Querying Big Data

Boris Novikov

Dept. of Computer Science
Saint Petersburg University
Saint Petersburg, Russia

The Talk Outline

- Introduction: What are Big Data?
- The Algebra
  - Similarity and Keyword Search
  - Filtering and Calibration
  - Advanced search: semi-joins
  - Objects or Facts?
- Implementation
  - Optimization and Evaluation
  - Approximate Evaluation
  - Optimization Issues
  - Adaptive processing
- Open issues
- Conclusion

Introduction

Big Data: It's Not Just running the Analytics on the cloud

H. V. Jagadish

What are Big Data

What is BIG?

- Volume
  - Huge amounts to be stored and processed
- Variety
  - Models and types
  - Distributed, heterogeneous, and autonomous
- Velocity
  - Real time

What is hard?

- Acquire
- Extract
- Find, combine, confront, and sort
  - Complex search
  - Uncertainty
  - (Near) duplicates
- Make sense of it
  - Complex analytics

Why to Query?

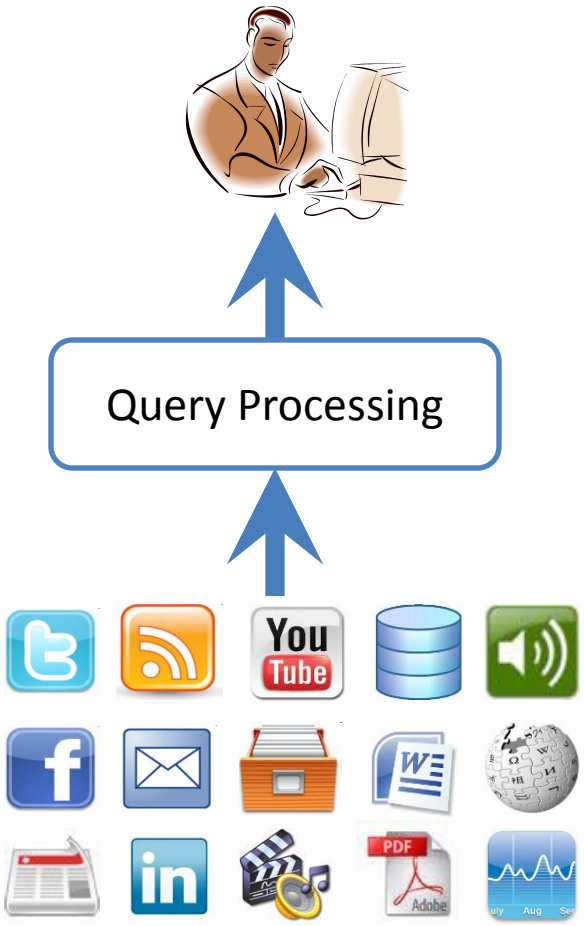
Manage data via querying

Query languages:

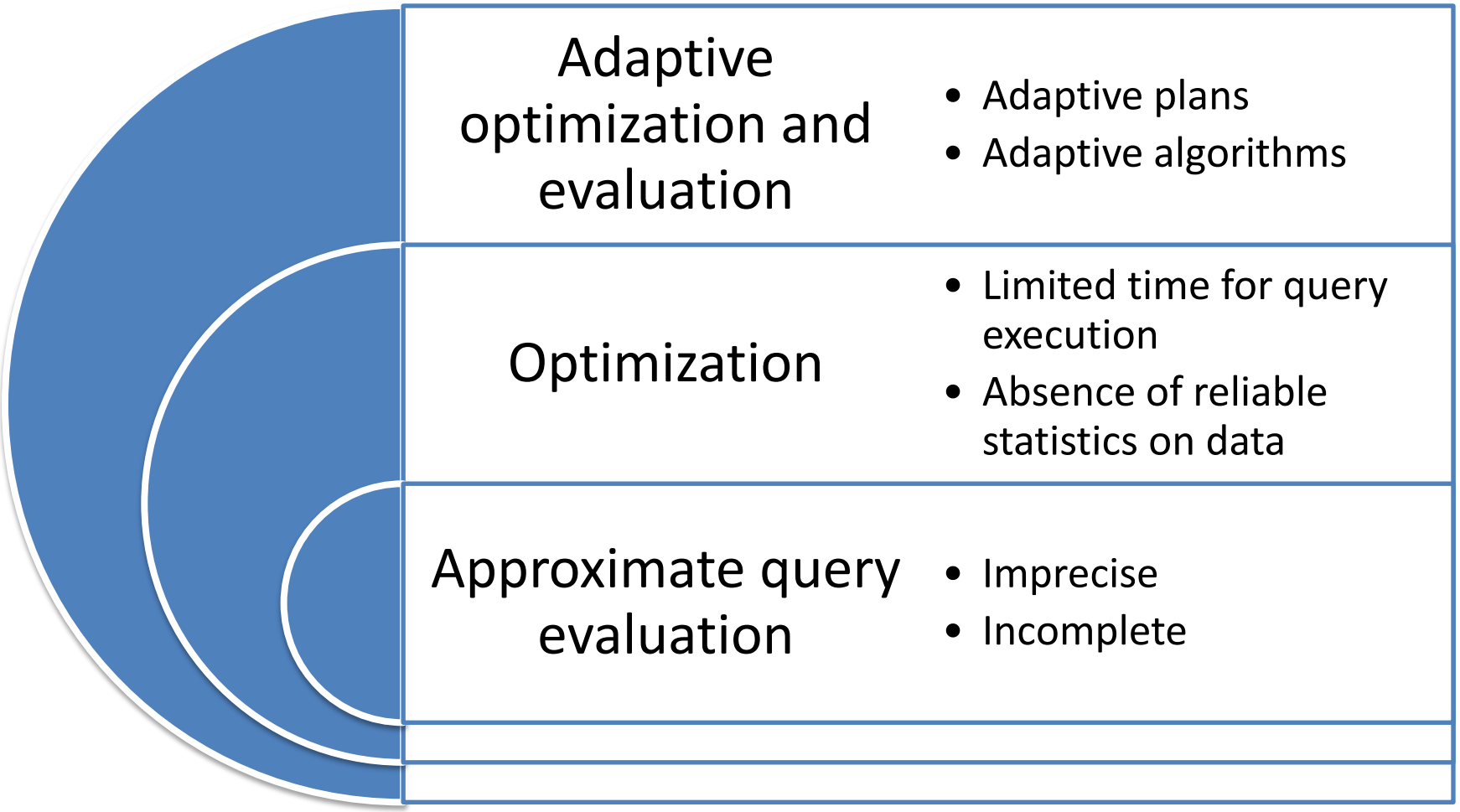
- are highly expressive
- are declarative
- can be efficiently implemented
- powerful optimization

