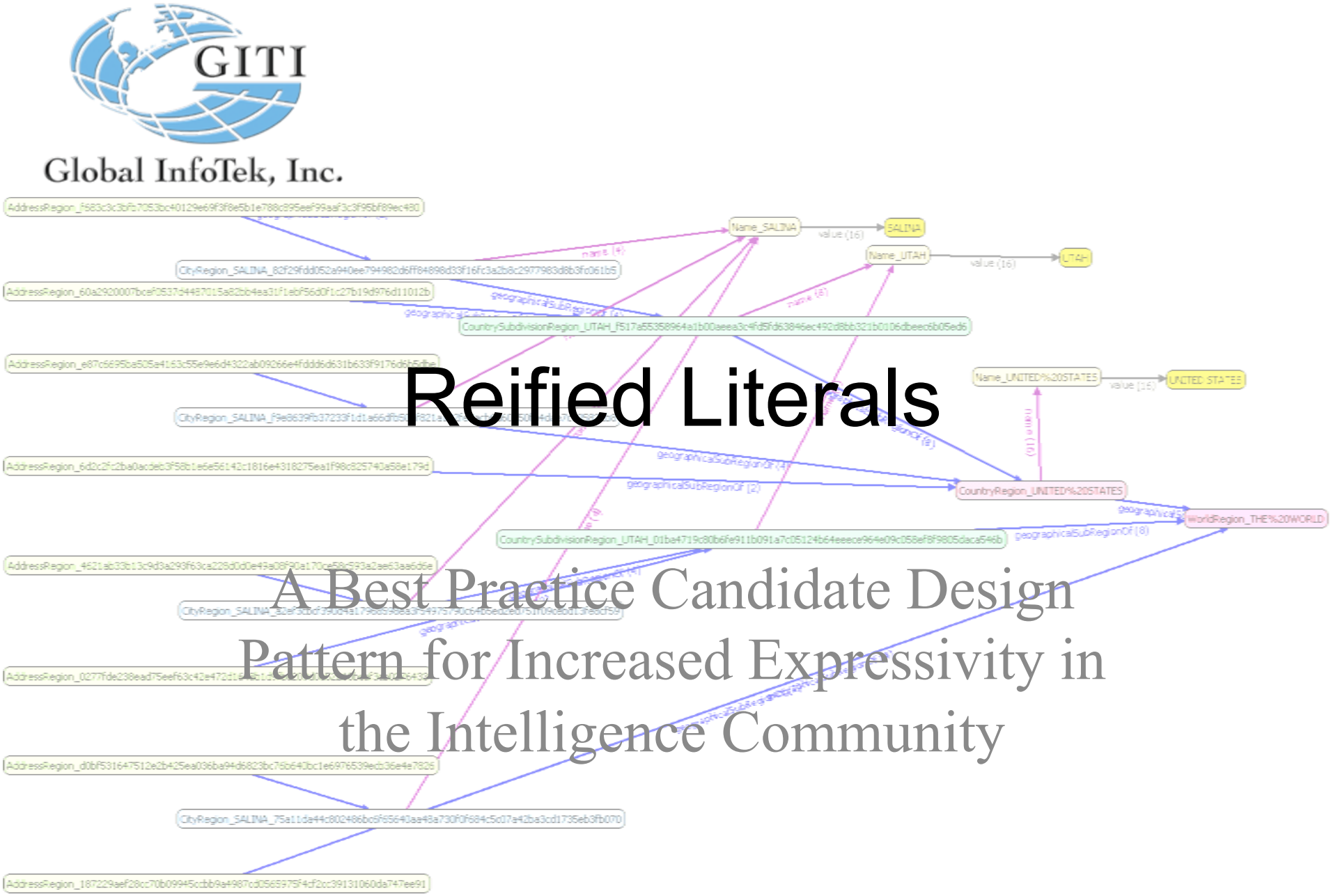
