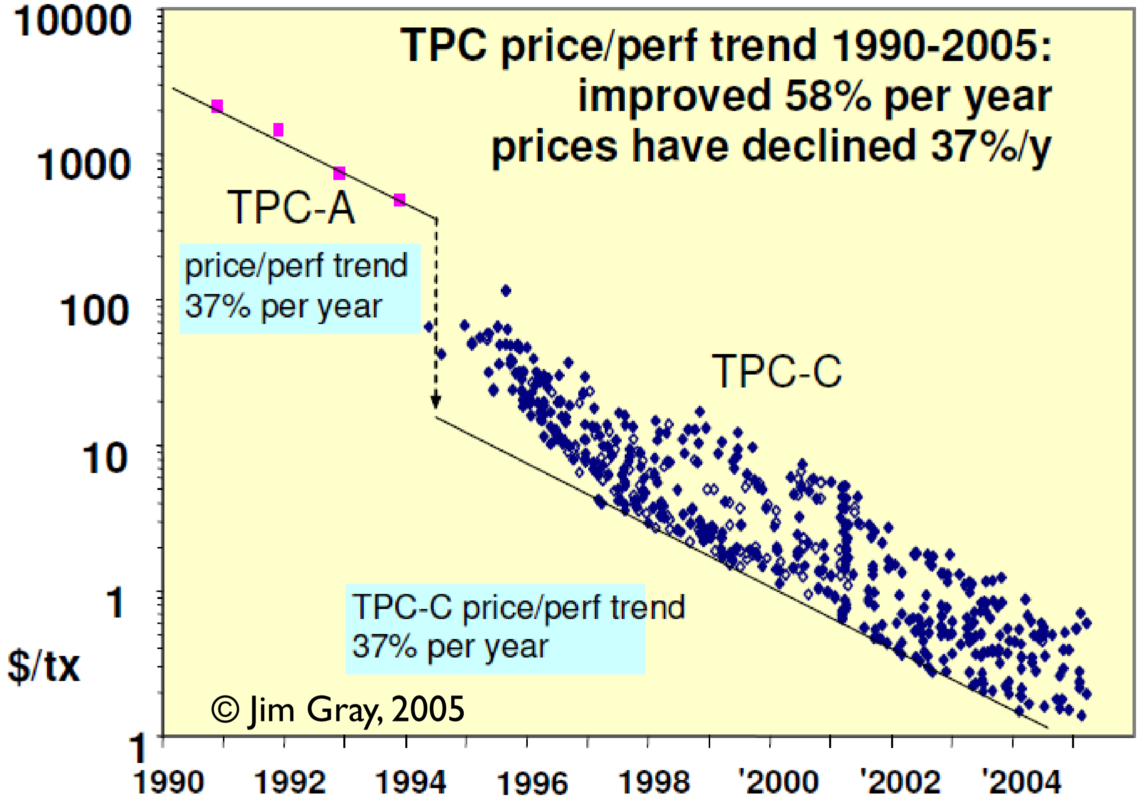
Benchmarking Graph Data Management Systems

Peter Boncz

Why Benchmarking?



- make competing products comparable
- accelerate progress, make technology viable

What is the LDBC?

Linked Data Benchmark Council = LDBC

- Industry entity similar to TPC (www.tpc.org)
- Focusing on graph and RDF store benchmarking

Kick-started by an EU project

- Runs from September 2012 – March 2015
- 9 project partners:

- Will continue independently after the EU project

LDBC Benchmark Design

Developed by so-called "task forces"

- Requirements analysis and use case selection.
  - Technical User Community (TUC)
- Benchmark specification.
  - data generator
  - query workload
  - metrics
  - reporting format
- Benchmark implementation.
  - tools (query drivers, data generation, validation)
  - test evaluations
- Auditing
  - auditing guide
  - auditor training

LDBC: what systems?

Benchmarks for:

- RDF stores (SPARQL speaking)
  - Virtuoso, OWLIM, BigData, Allegrograph,…
- Graph Database systems
  - Neo4j, DEX, InfiniteGraph, …
- Graph Programming Frameworks
  - Giraph, Green Marl, Grappa, GraphLab,…
- Relational Database systems

LDBC: functionality

Benchmarks for:

- Transactional updates in (RDF) graphs
- Business Intelligence queries over graphs
- Graph Analytics (e.g. graph clustering)
- Complex RDF workload, e.g. including reasoning, or for data integration

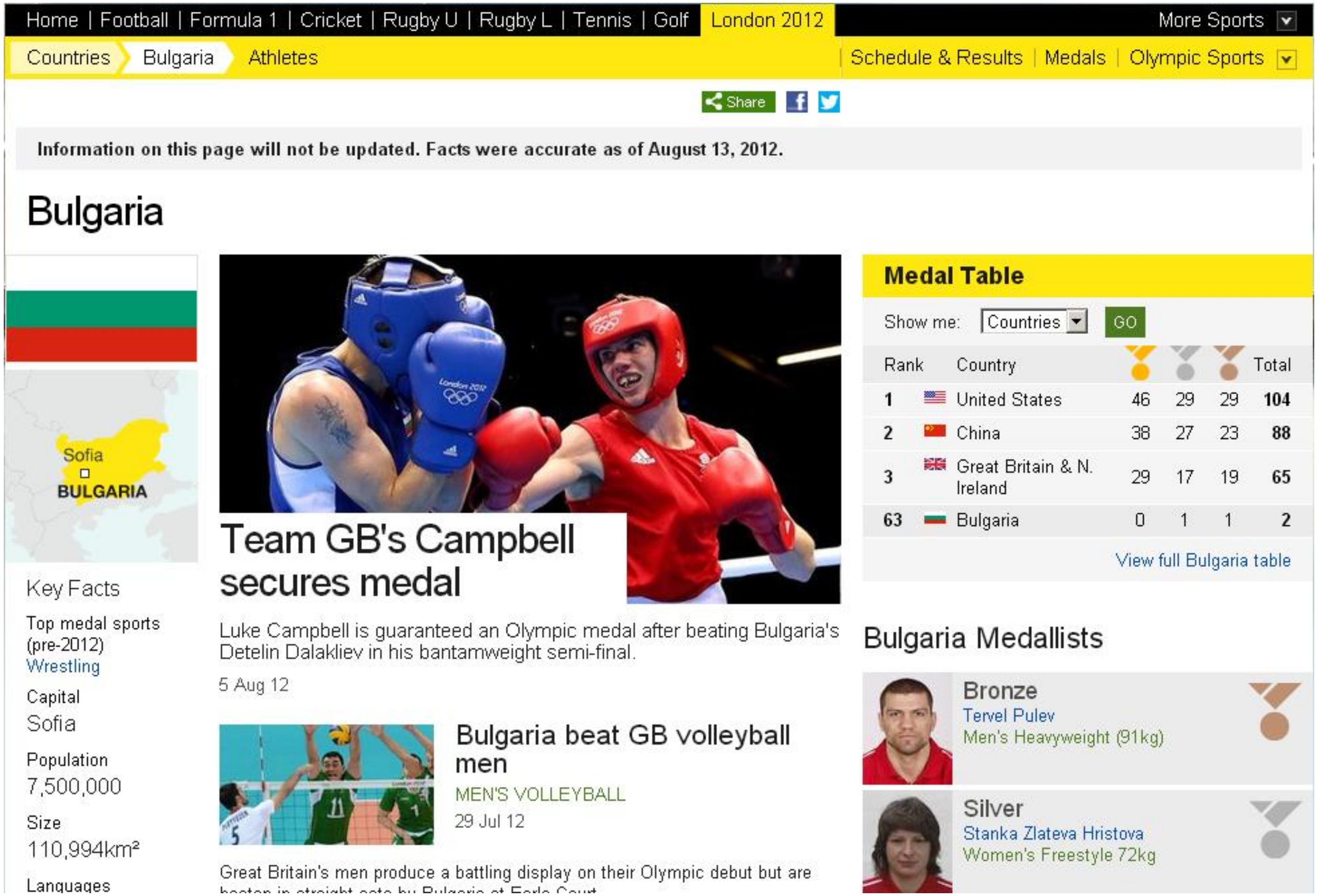
Anything relevant for RDF and graph data management systems

LDBC:organization

- Board of Directors
  - Formed by LDBC member organizations
- Task Forces

  Takes care of a Benchmark or set of benchmarks from beginning to end
    - **Semantic Publishing Benchmark** (SPB)
    - **Social Network Benchmark** (SNB)
- Technical User Community (TUC)
  - Regular meetings with professional users
- End User Community
  - Initiates activities spring 2014
  - Draft Benchmark launches SPB & SNB

SPB scenario: Semantic Publishing



SPB scope

- The scenario involves a media/ publisher organization that maintains semantic metadata about its Journalistic assets (articles, photos, videos, papers, books, etc), also called Creative Works
- The Semantic Publishing Benchmark simulates:
  - Consumption of RDF metadata (Creative Works)
  - Updates of RDF metadata, related to Annotations
- Aims to be an industrially mature RDF database benchmark (SPARQL1.1, some reasoning, text and GIS queries, backup&restore)

SNB Scenario: Social Network Analysis

- Intuitive: everybody knows what a SN is
  - Facebook, Twitter, LinkedIn, …
- SNs can be easily represented as a graph
  - Entities are the nodes (Person, Group, Tag, Post, ...)
  - Relationships are the edges (Friend, Likes, Follows, …)
- Different scales: from small to very large SNs
  - Up to billions of nodes and edges
- Multiple query needs:
  - interactive, analytical, transactional
- Multiple types of uses:
  - marketing, recommendation, social interactions, fraud detection, ...