The Dimensions of Variety

Data Models	• Structured (Databases) • Semi-structured • Unstructured (Text, Multimedia)
Dynamics	• Stable (DBs, Web) • Highly dynamic (Streams)
Querying paradigm	• Exact • Uncertain • Similarity
Schemas	• Prescriptive • Descriptive • Imposed



The Velocity: How to Address



Big Data are here for Quite While

- Integration
  - Schema mappings
    - Queries based on a global schema
  - Transactions
- Data spaces
  - Schemaless
  - Search languages
- Scientific data
  - Volume
  - Complex processing
- Mining in social networks
- Linked Data

Query Processing: Objectives

- Variety
  - Extensibility
  - Build on top of data sources, not instead
    - Rely on query processing capabilities
    - Avoid unneeded detailed schema mappings
- Volume
  - Parallel/distributed computations
- Velocity
  - Approximate evaluation
  - Optimization
    - Adaptive
    - Performance trade-offs

The Algebra

Preliminaries

Concepts

Keyword Search

Calibration

Joins & Semi-Joins

The Ternary Model

<object, attribute, value>

+ Universal
+ Flexible
+ The base for RDF etc.
– Too fine-grained
– Well-known as a perfect slow-down tool
– Does not capture uncertainty

Data Spaces

- Variety of containers
- Schemaless
- Objects
  - Attributes
  - Nesting of attributes
  - Identity?
- Search:
  - Keyword
  - (Implicit) links
- Similarity/uncertainty

Object DBs failed to meet expectations. Why?

- Conceptually relations are constants
- Objects are mutable, have identity and state
- Semantics of relational calculus is crystal
- Semantics of objects calculi depends on behavior
- Bulk processing vs. unary processing

Identity

- Based on natural properties
  - Paper title and list of authors
  - Fingerprints
- Surrogates: external and internal
  - ISBN
  - DOI
- Based on location
  - URL, URI

Similarity and Distance Basics

- Express application semantics (relevance, proximity, etc.)
- Used interchangeably
- Examples: Euclidian, l_p, cosine

Similarity

- Usually

  S(a,a) = 1

  0 < S(a,b) < 1

Distance

- Semi-metrics

  d(a,b) >= 0

  d(a,b) = d(b,a)

  d(a,a) = 0

- Metrics

  d(a,b)+d(b,c) >= d(a,c)

Fuzzy Queries and Results

- Query examples

  Find recent publications on query processing and optimization

  Find historical buildings in walking distance from the conference venue

- The relevance of an object is estimated with a score
- Scores may be calculated as similarity between an object and the query

The Algebra

Preliminaries

Concepts

Keyword Search

Calibration

Joins & Semi-Joins

Q-Sets

- A set of objects with scores over a base set
- Minimalist's schema: (data definition)
  - Query-driven
  - Base sets represent object types but do not impose structure
  - Objects
    - Strong identity
    - Attributes
    - Set-valued attributes
    - Nesting
- Q-sets may be viewed as fuzzy sets (scores in [0,1])

What is Special with Q-Sets?