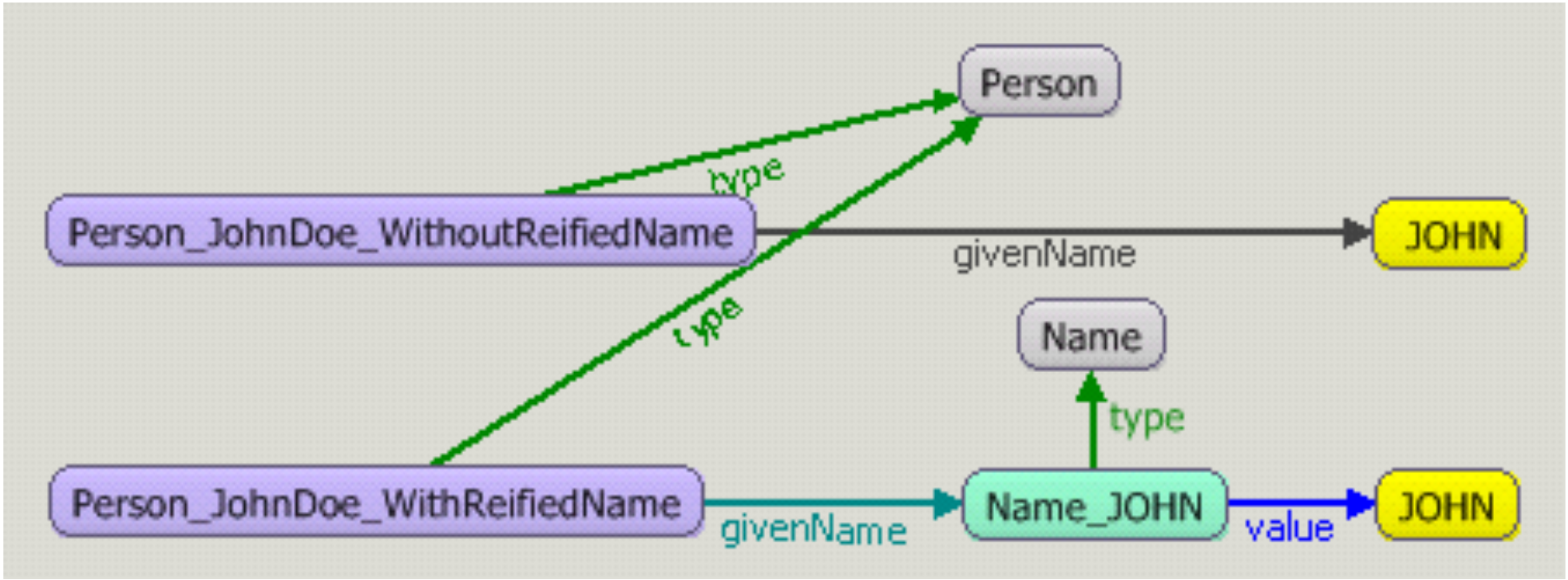