Audience

- For **developers** facing graph processing tasks
  - recognizable scenario to compare merits of different products and technologies
- For **vendors** of graph database technology
  - checklist of features and performance characteristics
- For **researchers**, both industrial and academic
  - challenges in multiple choke-point areas such as graph query optimization and (distributed) graph analysis

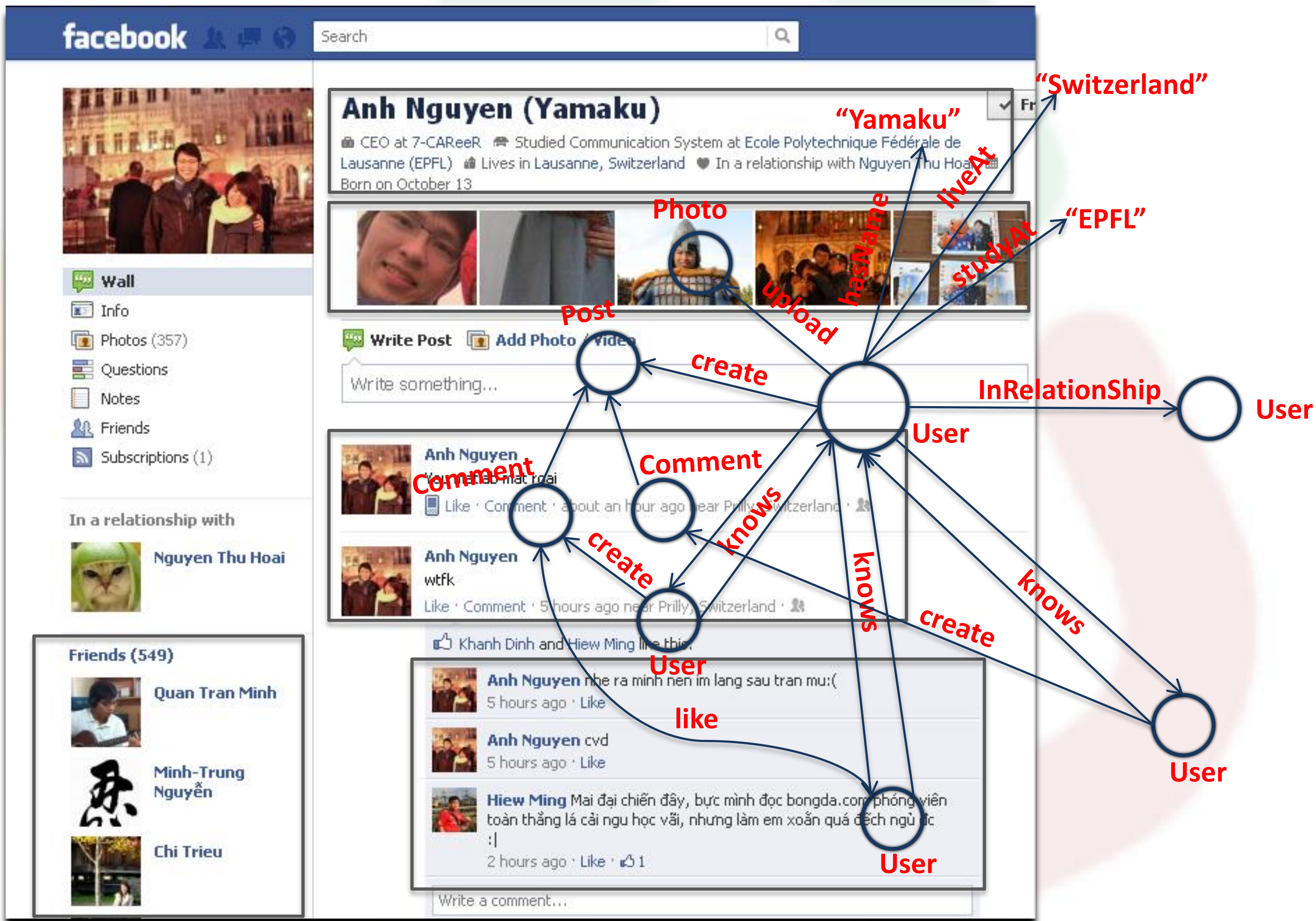
What was developed?

- Four main elements:
  - *data schema*: defines the structure of the data
  - *workloads*: defines the set of operations to perform
  - *performance metrics*: used to measure (quantitatively) the performance of the systems
  - *execution rules*: defined to assure that the results from different executions of the benchmark are valid and comparable
- Software as Open Source (GitHub)
  - data generator, query drivers, validation tools, ...

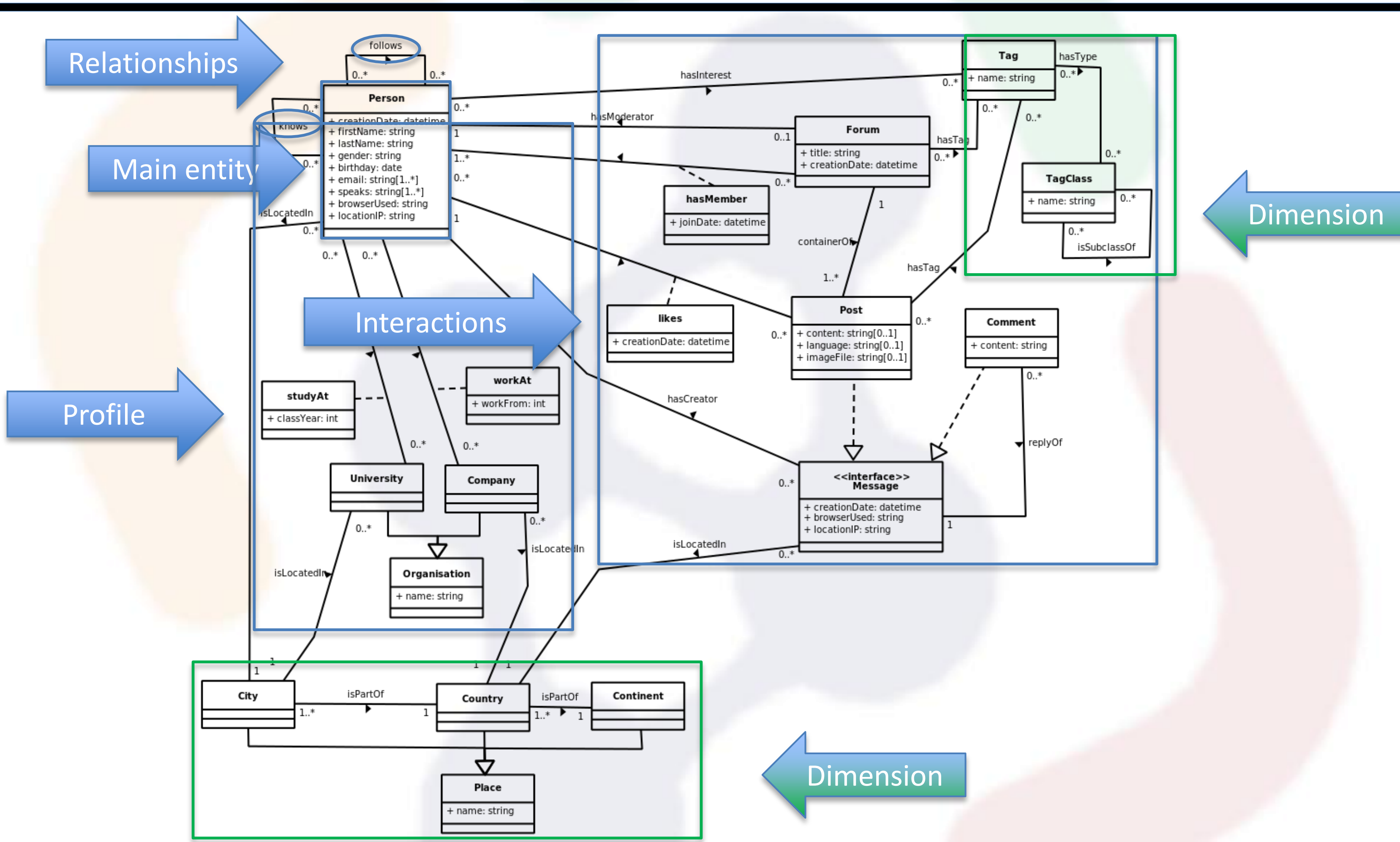
SNB: Data Generator

- Specified in UML for portability
  - Classes
  - associations between classes
  - Attributes for classes and associations
- Some of the relationships represent dimensions
  - Time (Y,QT,Month,Day)
  - Geography (Continent,Country,Place)
- Data Formats
  - CSV
  - RDF (Turtle + N3)

LDBC Social Network Benchmark (SNB)