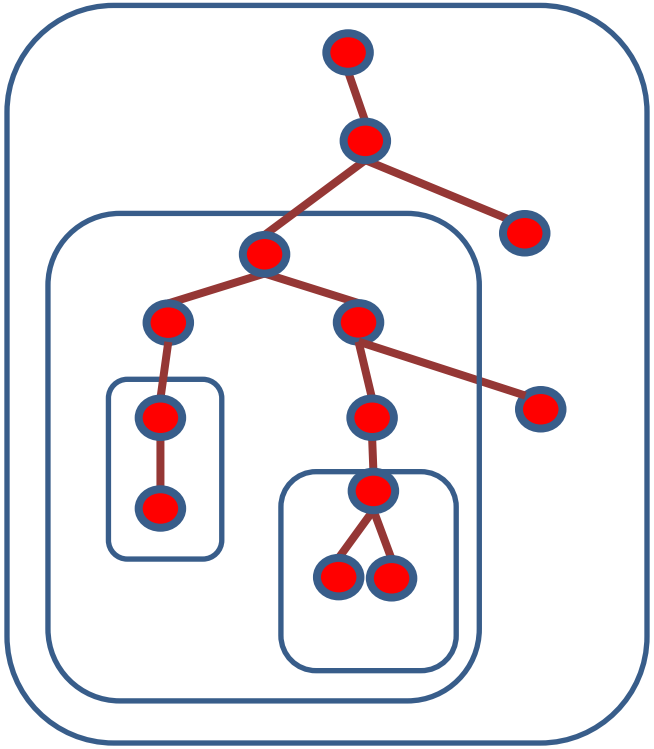
- A Q-set represents a query together with the result of evaluation
- Queries are abstract : language, formula, or algorithm are not required to be known
- A score can be considered as an abstract similarity between an object in the Q-set and the query

Mapping the Variety to Q-sets

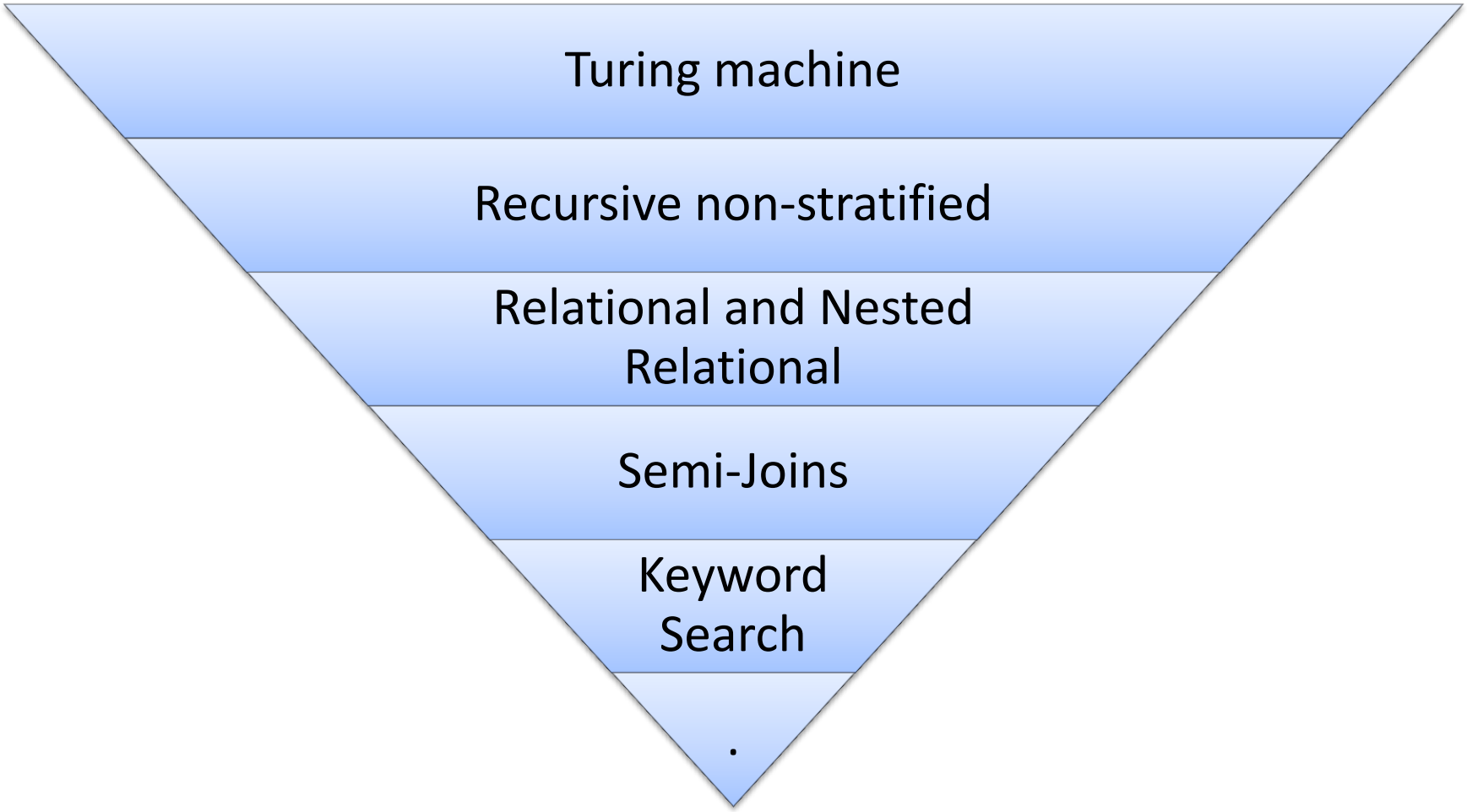
- Exact (database) queries are represented with discrete membership function (scores in {0,1})
- Ranked/scored lists from search engines
- Scores may express
  - Similarity
  - Proximity
  - Relevance
  - Confidence
  - Probability