GITI
Global InfoTek, Inc.
Reified Literals
A Best Practice Candidate Design Pattern for Increased Expressivity in the Intelligence Community

Global InfoTek, Inc.

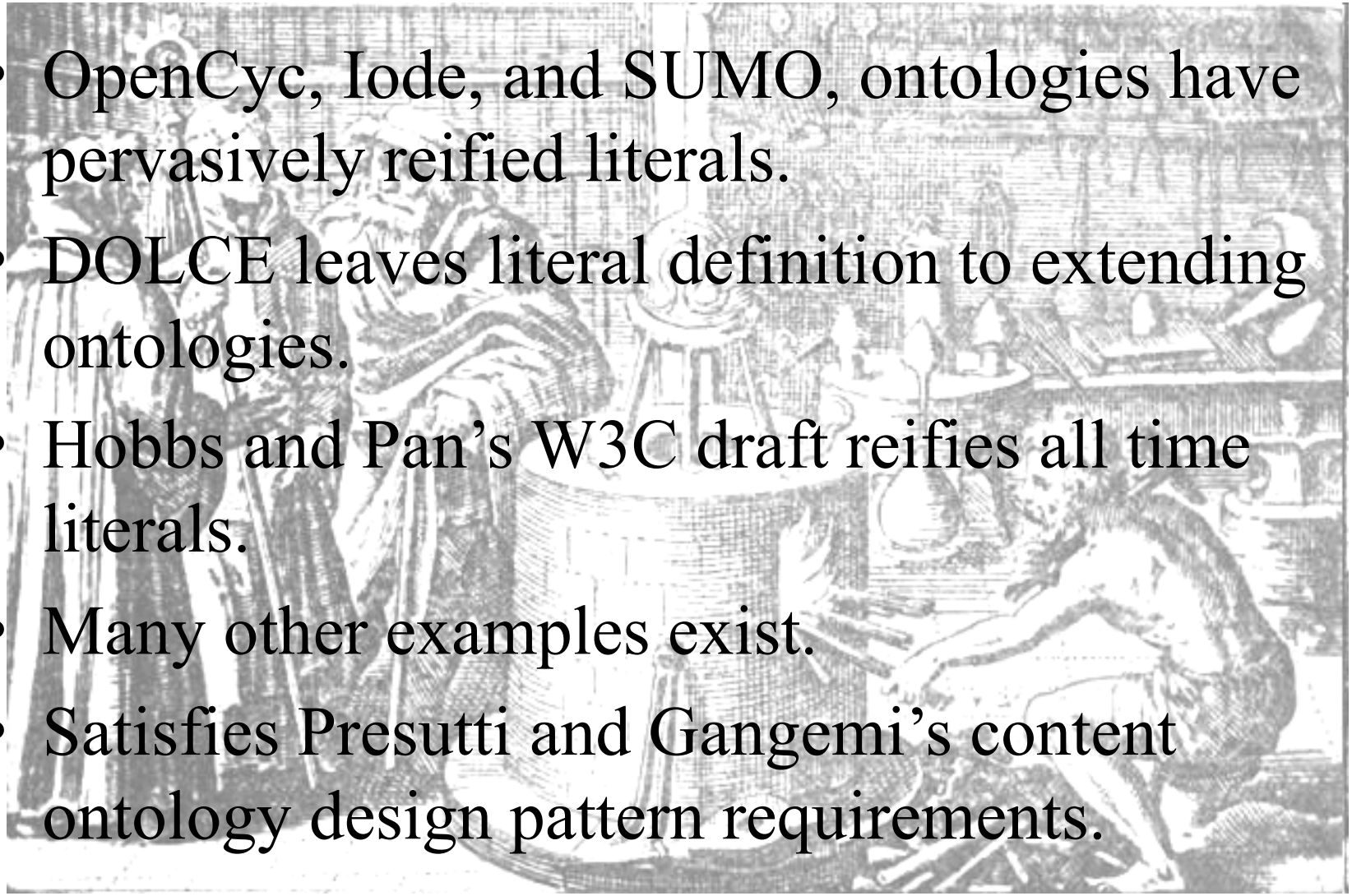
# Introduction

- Reified Literals are clearly expressive
- Reified Literals appear
  - Wasteful in memory
  - Slow at query time
  - More Complex



Global InfoTek, Inc.

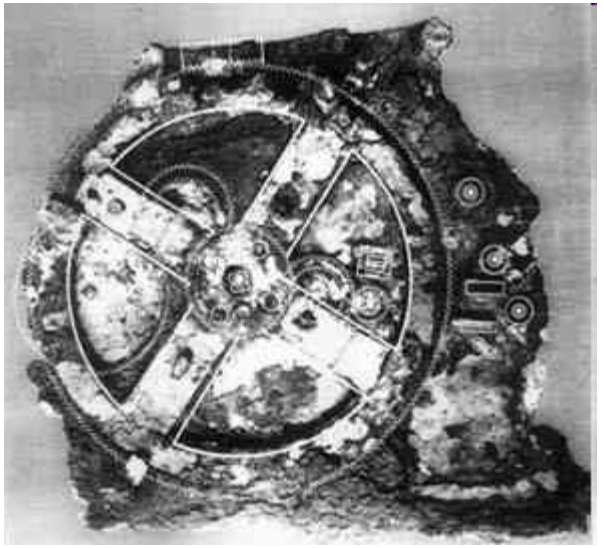
# Current Practice, Related Work



- OpenCyc, Iode, and SUMO, ontologies have pervasively reified literals.
- DOLCE leaves literal definition to extending ontologies.
- Hobbs and Pan's W3C draft reifies all time literals.
- Many other examples exist.
- Satisfies Presutti and Gangemi's content ontology design pattern requirements.