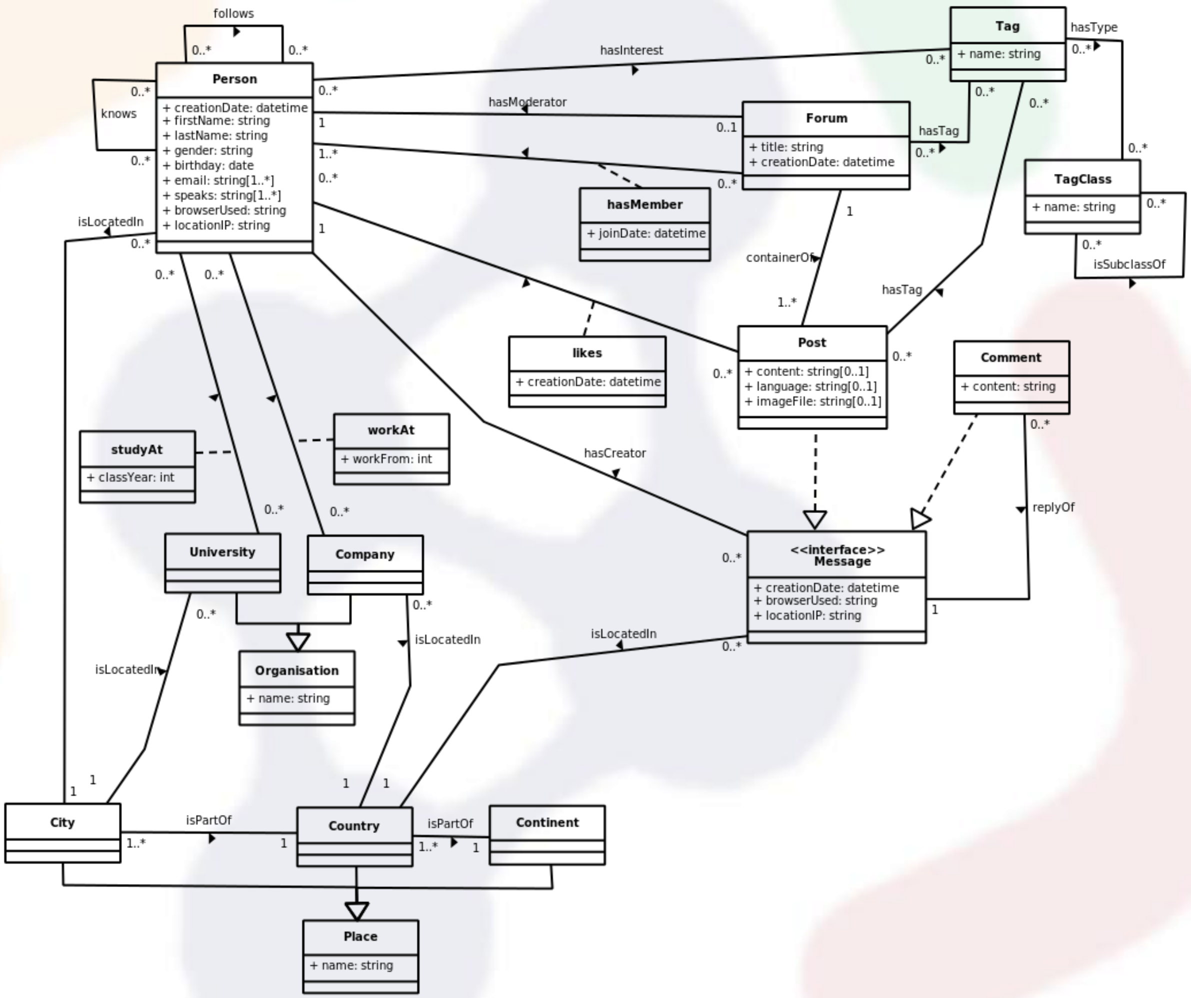


Data Schema



LDBC
Linked Data Benchmark Council

Data Schema



LDBC Linked Data Benchmark Council

Workloads

- **On-Line**: tests a system's throughput with relatively simple queries with concurrent updates
  – *Show all photos posted by my friends that I was tagged in*
- **Business Intelligence**: consists of complex structured queries for analyzing online behavior
  – *Influential people the topic of open source development?*
- **Graph Analytics**: tests the functionality and scalability on most of the data as a single operation
  – *PageRank, Shortest Path(s), Community Detection*

Workloads by system

System	Interactive	Business Intelligence	Graph Analytics
Graph databases	Yes	Yes	Maybe
Graph programming frameworks	-	Yes	Yes
RDF databases	Yes	Yes	-
Relational databases	Yes	Yes	Maybe, by keeping state in temporary tables, and using the functional features of PL-SQL
NoSQL Key-value	Maybe	Maybe	-
NoSQL MapReduce	-	Maybe	Yes

Roadmap for the Keynote

Choke-point based benchmark design

- What are Choke-points?
  - examples from good-old TPC-H
  - → relational database benchmarking
- A Graph benchmark Choke-Point, in-depth:
  - **Structural Correlation in Graphs**
  - and what we do about it in LDBC
- Wrap up

Keynote Roadmap

- LDBC and its benchmarks
- Benchmark Design ➔ "choke points"
- Correlated Graph Generation
- SNB Details & Status
- Conclusion

Database Benchmark Design

Desirable properties:

- Relevant.
- Representative.
- Understandable.
- Economical.
- Accepted.
- Scalable.
- Portable.
- Fair.
- Evolvable.
- Public.

Jim Gray (1991) *The Benchmark Handbook for Database and Transaction Processing Systems*

Dina Bitton, David J. DeWitt, Carolyn Turbyfill (1993) *Benchmarking Database Systems: A Systematic Approach*

Multiple TPCTC papers, e.g.:
Karl Huppler (2009) *The Art of Building a Good Benchmark*

Stimulating Technical Progress

- An aspect of 'Relevant'
- The benchmark metric
  - depends on,
  - or, rewards:

solving certain
technical challenges

(not commonly solved by technology at benchmark design time)

Benchmark Design with Choke Points

Choke-Point = well-chosen difficulty in the workload

- "difficulties in the workloads"
  - arise from Data (distribs)+Query+Workload
  - there may be different technical solutions to address the choke point
    - or, there may not yet exist optimizations (but should not be NP hard to do so)
    - the impact of the choke point may differ among systems

Benchmark Design with Choke Points

Choke-Point = well-chosen difficulty in the workload

- “difficulties in the workloads”
- “well-chosen”
  - the majority of actual systems do not handle the choke point very well
  - the choke point occurs or is likely to occur in actual or near-future workloads

Example: TPC-H choke points

- Even though it was designed without specific choke point analysis
- TPC-H contained a lot of interesting challenges
  - many more than Star Schema Benchmark
  - considerably more than XMark (XML DB benchmark)
  - not sure about TPC-DS (yet)

TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

TPC-H choke point areas (1/3)

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22

TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

TPC-H choke point areas (2/3)

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22

TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

TPC-H choke point areas (3/3)

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	

TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

CP1.4 Dependent GroupBy Keys

Q10

```
SELECT c_custkey,  c_name, c_acctbal,
   sum(l_extendedprice * (1 - l_discount)) as revenue,
   n_name,  c_address,  c_phone, c_comment
FROM  customer, orders,  lineitem,  nation
WHERE  c_custkey = o_custkey and l_orderkey = o_orderkey
   and o_orderdate >= date '[DATE]'
   and o_orderdate < date '[DATE]' + interval '3' month
   and l_returnflag = 'R` and c_nationkey = n_nationkey
GROUP BY
   c_custkey, c_name,   c_acctbal,  c_phone,  n_name,
   c_address, c_comment
ORDER BY revenue DESC
```

TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

CP1.4 Dependent GroupBy Keys

Q10

```
SELECT c_custkey,  c_name, c_acctbal,
   sum(l_extendedprice * (1 - l_discount)) as revenue,
   n_name,  c_address,  c_phone, c_comment
FROM  customer, orders,  lineitem,  nation
WHERE  c_custkey = o_custkey and l_orderkey = o_orderkey
   and o_orderdate >= date '[DATE]'
   and o_orderdate < date '[DATE]' + interval '3' month
   and l_returnflag = 'R` and c_nationkey = n_nationkey
GROUP BY
   c_custkey, c_name,   c_acctbal,  c_phone,
   c_address, c_comment, n_name
ORDER BY revenue DESC
```

TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

Exasol:
"foreign key check" phase after load

CP1.4 Dependent GroupBy Keys

- Functional dependencies:

  `c_custkey` ➔ `c_name`, `c_acctbal`, `c_phone`, `c_address`, `c_comment`, `c_nationkey` ➔ `n_name`

- Group-by hash table should exclude the colored attrs ➔ less CPU+ mem footprint
- in TPC-H, one can choose to declare primary and foreign keys (all or nothing)
  - this optimization requires declared keys
  - Key checking slows down RF (insert/delete)

TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

CP2.2 Sparse Joins

- Foreign key (N:1) joins towards a relation with a selection condition
  - Most tuples will *not* find a match
  - Probing (index, hash) is the most expensive activity in TPC-H
- Can we do better?
  - Bloom filters!

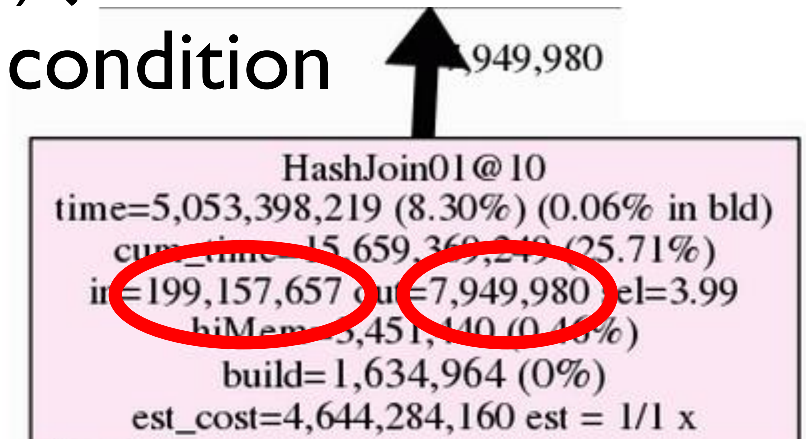
TPCTC 2013: www.cwi.nl/~boncz/tpctc2013_boncz_neumann_erling.pdf
"TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark"

CP2.2 Sparse Joins

- Foreign key (N:1) joins towards a relation with a selection condition

Q21

probed: 200M tuples
result: 8M tuples
➔ 1:25 join hit ratio



Vectorwise:
TPC-H joins typically accelerate 4x
Queries accelerate 2x

2G cycles 29M probes ➔ cost would have been 14G cycles ~= 7 sec

```
#PROF 2021162220    OWN 28950172   9.8avg rdtsc  307565 calls vht_lookup_keys() "vht_lookup_keys" in con
#PROF 1575739535    OWN 199097581   7.9avg rdtsc  307534 calls sel_bitfiltercheck_uchr_col_slng_val_sint_
```

1.5G cycles 200M probes ➔ 85% eliminated

CP5.2 Subquery Rewrite

Q17

```
SELECT sum(l_extendedprice) / 7.0 as avg_yearly
FROM lineitem,  part
WHERE p_partkey = l_partkey
   and p_brand = '[BRAND]'
   and p_container = '[CONTAINER]'
   and l_quantity <( SELECT 0.2 * avg(l_quantity)
                     FROM lineitem
                     WHERE l_partkey = p_partkey)
```

This subquery can be extended with restrictions from the outer query.