Combining Q-Sets

- The goal is to build complex queries from queries represented as Q-sets
- The scores in different Q-sets are supposed to be comparable
- Meaningful interpretation of the output scores is essential



The Expressiveness of Query Languages



Implementing Query Languages

- High-level declarative (calculus, QBE-style, graphical, etc.)
- **Logical algebra**
- **Physical algebra(algorithms)**
- Interpretation of physical algebraic expressions

The Algebra for Q-Sets

- Primary
  Extract from a primary data source
- Filters
  Anything done on per-object bases
- (Fuzzy) set-theoretic operations
  Arguments with same base set
- (Fuzzy) joins and semi-joins
- (Fuzzy) Aggregations
- Nest/Unnest
- Group joins
  A combination of join and aggregation

Making it Formal

Specifying an expression

```
<query>
 <operation>
        <name>operation name</name>
        <parameters>
                <parameter name>value</parameter name>
        </parameters>
 </operation>
 <arguments>
        <query>sub query</query>
 </arguments>
</query>

OperationName(arg1, arg2,…,
        ParamerName1=value1, ParamerName2=value2, …)
```

An Example: Specifying a Join

```
<query>
        <operation>
                <name>left_join</name>
                <parameters>
                        <join_attr_l>attr name</join_attr_l>
                        <join_attr_r>attr name</join_attr_r>
                        <theta_name>predicate name</theta_name>
                </parameters>
        </operation>
        <arguments>
                <query>sub query</query>
                <query>sub query</query>
        </arguments>
</query>
```

Left_join(arg1, arg2, attr_l=…, attr_r=…, join_cond =…)

The Algebra

Preliminaries

Concepts

Keyword Search

Calibration

Joins & Semi-Joins

Why Keyword Search?

- Human (Easy to understand)
- Easy to interpret
- Intensively used in the context of data spaces
- Efficient algorithms
- Indexing
- Efficient (polynomial) semantic caching

Keyword and Similarity Search

- Human expectations for complex search
  - Chorus: several evidences are more trustworthy
  - Skimming: get the best from different sources
- Simple predicates
  - Exact (like in Boolean IR)
  - Similarity
- Logical operations (and, or, but not)
- Universal base set

An Example

Find inexpensive hotels located in a walking distance from CompSysTech'12 venue and having positive visitor's ratings

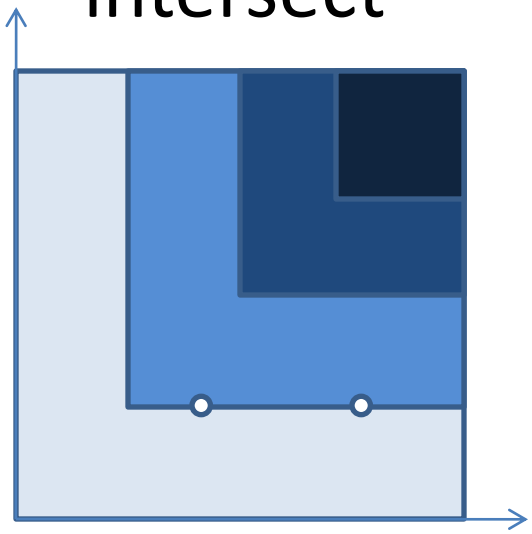
- Attributes needed for different criteria may be obtained from different primary sources
- Scores for different criteria might be incompatible

The Algebra: Set-Theoretic Operations on Q-Sets

- All operations are based on the object identity
- Merge attributes: parametric options
- In a fuzzy environment, the differences between operations are fuzzy:
  - Union max(s,u)
  - Intersect min(s,u)
  - Super-Union 1-(1-s)(1-u)
  - Super-Intersect su
  - Difference max(s-u,0)
  - More
    - Comb-sum s+u
    - CombMNZ
    - …