Global InfoTek, Inc.

# Contributions

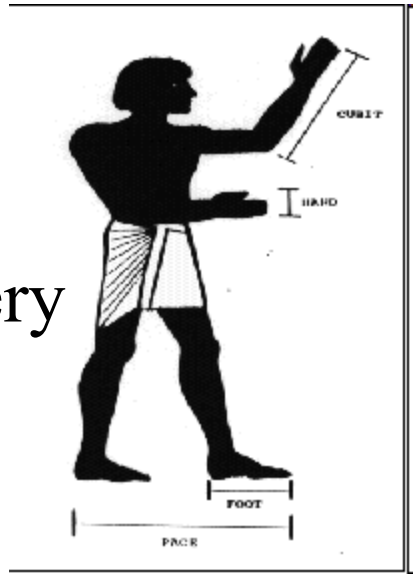


- Reified literals are not new.
- We characterize the pattern's virtues and cost.
  - We address common misconceptions about this design pattern: memory footprint cost, query speed, and design complexity.
  - We give examples of the pattern's expressivity.
- We show how to apply theses analyses to all literals.

Global InfoTek, Inc.

# Methods and Metrics

- Memory usage comparisons count triples.
- Query speed metrics count BGP's (query clauses).
- Query speed metrics note that all BGP's are not equal
  - Equijoins may be faster that string comparisons.
  - Partial string matching can dominate the query
- We use a qualitative comparison for the relative complexity of reified literals.