Hyper:
CP5.1+CP5.2+CP5.3
results in 500x faster
Q17

```
SELECT 0.2 * avg(l_quantity)
FROM lineitem
WHERE l_partkey = p_partkey
   and p_brand = '[BRAND]'
   and p_container = '[CONTAINER]'
```

+ CP5.3 Overlap between Outer- and Subquery.

Keynote Roadmap

- LDBC and its benchmarks
- Benchmark Design → "choke points"
- Correlated Graph Generation
- SNB Details & Status
- Conclusion

Data correlations between attributes

```
SELECT personID from person
WHERE firstName = 'Joachim' AND addressCountry = 'Germany'
```

Anti-Correlation

```
SELECT personID from person
WHERE firstName = 'Cesare'  AND addressCountry = 'Italy'
```

- Query optimizers may underestimate or overestimate the result size of conjunctive predicates

Joachim Cesare Loew   Joachim Cesare Prandelli

Data correlations **between attributes**

```
SELECT COUNT(*)
FROM paper pa1 JOIN conferences cn1 ON pa1.journal = jn1.ID
     paper pa2 JOIN conferences cn2 ON pa2.journal = jn2.ID
WHERE  pa1.author = pa2.author    AND
              cn1.name = 'VLDB'   AND  cn2.name = 'SIGMOD'
```

Data correlations **over joins**

```
SELECT COUNT(*)
FROM paper pa1 JOIN conferences cn1 ON pa1.journal = cn1.ID
     paper pa2 JOIN conferences cn2 ON pa2.journal = cn2.ID
WHERE  pa1.author = pa2.author    AND
              cn1.name = 'VLDB'   AND  cn2.name = 'SIGMOD' 'Nature'
```

- A challenge to the optimizers to adjust estimated join hit ratio

  `pa1.author = pa2.author`

  depending on other predicates

Correlated predicates are still a frontier area in database research

Handling Correlation: a choke point for Graph DBs

- What makes graphs interesting are the connectivity patterns
  - who is connected to who?
  - ➔ structure typically depends on the (values) attributes of nodes
- **Structural Correlation (➔ choke point)**
  - amount of common friends
  - shortest path between two persons search complexity in a social network varies wildly between two random persons
  - e.g. colleagues at the same company
- No existing graph benchmark specifically tests for the effects of correlations
- Synthetic graphs used for benchmarking do not have structural correlations

Need a data generator generating synthetic graph
with data/structure correlations

Generating **Correlated** Property Values

- How do data generators generate values? E.g. **`FirstName`**

Generating Property Values

- How do data generators generate values? E.g. `FirstName`
- **Value** Dictionary **D()**
  - a fixed set of values, e.g.,

    {"Andrea","Anna","Cesare","Camilla","Duc","Joachim", .. }
- **Probability** density function **F()**
  - steers how the generator chooses values
    - cumulative distribution over dictionary entries determines which value to pick
  - could be anything: uniform, binomial, geometric, etc…
    - geometric (discrete exponential) seems to explain many natural phenomena

Generating **Correlated** Property Values

- How do data generators generate values? E.g. `FirstName`
- **Value** Dictionary **D()**
- **Probability** density function **F()**
- **Ranking** Function **R()**
  - Gives each value a unique rank between one and **|D|**
    - determines which value gets which probability
  - Depends on some parameters (parameterized function)
    - value frequency distribution becomes correlated by the parameters or **R()**

Generating **Correlated** Property Values

- How do data generators generate values? E.g. `FirstName`
- **Value** Dictionar
  {"Andrea"," ... **'O**
- **Probability** d
  geometric d
- **Ranking** Function **R(gender,country,birthyear)**
  - **gender, country, birthyear → correlation parameters**

How to implement R()?

We need a table storing

limited #combinations

|Gender| X |Country| X |BirthYear| X |D|

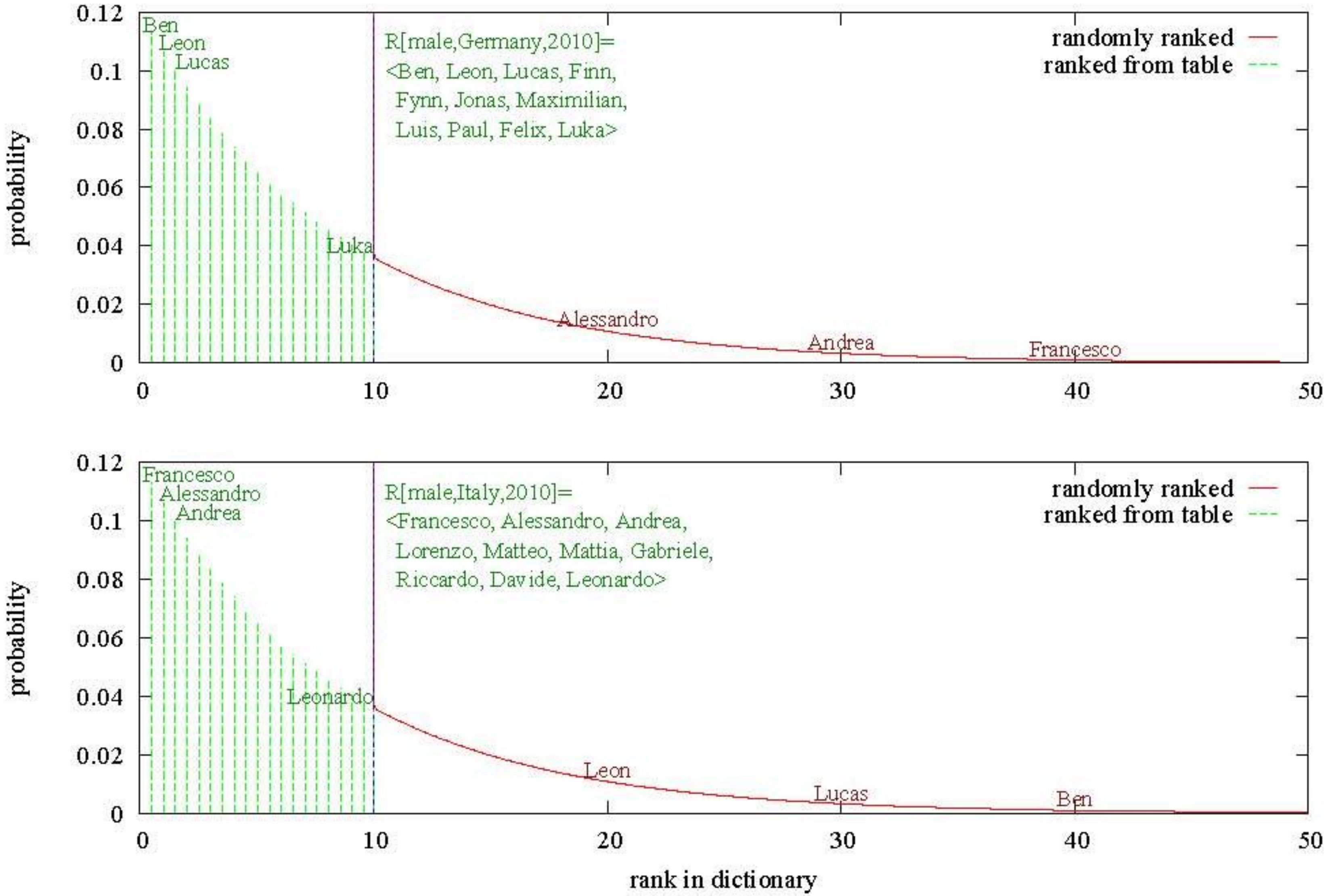
Potentially Many! ☹

Solution:

- Just store the rank of the **top-N** values, not all **|D|**
- Assign the rank of the other dictionary values randomly

Compact Correlated Property Value Generation

Using geometric distribution for function **F()**



Correlated Value Property in LDBC SNB

- Main source of dictionary values from DBpedia (http://dbpedia.org)