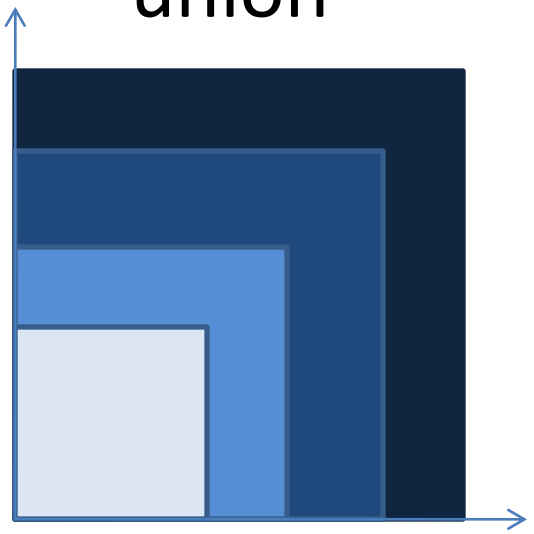
Fuzzy Union and Intersect

intersect



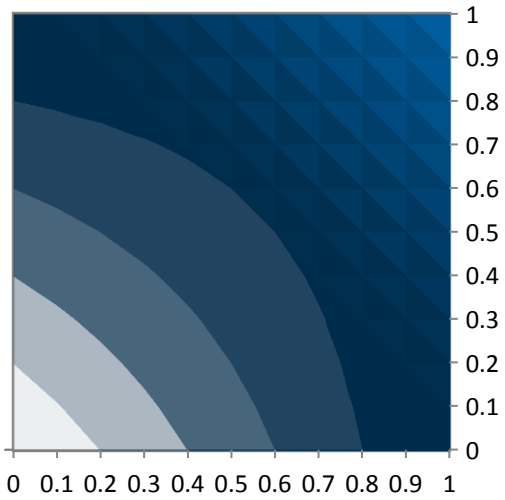
Min(s_1, s_2)

union

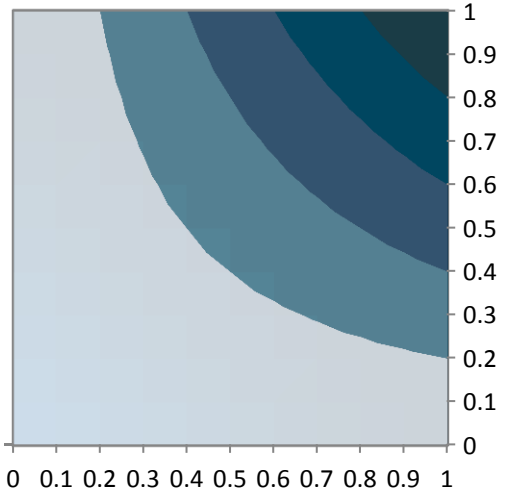


Max(s_1, s_2)

Superunion and Superintersect



Superunion
1-(1-s)*(1-u)



Super-Intersect
su

Filters

- Simple filters
  - Selection criteria (conditions on the attribute values)
  - Truncate low scores
  - Discretization
- Enrichment
  - Advanced Filters can extract implicit information from objects of Q-sets
  - Attributes of an object obtained from different sources can be merged (in binary operations)
- Calibration

The Algebra

Preliminaries

Concepts

Keyword Search

Calibration

Joins & Semi-Joins

Why to Calibrate?

Scores from different sources may vary significantly.

The calibration is needed

- To make scores coming from different Q-sets comparable
- To avoid undesirable domination of a Q-set
- To enforce domination of a better quality scores
- To take into account specific application requirements
- To adjust for personalized preferences or feedback

The High-level algebraic approach is suitable for expression of human expectations for specific applications

What is a Good Calibration?

- Criteria
  - Sensitiveness to outliers
  - Skew
  - Effectiveness
- Filter operations
  - Normalize
  - Weaken/Strengthen

Normalization

Operation	Formula	Sensitiveness to outliers	Skew
Norm-maxmin	$\frac{score(e) - \min(score(x))}{\max(score(x)) - \min(score(x))}$	-	-
Norm-avg	$similaring(\frac{distancing(score(e))}{avg(distancing(score(x)))})$	+	-
Norm-z-score	$\frac{score(e) - \mu}{\sigma}$	+	+
Normalize-score$_p$	$score(e) * B$	+	+

The Algebra

Concepts

Objects vs. Relations

Keyword Search

Calibration

Joins & Semi-Joins

Linking Objects

- Keyword search does not capture dependencies between objects
- Semi-joins: support for conditions in linked objects

  (similar to a relational semi-join which is a projection of a join on the first argument)
- Semi-join preserves the identity (of the objects from the argument Q-set)

An Example

Find inexpensive hotels located closely to congress centers capable to accommodate a large conference and reachable in at most one hour from an international airport

- Several semi-joins from different sources
- All links are implicit
- Distance calculation over the road networks

Joins and Aggregations

- Full relational power and even
- Nested-relational with Nest/Unnest
- Unfortunately, the identity is lost
- The output contains facts, rather than objects