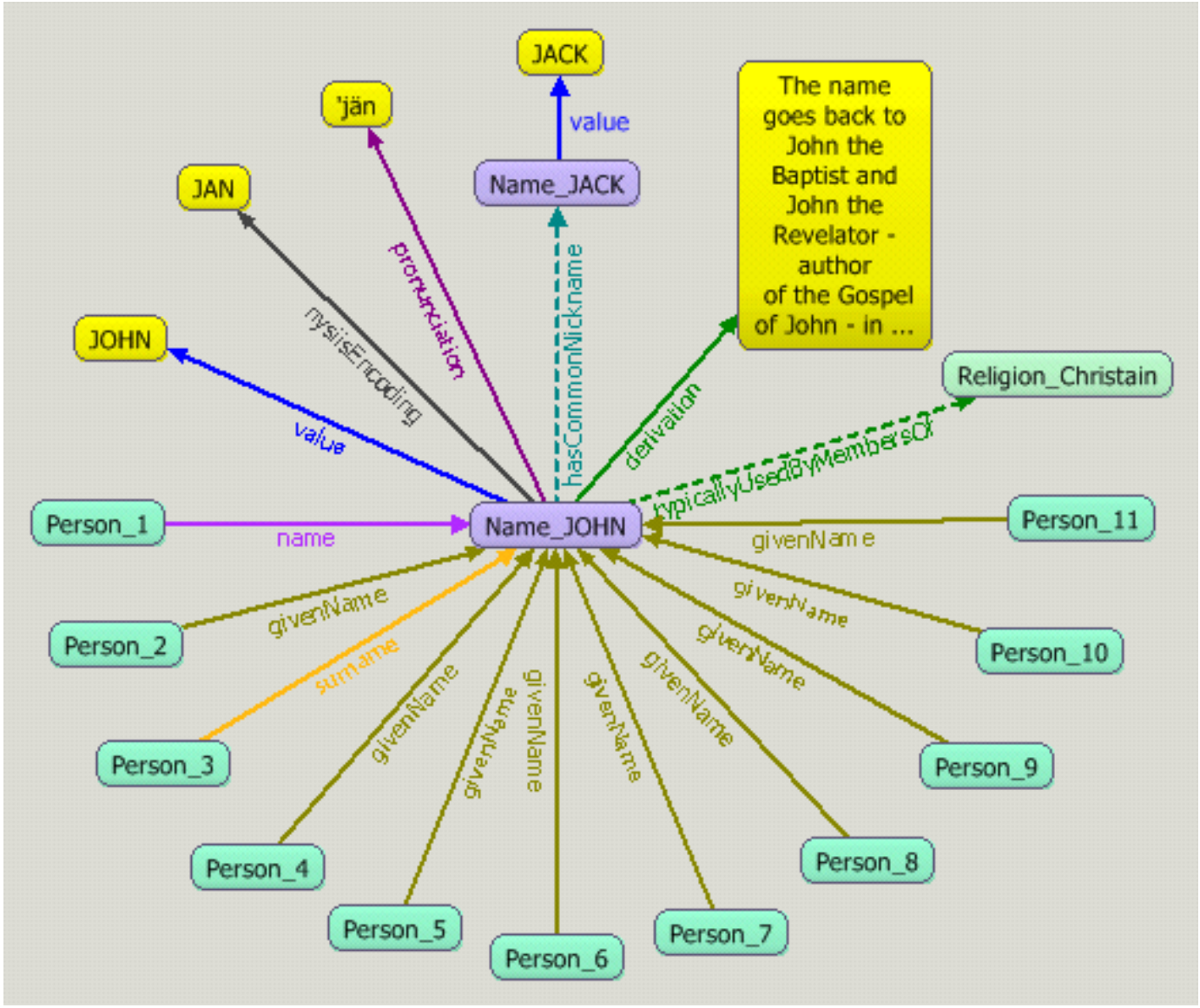



# Case Study: Reified vs. Non-reified Names

- Memory Waste:
  - If name is common, cost is amortized.
  - Net reified name cost is a function of the average number of references to the names.
- Query Speed for Exact Name Matching:
  - Queries need not be string based.
  - Implementations can use integer comparison.
  - Can be equal to or faster than un-reified matching.
- Query Speed for Inexact Name Matching
  - It requires an extra join when using reified names.
  - This increase may be inconsequential.

Global InfoTek, Inc.