- Various realistic property value correlations (→)

  e.g.,

  (person.location,person.gender,person.birthDay) → person.firstName

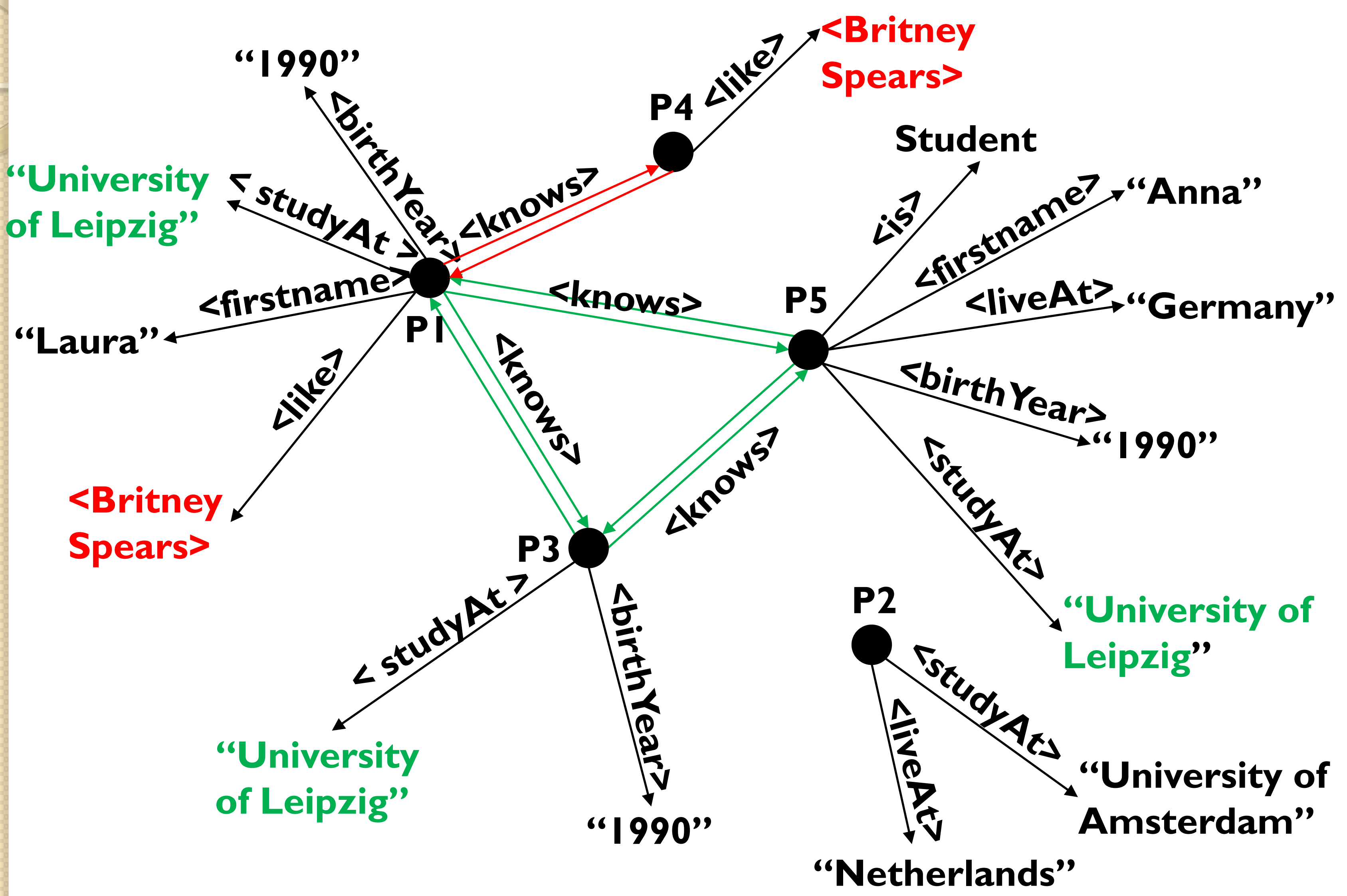
  person.location → person.lastName

  person.location → person.university

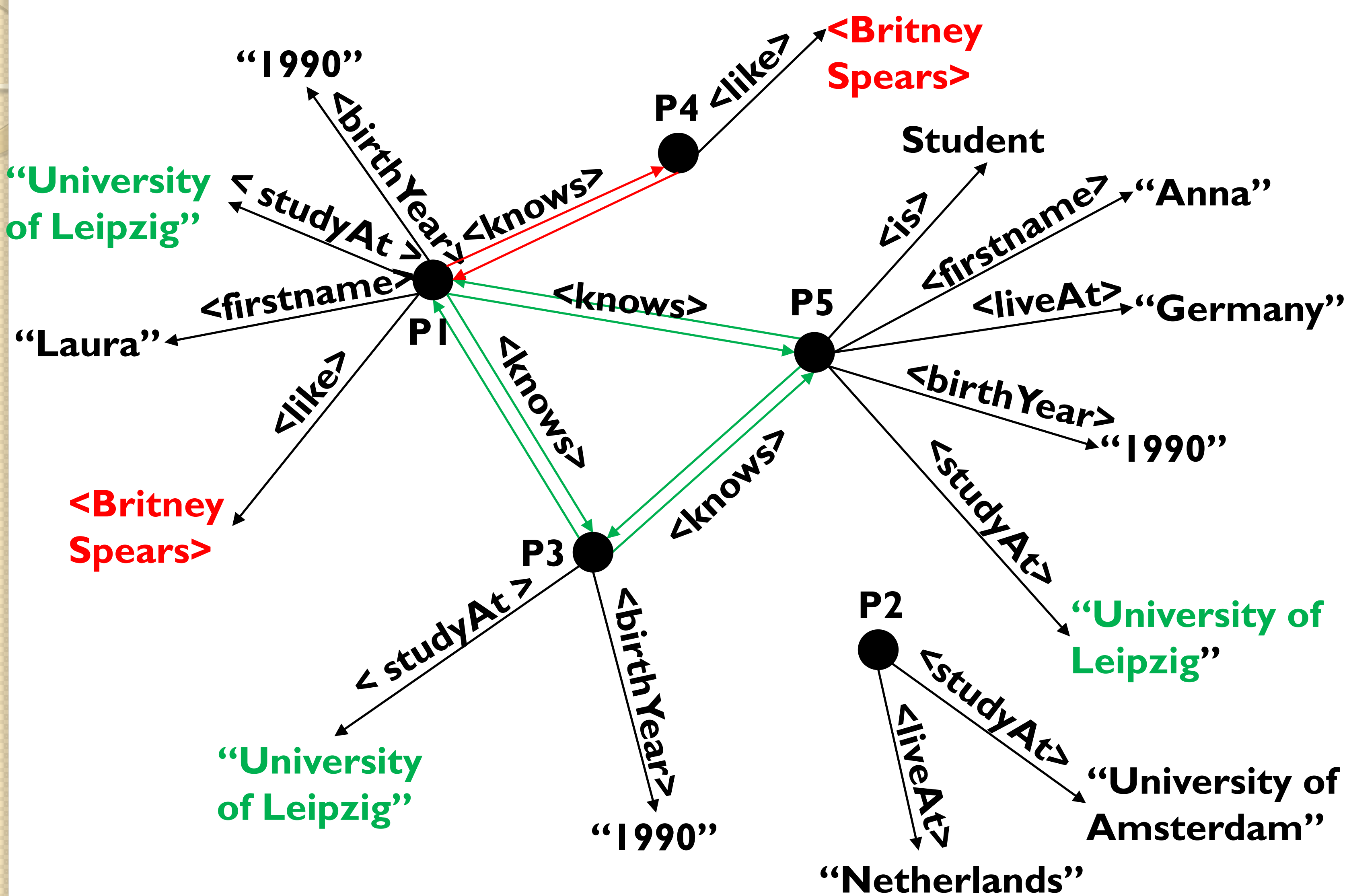
  person.createdDate → person.photoAlbum.createdDate

  ….