Aggregations

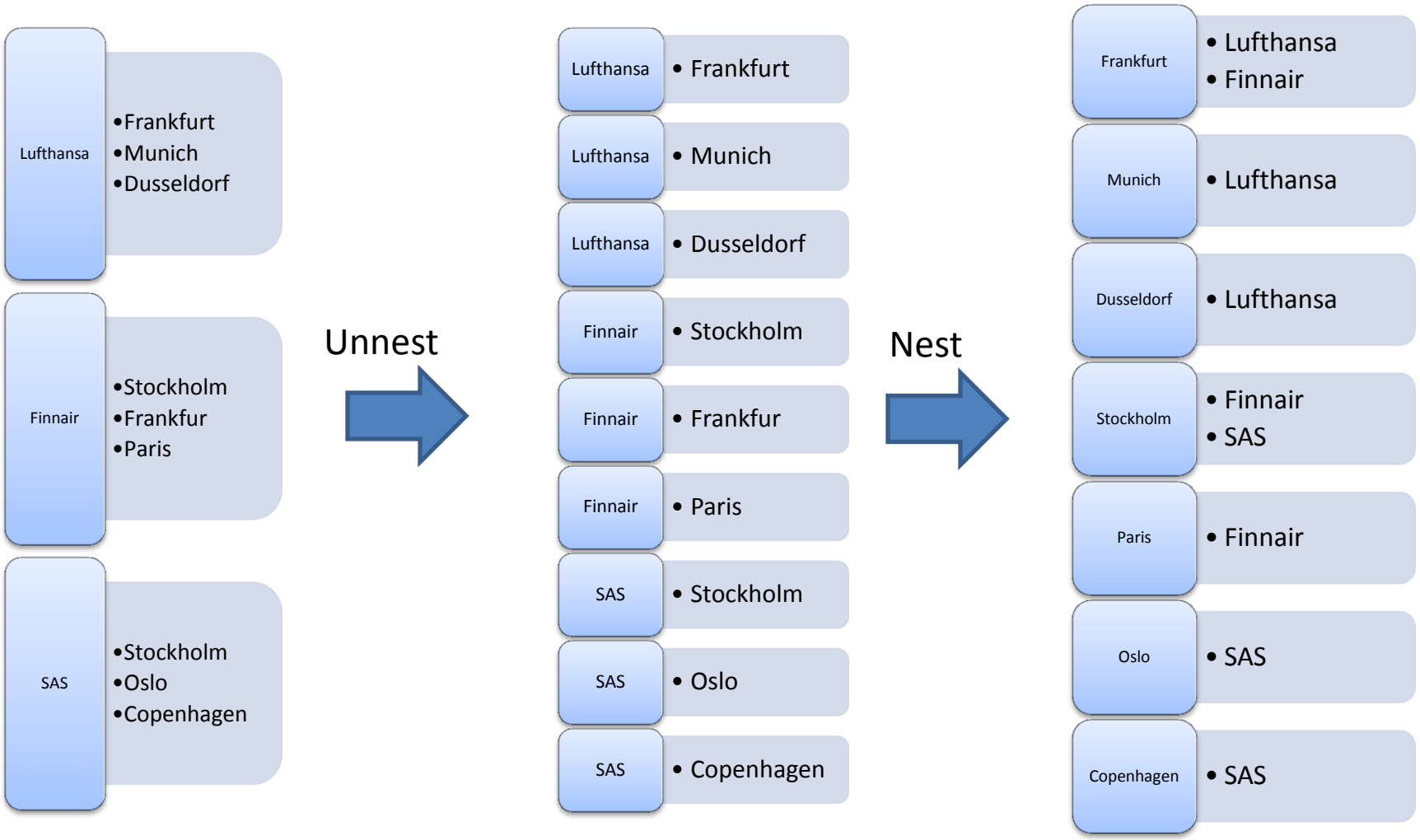
- A generalization of *SQL group by* clause
- Aggregate on grouping functions:
  - Values of certain attribute(s) (group by)
  - Clusters of objects
  - Classes of objects
  - Library functions
- Fuzzy aggregation
- A library of aggregate functions (like sum, max, avg, concat)
- Identification of aggregated objects: surrogates