# Reified Names (Parsimony and Expressivity)

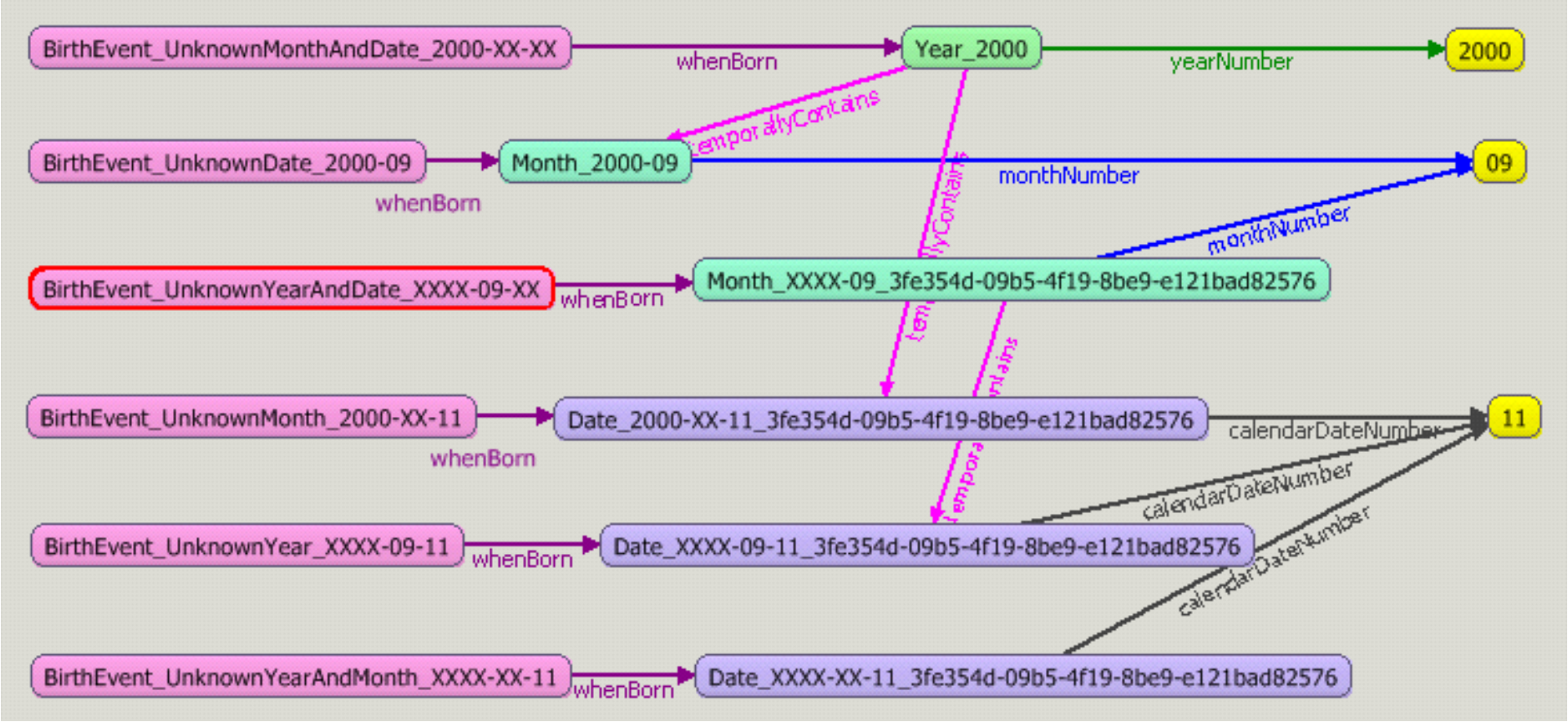



# Extending Name Reification Results

- Reified Date Literals
  - Many references per date (negligible cost)
  - May have same speed increase for exact matching.
  - Range queries require additional equijoin.
  - Huge expressivity boost.
- Reified Heights and Weights
  - Many references point to few heights and weights.
  - Exact matching less important.
  - Range queries require additional equijoin.
  - Moderate expressivity boost.