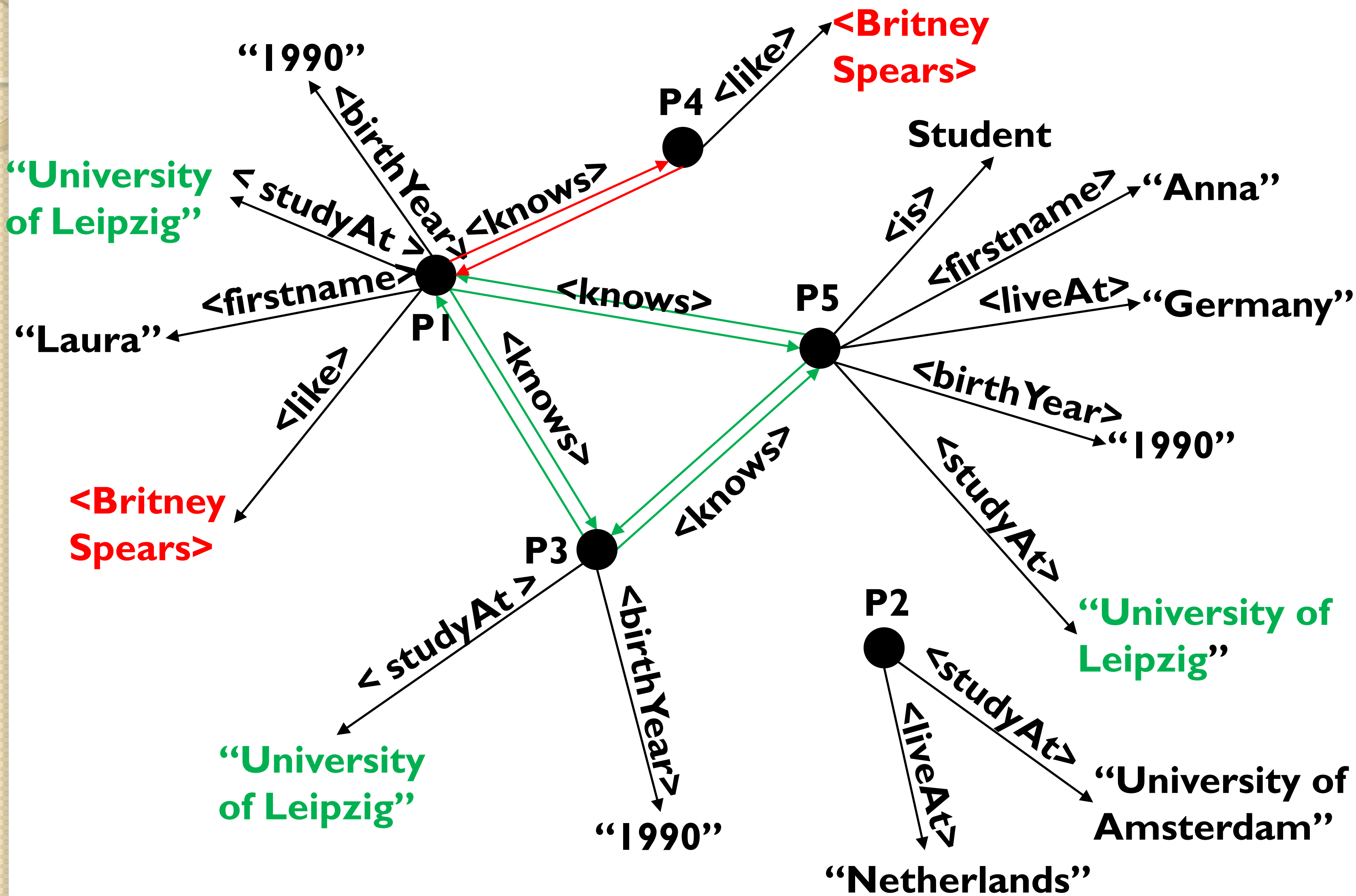
Correlated Edge Generation



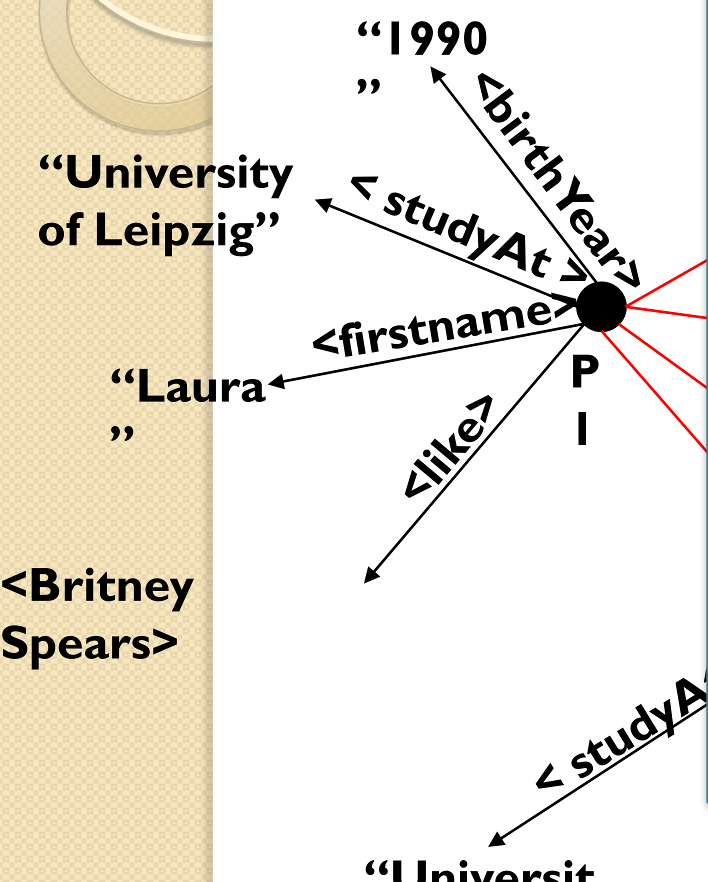
Correlated Edge Generation



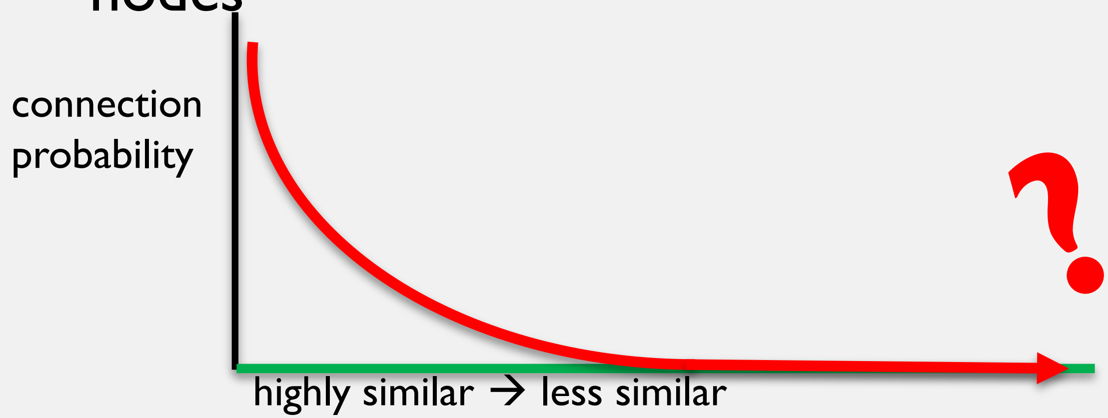
Correlated Edge Generation



Simple approach

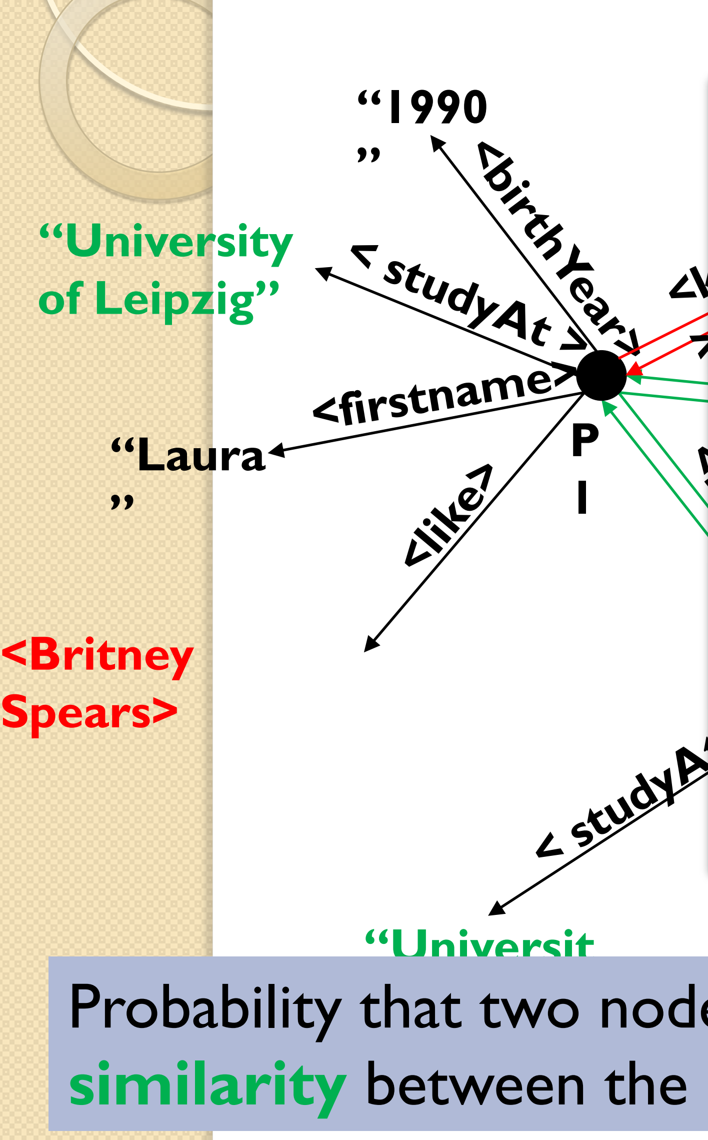


- Compute **similarity** of two nodes based on their (correlated) **properties**.
- Use a **probability density function** wrt to this similarity for connecting nodes

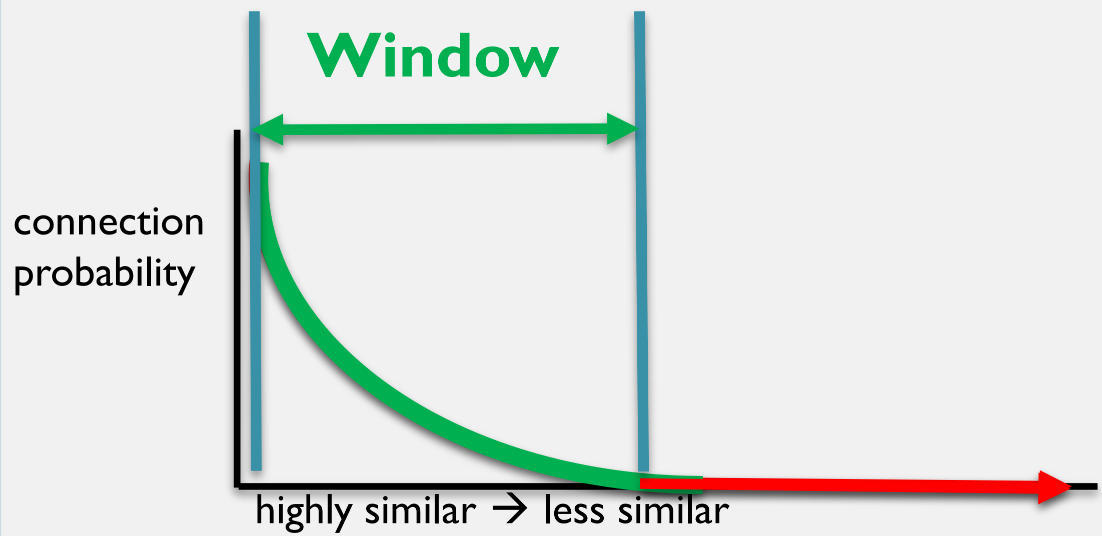


Danger: this is very expensive to compute on a large graph!
(quadratic, random access)

Our observation



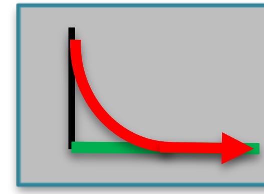
Trick: disregard nodes with too large similarity distance (only connect nodes in a **similarity window**)



Probability that two nodes are connected is **skewed** *w.r.t* the **similarity** between the nodes (due to probability distr.)