Nest and Un-nest

- Nest is a special case of aggregation

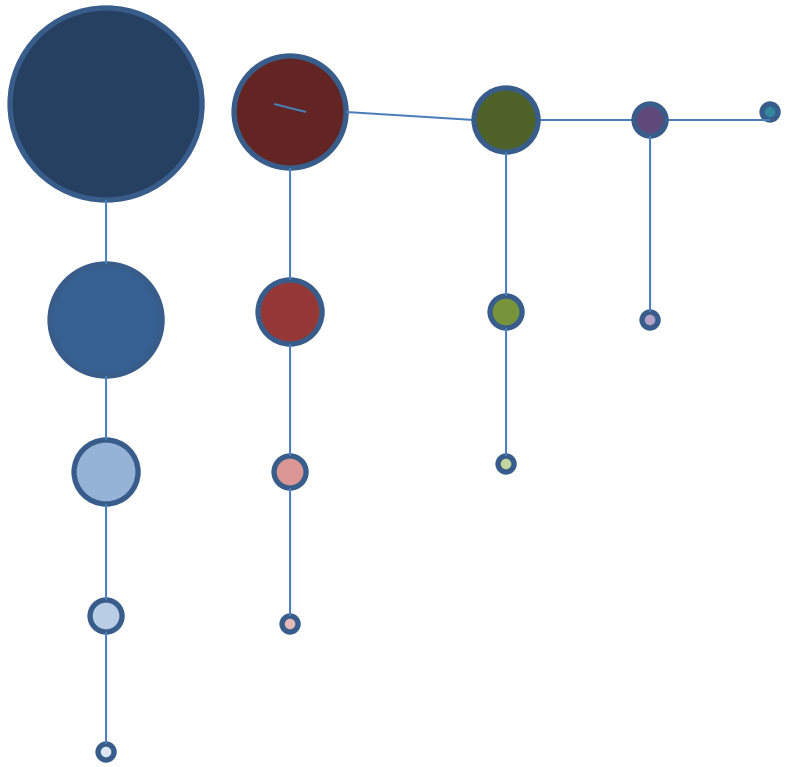
  The aggregation function constructs a set containing grouped objects together with scores
- Unnest constructs a Q-set from objects with a nested attribute
- The identity of output objects: surrogates

Nest and Unnest: Airlines



Merge Several Q-sets

- Unnest effectively produces a union of nested Q-sets
- An Example: Search in a partially annotated collection
  - Retrieve annotations (text)
  - Find similar for each annotated item
  - Unnest



The Power of the Group Join

- The group join is a generic join of Q-sets followed by an aggregation (on the identity of the first argument
- Preserves the identity of the objects in the first argument Q-set
- Algebraic properties of join are not valid
- Special cases are:
  - Semi-join
  - Aggregation
  - Join

Group Join Example

Find inexpensive hotels located closely to congress centers capable to accommodate a large conference and reachable in at most one hour from an international airport

```
Group_join(
        Group_join(
                filter(hotels, ‘inexpensive’),
                filter(congress_centers,
                        ‘capacity >500’),
                ‘walking distance’)),
        airports,
        ‘travel_time < 1 h’)
```

```
Group_join(
        filter(hotels, ‘inexpensive’),
        Group_join(
                filter(congress_centers,
                        ’capacity >500’),
                airports,
                ‘travel_time < 1 h’),
        ‘walking distance’))
```

Implementation

Optimization

Adaptive Processing

The Prototype

Optimization Issues

- Algebraic properties are not as good as for relational algebra
- Cost models for primary data sources are hardly available
- Expensive filters are hard to combine with classical join ordering optimization

Approximate Query Evaluation

- Exact evaluation is neither feasible nor meaningful for similarity-based queries
- Approximate evaluation may be
  - Incomplete
  - Imprecise
- The cost of approximate evaluation is expected to be lower
- Trade-off between the cost and the quality
- What is the quality?

Approximate Algorithms

- The approximate algorithms for operations are based on
  - Sampling
  - Thresholds
- Additional parameters are needed to control the quality of an output
- Again? What is the quality?
- The quality if a non-decreasing function of the cost for given arguments
- Extended cost models provide both cost and quality

What to Optimize? (Re-Formulating the Problem)

- Classical (exact query evaluation): Find an execution plan with minimal cost
- Balancing cost and quality
  - Guaranteed quality: Find a plan yielding at least the required quality with minimal cost
  - Limited cost: Find a plan yielding the best quality for at most given cost
  - Multi-criteria optimization

How to Optimize?

- Construct an optimal execution plan and allocate available resources to operations
  - Computationally hard
- Heuristics: build a plan and then allocate resources
  - The quality of the plan and resource allocation is controllable

Implementation

Optimization

Adaptive Processing

The prototype

Why optimization sometimes fails?

- Optimization is based on estimations rather than actual costs of operations
- Cost models cannot be precise
- Cost models may use incomplete or unreliable statistics
- Expensive predicates and filters can be unpredictable
- Data skew is hard to predict

Adaptive Execution Approaches

- Optimization and execution are mixed or interleaved
- Materialization points: re-optimize the remainder of the plan
- Routing: dynamically re-schedule objects to operations in a pipelined execution

Adaptive Processing for Q-Sets

- Materializing primary
  - Obtain reliable statistics
  - A kind of caching
- Materialization points after heavy and hardly predictable filters
- Map-reduce: materializations are done anyway
- Pipelining?

