NSWI144 – Linked Data – Lecture 10 – 17 December 2012

Querying and Indexing

Martin Svoboda

Faculty of Mathematics and Physics

Charles University in Prague

Outline

- Querying systems
  - **General issues**
  - **Existing approaches**
    - Relational databases
    - Native solutions
  - **Common practices**
  - **Open problems**

Querying Systems

- Issues
  - Data structures
    - **Storage**
    - **Indices**
  - Algorithms
    - **Query processor**
- Problems
  - Data **scalability**, **distribution** and **dynamicity**

Querying Systems

- Architecture
  - **Local**
    - Efficient processing
    - Independent data
    - Storage requirements
  - **Distributed**
    - Runtime requests
    - Up-to-date data
    - Network throughput

Querying Systems

- Use cases
  - **Querying**
    - Local or distributed data
    - Structural queries
    - Complete results
  - **Searching**
    - Global data cloud
    - Full text queries
    - Imprecise results

Querying Systems

- Storages
  - **Relational databases**
    - Decades of research results
    - Plenty of implementations
    - Usage of standard indices
    - Queries translated into SQL
  - **Native approaches**
    - Novel ideas and approaches
    - Based on graph logical model

Relational Databases

- Approaches
  - **Triple table**
    - One huge table of all triples
  - **Binary tables**
    - One table for each predicate
  - **Property tables**
    - Set of tables for clustered predicates

Relational Databases

- **Triple table**
  - Idea
    - One huge table with all triples
    - Each triple is represented as one row
  - Schema
    - Table(Subject, Predicate, Object)

Subject	Predicate	Object
S1	P1	O1
S1	P1	O2
S2	P2	O2

Relational Databases

- Triple table
  - **Advantages**
    - Without NULL values
    - Support of multi-value properties
    - Symmetric access patterns
  - **Disadvantages**
    - Inappropriate for less selective queries
    - Problematic self joins

Relational Databases

- **Binary tables**
  - Idea
    - Separate table for each predicate
  - Schema
    - $Predicate_1(Subject, Object)$
    - …

P1	
Subject	**Object**
S1	O1
S1	O2

P2	
Subject	**Object**
S2	O2

Relational Databases

- Binary tables
  - **Advantages**
    - Support of multi-value properties
    - Suitable for column storages
  - **Disadvantages**
    - Queries with unbound predicates
    - Large number of tables
    - Asymmetric access patterns

Relational Databases

- **Property tables**
  - Idea
    - Combining all/some properties for a given subject
      - Disjoint or potentially overlapping sets of properties
  - Schema
    - $Table_1(Subject, Predicate_1, Predicate_2, \ldots)$
    - …

Subject	P1	P2
S1	O1	*NULL*
S2	*NULL*	O2

Relational Databases

- Property tables
  - **Advantages**
    - Fewer join operations
    - Follow relational model
  - **Disadvantages**
    - Clustering is not trivial
    - Do not support multi-value properties
    - Potentially very sparse
    - Asymmetric access patterns

Native solutions

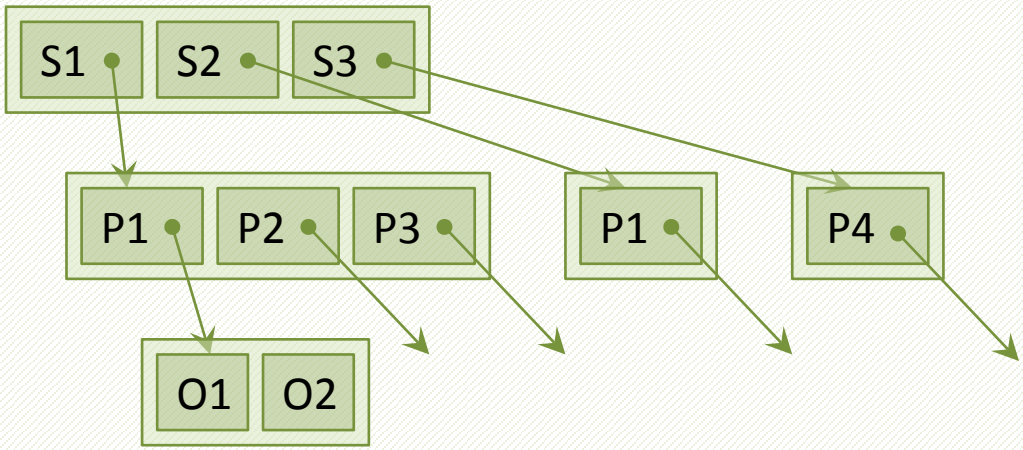
- Approaches
  - **Sextuple index**
    - Ordered nested lists for local data
  - **BitMat index**
    - Compressed matrix slices for local data
  - **Data summaries**
    - Q-trees for distributed data