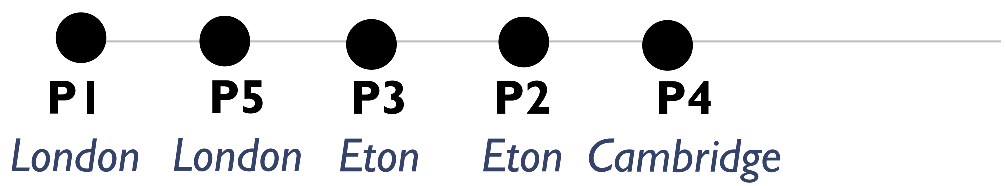
Correlation Dimensions

Similarity metric + Probability function



- **Similar metric**

  Sort nodes on similarity (similar nodes are brought near each other)



<Ranking along the “*Having study together*” dimension>

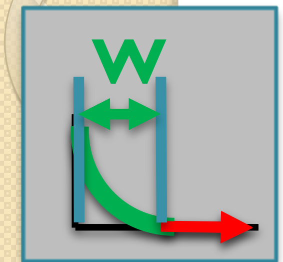
we use ***space filling curves*** *(e.g. Z-order) to get a linear dimension*

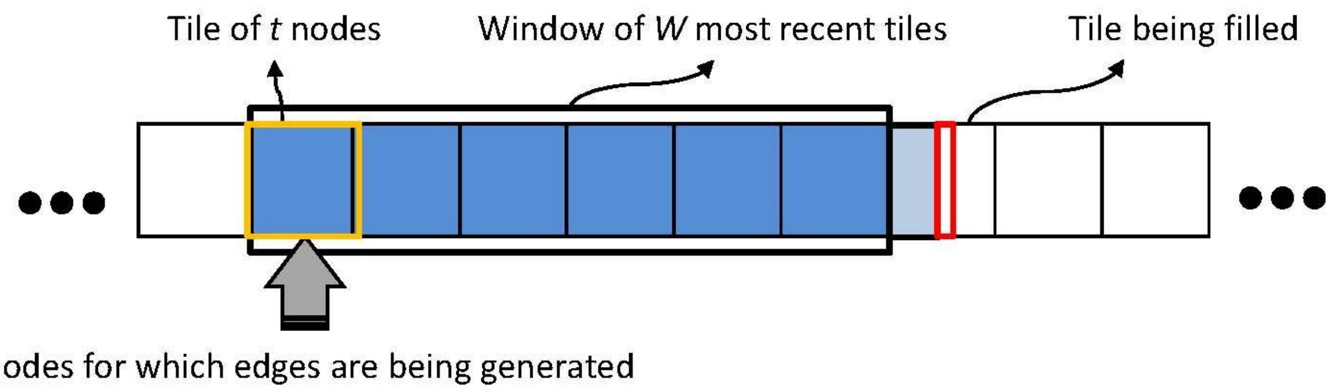
- **Probability function**

  Pick edge between two nodes based on their **ranked distance**

  (e.g. geometric distribution, again)

Generate edges along correlation dimensions





- Sort nodes using **MapReduce** on similarity metric
- Reduce function keeps a **window** of nodes to generate edges
  - Keep low memory usage (sliding window approach)

- Slide the window for **multiple passes**, each pass corresponds to one correlatio dimension (multiple MapReduce jobs)
  - for each node we choose **degree** per pass (also using a prob. function)

    steers how many edges are picked in the window for that node

Correlation Dimensions in LDBC SNB

- Having studied together
- Having common interests (hobbies)
- Random dimension
  - motivation: not all friendships are explainable (…)

(of course, these two correlation dimensions are still a gross simplification of reali
but this provides some interesting material for benchmark queries)

TPCTC 2012: www.cwi.nl/~boncz/tpctc2012_pham_boncz_erling.pdf
"S3G2: A Scalable Structure-correlated Social Graph Generator"

SNB Data Generator results

- **Social graph characteristics**
  - Output graph has similar characteristics as observed in real social network (i.e., *"small-world network"* characteristics)

    - Power-law social degree distribution

    - Low average path-length

    - High clustering coefficient

- **Scalability**
  - Generates up to **1.2 TB** of data (1.2 million users) in **half an hour**

    - Runs on a cluster of 16 nodes

    (part of the SciLens cluster, www.scilens.org)

  - **Scales** out **linearly**

TPCTC 2012: www.cwi.nl/~boncz/tpctc2012_pham_boncz_erling.pdf
"S3G2: A Scalable Structure-correlated Social Graph Generator"

Summary

- correlation between values (“properties”) and connection pattern in graphs affects many real-world data management tasks

  ➔use as a choke point in the Social Network Benchmark

- generating huge correlated graphs is hard!

  ➔MapReduce algorithm that approximates correlation probabilities with windowed-approach

  See: for more info

  - https://github.com/ldbc
  - SNB task-force wiki http://www.ldbc.eu:8090/display/TUC

Keynote Roadmap

- LDBC and its benchmarks
- Benchmark Design → "choke points"
- Correlated Graph Generation
- SNB Details & Status
- Conclusion

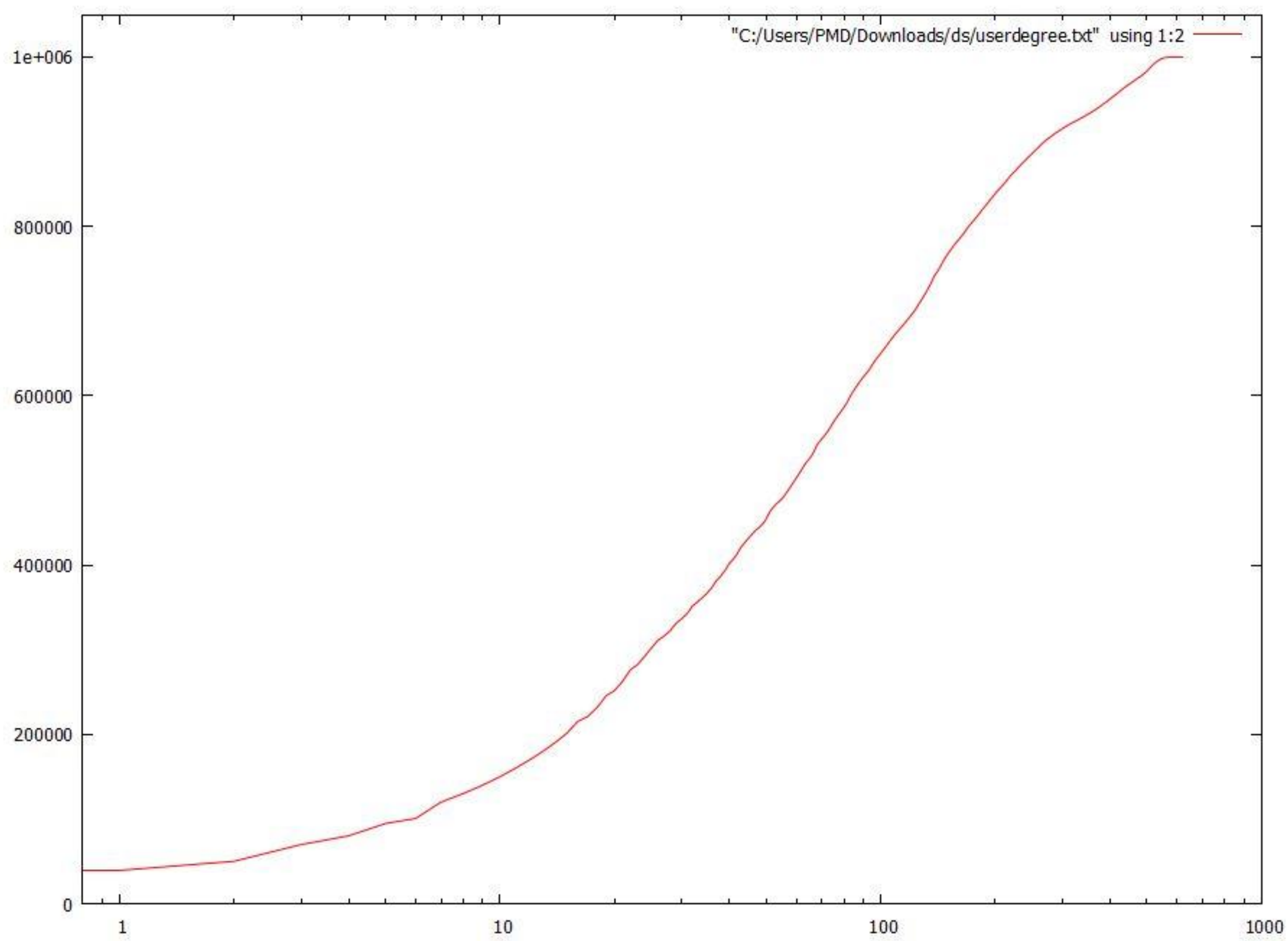
Validation: Metrics

- Largest Connected Component
- Average Clustering Coefficient
- Diameter
- Average Path Length
- Hop-plot User-Knows
- Attribute distributions
- Degree distributions
- Time evolution

Statistics (100K users / 1 year)

Group	Statistic	Value
Settings	Number of users (Person instances)	100,000
	Number of years	1
Elements	Nodes	80,767,146
	Edges	350,352,746
	Attribute Values	500,108,979
	RDF triples	942,563,664
Metrics	Largest connected component (community)	99.78%
	Average path length (small world)	3.93
	Average clustering coefficient (transitivity)	0.11
	Largest distance between two nodes (diameter)	11
Knows relationship	Edges	2,887,796
	Diameter	6