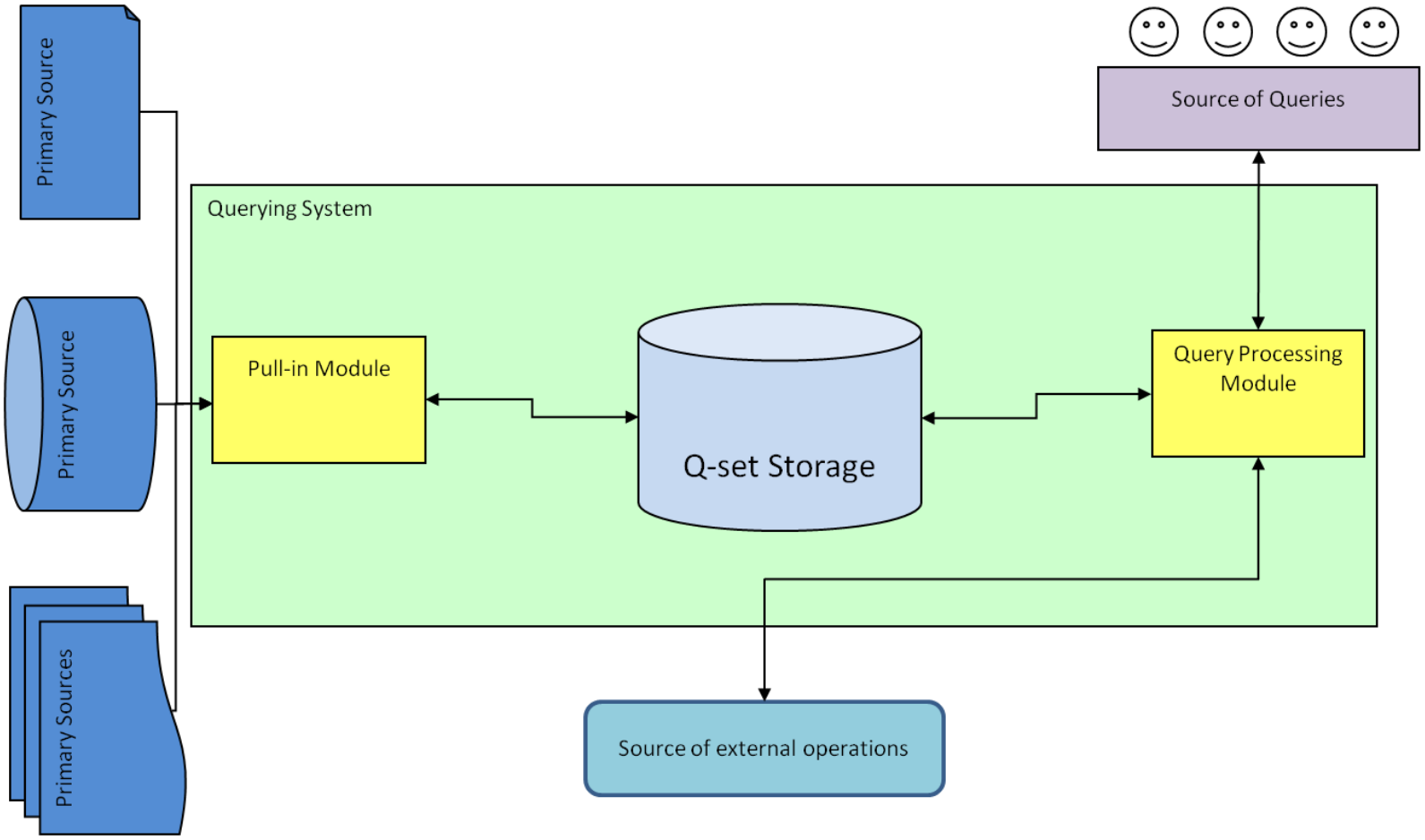
Implementation

Optimization

Adaptive Processing

The Prototype

The Query Processing Engine Prototype



Building the Prototype

- Implement as a middleware
- Core functionality
  - Algebraic operations
  - Adaptive optimization
  - Flexible cost models
- Rich and extendible library
  - Predicates, grouping, aggregate functions
  - Enrichment filters
  - Algebraic expressions (views, design patterns)
- Alternative platforms for operations
  - Traditional (parallel) DBMS
  - Map-reduce (Hadoop)
  - ASTERIX
  - StratoSphere

Open Issues

Getting It

Actually, not in the scope of this talk, but

- J. Gray: *We can store more data than we can process* (even worse than when it was written)
- Delivery of raw data is hardly possible
  - Need in pre-processing and aggregation immediately when data are produced

Processing It

- How to process?
  - Low-level object-oriented programming might be dangerous
  - High level specifications are essential
- Modeling and controlling the quality yield by approximate algorithms
- Approximate evaluation might be inappropriate when exceptions are needed

Making Sense of It

Mostly not in the scope of this talk, but related

- Controlling the quality
  - Automatic calibration
  - Operations improving the quality
- Verification
- Privacy and security is always an issue

Finally…

Acknowledgements

The work presented here is supported by

- HP Labs
- Russian Foundation for Basic Research
- Saint Petersburg University

Co-Authors and the Team

- The talk is based on research co-authored by and prepared with invaluable help of
  - Natalia Vassilieva
  - Anna Yarygina
- The information management research group at Saint Petersburg University http://meta.math.spbu.ru/en/papers.shtml

Conclusions

- High level declarative tools can be effective and efficient
- Query processing in the context of big data is feasible and promising for
  - Advanced analysis
  - Specification of data processing workflows
- Making the outcome of analysis reliable is a big challenge