Sextuple Indexing

- Paper
  - Cathrin Weiss et al.: **Hexastore: Sextuple Indexing for Semantic Web Data Management**
- Overview
  - **Local database** of RDF triples
  - Based on **ordered nested lists**
  - Supports all access patterns

Sextuple Indexing

- **Index model**
  - SPO, SOP, OSP, OPS, PSO, POS nested lists
  - Duplicated lists are shared
    - For example: P lists for corresponding SO and OS

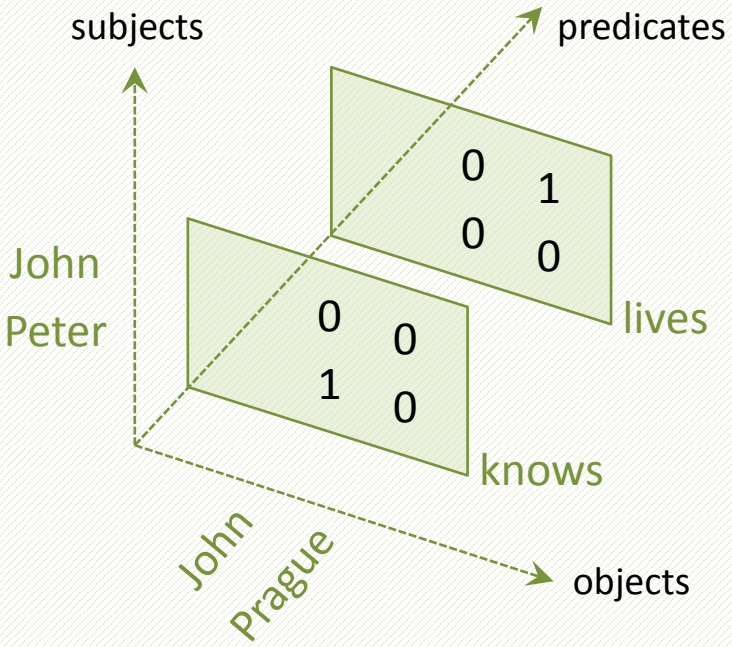


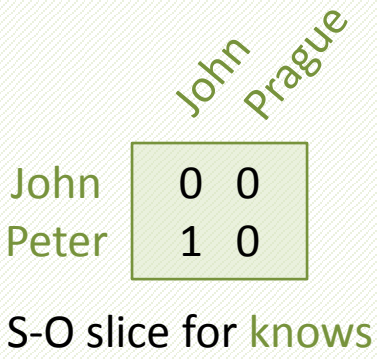
BitMat Index

- Paper
  - Medha Atre et al.: **Matrix "Bit"loaded: A Scalable Lightweight Join Query Processor for RDF Data**
- Overview
  - **Local database** with RDF triples
  - **Low selectivity** of conjunctive queries
    - Individual patterns and joining as well
  - **Low memory requirements**
    - Without intermediate result materialization

BitMat Index

- **Index model**





BitMat Index

- Index model
  - **Terms**
    - Active domains of all subjects, predicates and objects
    - Assignment of **unique integer identifiers**
  - **Index**
    - **3-dimensional matrix** with bit values 0 or 1
    - Dimensions for subjects, predicates and objects
  - **Slices**
    - SO and OS for each predicate
    - PO for each subject and PS for each object

