Hexastore: Sextuple Indexing for Semantic Web Data Management

C Weiss, P Karras, and A Bernstein

Presented by Gregory Todd Williams
Introduction

• Discuss existing triplestore approaches to storage and indexing

• Present Hexastore, an indexing scheme for RDF triples

• Show significant advantages of using Hexastore in query answering with both real world and synthetic data
Hexastore: A Full Index

- Benefits of vertical partitioning without treating predicates as special
- Benefits of multiple (full!) indexing: Materialization of all possible access schemes for a triple pattern
- No NULLs or arbitrary cyclic ordering decisions to worry about
Storing RDF: Existing Approaches

- **Triple tables**

- **Property tables**

- **Column stores (vertical partitioning)**
Indexing: Previous Work

• [Harth05, Wood05] proposed multiple indexing on \{s,p,o,g\}
  • Six indices that cover all 16 quad access patterns.
  • Only indexes cyclic triple orderings: \{s,p,o\}, \{p,o,s\}, \{o,s,p\}
• [Abadi07] - Vertical partitioning for RDF
  • Each property table sorted by subject
Hexastore's Six Indices

- All six triple pattern access schemes:
  - spo
  - sop
  - psq \approx \text{traditional vertical partitioning}
  - pos
  - osp
  - ops
Index Example: SPO

Triples:
- $s_1, p_1, o_1$
- $s_1, p_1, o_2$
- $s_1, p_1, o_7$
- $s_1, p_3, o_{10}$
- $s_1, p_3, o_{17}$
- $s_1, p_3, o_{30}$
- $s_1, p_8, o_3$
- $s_1, p_8, o_5$
- $s_1, p_8, o_9$
Index Example: PSO

Triples:
S₁, p₃, o₁₀
S₁, p₃, o₁₇
S₁, p₃, o₃₀
S₆, p₃, o₃
S₆, p₃, o₈

p₃

S₁  ...  S₆

o₁₀
o₁₇
...

S₆

o₃₀

o₃

o₈
Data Sharing

• Terminal node lists can be shared
  • Ex: SPO and PSO share same object list
  • Each S, P, O only has one terminal node list
• Total size of index is at most five (not six) times the size of a triples table
Hexastore key points

• Features
  • Dense, efficient storage (no NULLs, no table scans)
  • Efficient pair-wise joins
  • (Mostly) efficient path queries

• Challenges
  • Index size
  • Slow updates
Evaluation

- Hexastore compared to two variants of vertical partitioning of [Abadi07]: (PSO) and (PSO, POS)

- Two data sets/query sets used
  - Barton library data
  - Lehigh University benchmark

- Hexastore generally gives 1-3 orders of magnitude performance improvement
Example LUBM Query

SELECT ?person WHERE {
    <AssociateProfessor10> :teacherOf ?course .
} ORDER BY ?course

• For triple pattern { ?person ?p ?course }
  • COVP (PSO) is worst possible index
  • COVP2 (POS) is better, but must still scan all predicates
  • Hexastore uses OSP to access data directly
Example Query Results

LUBM Query 4

Query response time (seconds) vs. Number of Triples
Conclusion

- Hexastore combines benefits of vertical partitioning, multiple indexing
- Predictable tradeoff between storage size and time-efficiency
- Shows significant benefits in query answering (several orders of magnitude improvement)
Big Challenge

• Handling named graphs (SPARQL)

• 24 access schemes for \{s, p, o, g\} (Icositetra?)

• Probably unacceptable memory and load time requirements
Questions?

