` (pre-1970), `moderncinema`, etc.
- **franchise groups** - identifies sequels and shared characters to create `actionfranchise`, `horrorseries` classes.

reason with ontologies

why this beats traditional approaches

- **rdbms:**
  - manually create every table and relationship. miss something?
  - too bad - it doesn't exist in the database.
- **oop:**
  - hardcode class hierarchies upfront.
  - want to find "prolific directors"?
  - write custom queries every time.
- **ontology + reasoner:**
  - define the rules once ("someone who directed 10+ movies is prolific"), and it automatically discovers and classifies people.
  - it's like having a smart assistant that spots patterns you didn't even think to look for.

the **big win**?

discovery over hard-coded storage.

instead of just keeping data, you're generating new insights automatically.

reason with ontologies

Integration Benefits

- **Data Merging** - Combine IMDB, Netflix, Rotten Tomatoes automatically. The ontology knows `"director" = "filmmaker" = "réalisateur"` and maps them instantly.
- **No Schema Headaches** - Add "`showrunner`" or "`voice actor`"? Just declare it as a `Person type`. No database changes needed.

Analysis Power

- **Smart Queries** - Ask "sci-fi movies with action actors" in plain language. Traditional SQL needs complex joins and guesswork.
- **Auto Error-Catching** - Finds impossible data like "person born 1990, directed 1985 movie."
- **Easy Connections** - Link movies to music (soundtracks), books (adaptations), places (filming locations) without rebuilding anything.

The Declarative Advantage

Here's the key difference: **You declare what things mean, not how to find them.**

Traditional approach: Write step-by-step code to find veteran actors. Ontology approach: Declare "veteran = person with 20+ movies" and the reasoner does the work.

Real Benefits:

- APIs with different field names? Auto-mapped
- ML training data? Relationships already explicit
- Future tech like VR movies? Just declare new types

Bottom line: Databases store facts. Ontologies understand meaning. You describe the world once, get insights forever.

reason with ontologies

Integration Benefits

- **Data Merging** - Combine IMDB, Netflix, Rotten Tomatoes automatically. The ontology knows `"director" = "filmmaker" = "réalisateur"` and maps them instantly.
- **No Schema Headaches** - Add "`showrunner`" or "`voice actor`"? Just declare it as a `Person type`. No database changes needed.

Analysis Power

- **Smart Queries** - Ask "sci-fi movies with action actors" in plain language. Traditional SQL needs complex joins and guesswork.
- **Auto Error-Catching** - Finds impossible data like "person born 1990, directed 1985 movie."
- **Easy Connections** - Link movies to music (soundtracks), books (adaptations), places (filming locations) without rebuilding anything.

The Declarative Advantage

Here's the key difference: **You declare what things mean, not how to find them.**

Traditional approach: Write step-by-step code to find veteran actors. Ontology approach: Declare "veteran = person with 20+ movies" and the reasoner does the work.

Real Benefits:

- APIs with different field names? Auto-mapped
- ML training data? Relationships already explicit
- Future tech like VR movies? Just declare new types

Bottom line: Databases store facts. *Ontologies understand meaning*. You describe the world once, get insights forever.

reason with ontologies

Integration Benefits

- **Data Merging** - Combine IMDB, Netflix, Rotten Tomatoes automatically. The ontology knows `"director" = "filmmaker" = "réalisateur"` and maps them instantly.
- **No Schema Headaches** - Add "`showrunner`" or "`voice actor`"? Just declare it as a `Person type`. No database changes needed.

Analysis Power

- **Smart Queries** - Ask "sci-fi movies with action actors" in plain language. Traditional SQL needs complex joins and guesswork.
- **Auto Error-Catching** - Finds impossible data like "person born 1990, directed 1985 movie."
- **Easy Connections** - Link movies to music (soundtracks), books (adaptations), places (filming locations) without rebuilding anything.

The Declarative Advantage

Here's the key difference: **You declare what things mean, not how to find them.**

Traditional approach: Write step-by-step code to find veteran actors. Ontology approach: Declare "veteran = person with 20+ movies" and the reasoner does the work.

Real Benefits:

- APIs with different field names? Auto-mapped
- ML training data? Relationships already explicit
- Future tech like VR movies? Just declare new types

Bottom line: Databases store facts. *Ontologies understand meaning*. You describe the world once, get insights forever.

‘nuff said...

let’s build

Installation and Setup instructions for **The Silmaril**

The easiest thing is to get the latest Anaconda distribution.

We will largely be using:

- `rdflib`
- `owlready2`
- `pandas`
- `numpy`
- `jupyter lab`
- `vs code`

So kinda simple.

Use conda or pip or ruff or whatever you like to get these.

The code is at:

https://github.com/shauryashaurya/The-Silmaril