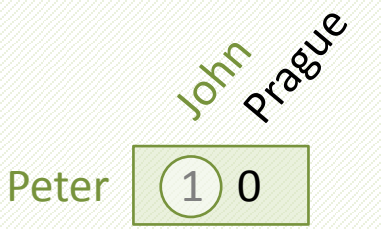
BitMat Index

- Index implementation
  - Ordinary file
  - **Compression**
    - Slices are stored per individual rows
    - Rows are compressed using bit runs
  - Query evaluation
    - All operations over compressed bit runs

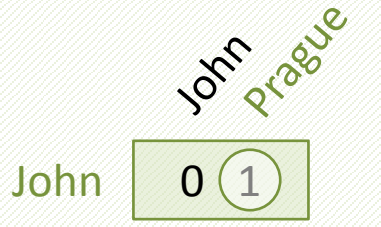
BitMat Index

- Query processing
  - **Initialization**
    - Loading required index components
  - **Optimizations**
    - Pruning of loaded index components
  - **Joining**
    - Stream joining of individual patterns
    - Based on nested loops algorithm
      - Idea from relational databases
      - Starting with the most selective pattern

(Peter knows ?P)
(?P lives Prague)



S-O map for knows



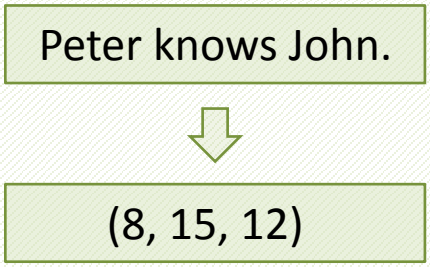
S-O map for lives

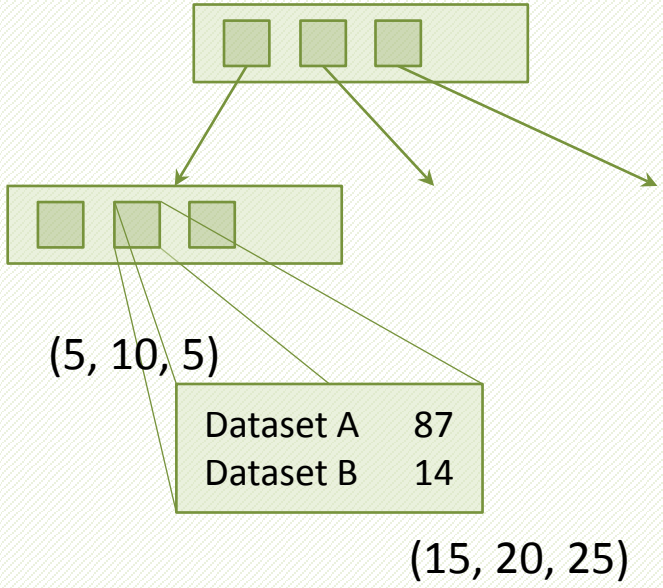
Data Summaries

- Paper
  - Andreas Harth et al.: **Data Summaries for On-Demand Queries over Linked Data**
- Overview
  - **High number of distributed datasets**
  - Conjunctive SPARQL queries
  - **Source selection algorithm**

Data Summaries

- Index model
  - **Numeric space**
    - 3-dimensional numeric space
    - Transforming triples to points
    - Based on hash functions