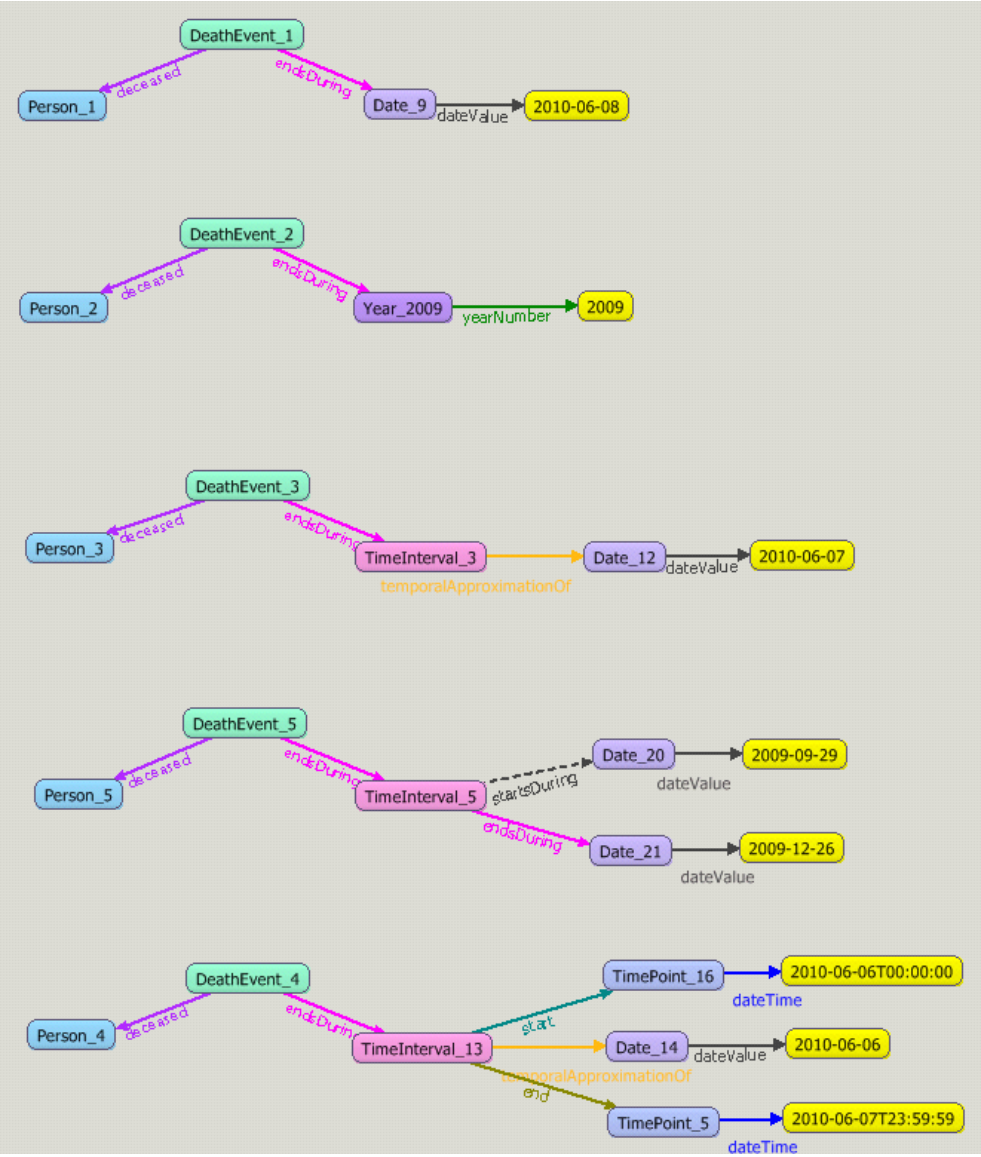
Global InfoTek, Inc.

# Reified Partial Dates



Global InfoTek, Inc.

# Approximate Dates




# Generalization of Analysis

- These analyses apply to all twenty five of our twenty five literals.
- All our literals are inherently sharable and, therefore, offer the same memory cost amortization potential.
- All but four date types and four scalars call for using exact match and partial match.
- The reified name analysis directly applies to these 17 literals.
- The two other scalar types behave just as height and weight with the range queries that are slower by one equijoin.
- The other date types are *Month, Year* and *TimePoint.* They are all simply date-like time intervals of various sizes, so their analyses are comparable to *Date.*


# Results

- Common reified literals have comparable in-memory cost to non-reified names.
- Query speed for exact matching of reified literals is at least as fast as non-reified names.
- Inexact matching is slower by an equijoin.
- We claim that the overall structure of reified names and their metadata is simpler.


# Reification Rules of Thumb

- Rare reified literals are individually costly, but the net cost is only a concern if there are very many rare types.
- Range queries such as with reified scalars and dates are slower by an equijoin.
- Inexact match queries over reified literals are slower by an equijoin. That equijoin is inconsequential on systems where inexact match dominates the query time.
- Otherwise the speed and memory cost is comparable.

Global InfoTek, Inc.

# Discussion

- We value expressivity
  - We found most of our literals to be reasonably strong candidates for reification.
- Our desire for expressivity also makes us less concerned about amortization.
- We found all of our scalars to be weaker/marginal candidates.
- We have used literal reification for over fifteen years in the IC and in two different data integration projects *at scale*.

# Conclusions

- Commonly referenced reified literals come at little or no significant cost in memory, speed, or complexity.
- Queries over such literals are never slower than the cost of one join with respect to unreified literals and are usually comparable.
- Where literal-related expressivity is needed or expected, reified literals should be considered.