

# Semantic Data Management - RDB and Graph-based Approaches

***Orri Erling***

*Program Manager, Virtuoso*





# State of the Art

---

- n Most RDF applications store RDF
- n Relatively less on the fly mapping of RDBs to RDF
- n SPARQL is becoming a full query language with 1.1
- n Scalability is getting there



# Mapping RDBs vs. Storing RDF

---

- n Map for publishing existing assets, e.g. product catalogs for presentation as linked data
- n Mapping works if not too many RDB's, few inter-RDB joins
- n For high numbers of sources, high overlap of schema between sources, high variability of data, storing RDF is best



# What More is Needed?

---

- n Agility will be better, but ...
- n Make RDF affordable, not much more expensive to run than the equivalent relational
- n We have won when:
  - n RDF is mainstream for ad hoc analysis
  - n Reuse of open data and vocabularies is a matter of course



# Factors of Performance

---

- n It is all about parallelism and locality
- n Disk kills, make data compact to run from memory
- n Network latency is almost as bad as disk:  
Ship functions to data, maximize asynchronicity
- n ACID has a bad name but need not kill, use intelligently



## Factors of Performance (cont'd)

---

- n With RDF, the key-value store idea of storing related data in a semi-structured chunk does not apply: Keys will in practice be spread all over the cluster, application - conscious data sharding is antithetical to RDF's value proposition
- n Eventual consistency with application semantics and distributed data will be hard



# Column Store, Column Per Predicate?

---

- n *Daniel Abadi's* experiments would come into their own if all this were transparent and automatic
- n RDF is quads, not triples
- n The baby goes with the bathwater if *a priori* strict schema, triples only is enforced
- n Column store techniques work if you make the row id a 3 part key instead



## Column Store, Column Per Predicate? (cont'd)

---

- n From this, you can project binary columns as needed for better space efficiency and locality
- n Do it automatically, no DBA will figure it out
- n Cache intermediates: MonetDB's cracking is an idea, but now adapt it to random access workloads and make it run distributed
- n Burn the candle from all ends: MonetDB's ROX is a promise but make it run distributed and do inference, recursion, aggregation too.



# Online vs. Analytics

---

- n RDB's have differentiated
- n We see online and analytics workload profiles arising with RDF also
- n Collaborative web and metadata are online apps with low transaction requirements
- n Highly transactional apps make no sense with RDF, but online ones do
- n Dealing with long running queries online? Anytime partial results? High per-query investment in building *ad hoc* indices, large hash join tables, &c may be a problem Make them incremental if can.



# Benchmarks

---

- n Lack of of most basic QL features in SPARQL 1.0 has produced lame benchmarks
- n This is not the authors' fault, need for interoperability made it so
- n The excuse is now gone, we must do better



# Social Web Benchmark

---

We propose:

- n Scalable synthetic data set for social web: Users make friends, post blogs and comments, tag things, send messages, construct personalized feeds
- n Online mix: Posting and reading messages, searching for people, dashboard views
- n Analytics mix: Identifying trends, graph traversal, page ranks, similarity, identity resolution, recommendation
- n Options for including full text and geospatial features



**[openlinksw.com/virtuoso](http://openlinksw.com/virtuoso)**