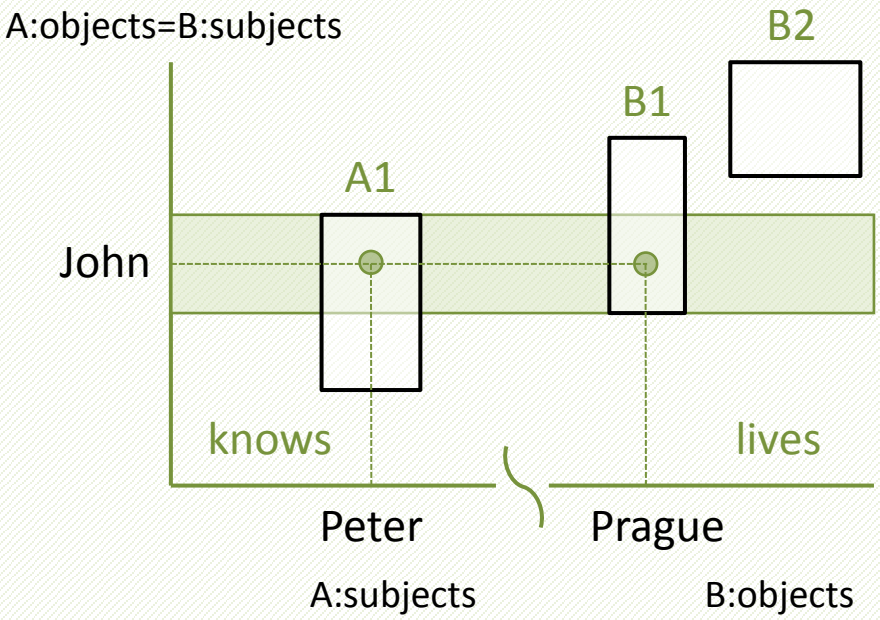
Data Summaries

- **Q-tree**
  - Based on R-trees and histograms
  - **Internal nodes**
    - Set of **bounding boxes**
      - These boxes may generally overlap themselves
  - **Leaf nodes**
    - **Buckets with summaries**
      - For each dataset a number of its corresponding triples
  - Features
    - Fixed (parameterized) size

Data Summaries

- Algorithm
  - Query transformation
    - Intervals for variables
  - **Source selection**
    - **Individual patterns**
      - Index traversal
      - Sets of buckets
    - **Inductive joins**
      - Required overlapping
  - **Query processing**

(Peter knows ?P)
(?P lives Prague)



Search Engines

- Approaches
  - **Swoogle**
  - **SWSE**
  - **Sindice**

Search Engines

- **Swoogle**
  - http://swoogle.umbc.edu/
  - Paper
    - Li Ding et al.: **Swoogle: A Search and Metadata Engine for the Semantic Web**
  - Idea
    - Search engine for **semantic documents**
      - Data documents / ontologies and schemata
    - Provided functionality
      - Document metadata
    - Based on IR techniques

Search Engines

- **SWSE**
  - **Semantic Web Search Engine**
  - http://swse.deri.org/
  - Paper
    - Andreas Harth et al.: **SWSE: Answers Before Links**
  - Idea
    - Search engine for RDF quads (triples with context)
    - Provided functionality
      - Keyword matching
      - Concept filtering

Search Engines

- **Sindice**
  - http://sindice.com/
  - Paper
    - Eyal Oren et al.: **Sindice.com: A Document-oriented Lookup Index for Open Linked Data**
  - Idea
    - Search engine for semantic documents
    - Provided functionality
      - Keyword matching
      - Inverse functional properties

Common Observations

- **String compression**
  - Repeating string values
    - URI references and literals
  - **Unique integer identifiers**
    - Efficient processing
    - Space requirements
  - **Translation maps**
    - Both directions
    - Based on B-trees

Common Observations

- **Data pruning**
  - Idea
    - Query optimization
    - Relevant data
  - Methods
    - **Filtered selections**
    - **Ordering of joins**
  - Problem
    - We have only incomplete knowledge…

Open Problems

- **Data distribution**
  - Motivation
    - **Datasets are distributed**
  - Problems
    - Network drawbacks
    - Independent datasets
    - Space requirements

Open Problems

- **Data scalability**
  - Motivation
    - **Web of Data size explosion**
      - September 2011:
      - 295 datasets, 31 billion triples, 504 million links
  - Problems
    - All so far discussed issues…
    - … and questions of **quality, provenance or trust**

Open Problems

- **Data dynamicity**
  - Motivation
    - Data often change through time
      - New triples are added, existing ones are modified, …
  - Problems
    - Management of updates
    - Dynamic structures

Conclusion

- Querying systems
  - Local / distributed approaches
  - Relational databases / native approaches
  - Common practices
  - Open problems