Friends Distribution @ 1M persons



Interactive Query Set

- Tests system throughput with relatively simple queries and concurrent updates
- Current set: 12 read-only queries
- For each query:
  - Name and detailed description in plain English
  - List of input parameters
  - Expected result: content and format
  - Textual functional description
  - Relevance:
    - textual description (plain English) of the reasoning for including this query in the workload
    - discussion about the technical challenges (Choke Points) targeted
  - Validation parameters and validation results
  - SPARQL query

Some SNB Interactive Choke Points

- **Graph Traversals**. Query execution time heavily depends on the ability to quickly traverse friends graph.
- **Plan Variablility**. Each query have many different best plans depending on parameter choices (eg. Hash- vs index-based joins).
- **Top k and distinct**: Many queries return the first results in a specific order: Late projection, pushing conditions from the sort into the query
- **Repetitive short queries**, differing only in literals, opportunity for query plan recycling

Choke Point Coverage

Group	Choke Point	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Aggregation Performance	1.2		+							+			
	1.6									+			
	1.7	+											
Join Performance	2.3	+											
	2.4		+					+					
	2.6					+		+	+		+		
	2.7		+		+	+		+		+	+	+	
Data Access Locality	3.3			+									
	3.5		+						+	+			
Expression Calculation	4.2a									+			
Correlated Subqueries	5.1									+			
	5.3									+			
Parallelism and Concurrency	6.3									+			
RDF and Graph Specifics	7.1	+								+			
	7.2						+						+
	7.3												+

Example: Q3

Name: Friends within 2 hops that have been in two countries

Description:

Find Friends and Friends of Friends of the user A that have made a post in the foreign countries X and Y within a specified period. We count only posts that are made in the country that is different from the country of a friend. The result should be sorted descending by total number of posts, and then by person URI. Top 20 should be shown. The user A (as friend of his friend) should not be in the result

Parameter:

- Person
- CountryX
- CountryY
- startDate - the beginning of the requested period
- Duration - requested period in days

Result:

- Person.id, Person.firstname, Person.lastName
- Number of post of each country and the sum of all posts

Relevance:

- Choke Points: CP3.3
- If one country is large but anticorrelated with the country of self then processing this before a smaller but positively correlated country can be beneficial

LDBC Linked Data Benchmark Council

Example: Q5 - SPARQL

```
select ?group count (*)
where {
   {select distinct ?fr
    where {
        {%Person% snvoc:knows ?fr.} union
        {%Person% snvoc:knows ?fr2.
         ?fr2 snvoc:knows ?fr. filter (?fr != %Person%)}
    }
   } .
   ?group snvoc:hasMember ?mem . ?mem snvoc:hasPerson ?fr .
   ?mem snvoc:joinDate ?date . filter (?date >= "%Date0%"^^xsd:date) .
   ?post snvoc:hasCreator ?fr . ?group snvoc:containerOf ?post
}
group by ?group
order by desc(2) ?group
limit 20
```

Example: Q5 - Cypher

```
MATCH (person:Person)-[:KNOWS*1..2]-(friend:Person)
WHERE person.id={person_id}
MATCH (friend)<-[membership:HAS_MEMBER]-(forum:Forum)
WHERE membership.joinDate>{join_date}
MATCH (friend)<-[:HAS_CREATOR]-(comment:Comment)
WHERE (comment)-[:REPLY_OF*0..]->(:Comment)-[:REPLY_OF]->(:Post)<-
   [:CONTAINER_OF]-(forum)
RETURN forum.title AS forum, count(comment) AS commentCount
ORDER BY commentCount DESC
MATCH (person:Person)-[:KNOWS*1..2]-(friend:Person)
WHERE person.id={person_id}
MATCH (friend)<-[membership:HAS_MEMBER]-(forum:Forum)
WHERE membership.joinDate>{join_date}
MATCH (friend)<-[:HAS_CREATOR]-(post:Post)<-[:CONTAINER_OF]-(forum)
RETURN forum.title AS forum, count(post) AS postCount
ORDER BY postCount DESC
```

Example: Q5 - DEX

```
v.setLongVoid(personId);
long personOID = graph.findObject(personId, v);
Objects friends = graph.neighbors(personOID, knows, EdgesDirection.Outgoing);
Objects allFriends = graph.neighbors(friends, knows, EdgesDirection.Outgoing);
allFriends.union(friends);
allFriends.remove(personOID);
friends.close();
Objects members = graph.explode(allFriends, hasMember, EdgesDirection.Ingoing);
v.setTimestampVoid(date);
Objects candidate = graph.select(joinDate, Condition.GreaterEqual, v, members);
Objects finalSelection = graph.tails(candidate);
candidate.close();
members.close();
Objects posts = graph.neighbors(allFriends, hasCreator, EdgesDirection.Ingoing);
ObjectsIterator iterator = finalSelection.iterator();
while (iterator.hasNext()) {
      long oid = iterator.next();
      Container c = new Container();
      Objects postsGroup = graph.neighbors(oid, containerOf, EdgesDirection.Outgoing);
      Objects moderators = graph.neighbors(oid, hasModerator, EdgesDirection.Outgoing);
      long moderatorOid = moderators.any();
      moderators.close();
      Objects postsModerator = graph.neighbors(moderatorOid, hasCreator, EdgesDirection.Ingoing);
      postsGroup.difference(postsModerator);
      postsModerator.close();
      postsGroup.intersection(posts);
      long count = postsGroup.size();
      if (count > 0) {
        graph.getAttribute(oid, forumId, v);
        c.row[0] = db.getForumURI(v.getLong());
        c.compare2 = String.valueOf(v.getLong());
        c.row[1] = String.valueOf(count);
        c.compare = count;
        results.add(c);
      }
      postsGroup.close()
}
```

LDBC query driver

- Manages multiple parallel database clients
  - High-throughput testing, cluster-ready
  - Started out as a fork of YCSB
- Interactive Workload
  - Insert queries:
    - Bulk load first years of dataset
    - Play out "last year" of daaset as inserts

    ➔challenge: **respect data dependencies in the graph**

    time window protocol between client processes
  - Read-only Query Set
    - Query set with parameters

    ➔ challenge: **generate relatively stable query behavior**

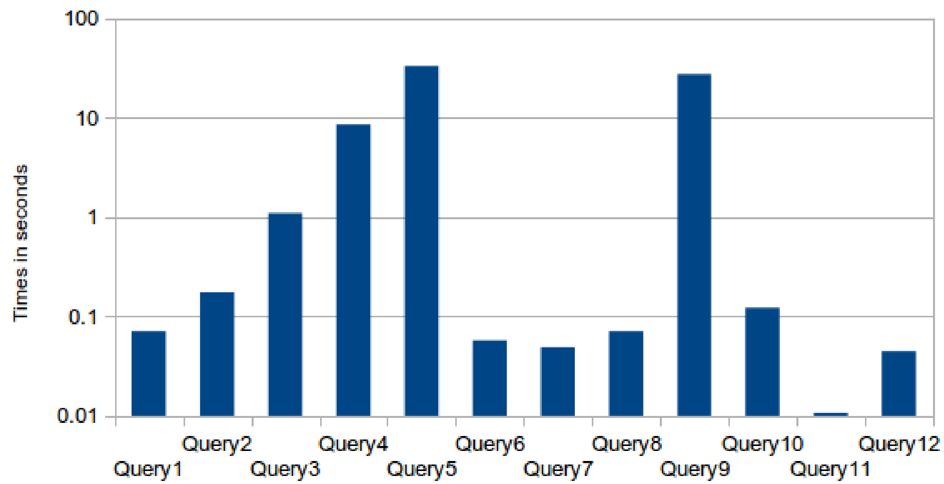
    use data mining on dataset to find "equivalence classes" in parameters

Some Experiments

- Virtuoso (RDF)
  - 100k users during 3 years period (3.3 billion triples, 60GB)
  - Ten SPARQL query mixes
  - 4 x Intel Xeon 2.30GHz CPU, 193 GB of RAM
- DEX (Graph Database)
  - Validation setup: 10k users during 3 years (19GB)
  - Validation query set and parameters (API-based)
  - 2 x Intel Xeon 2.40Ghz CPU, 128 GB of RAM

Virtuoso Interactive Workload

- Some queries could not be considered as truly interactive
  - e.g. Q4, Q5 and Q9
  - … still all queries are very interesting challenges
- ”Irregular” data distribution reflecting the reality of the SN
  - … but complicates the selection of query parameters

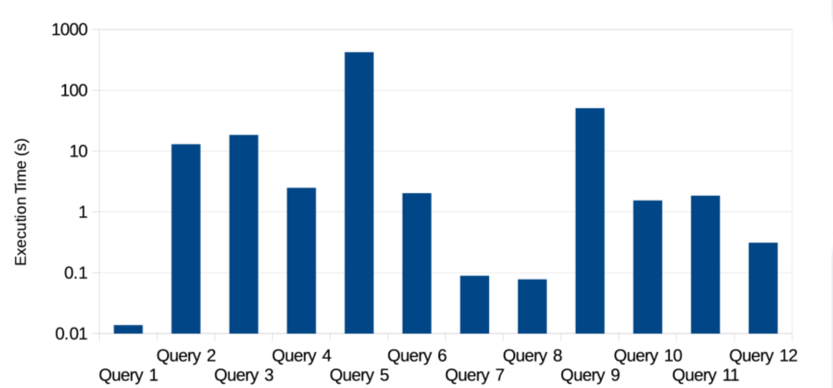


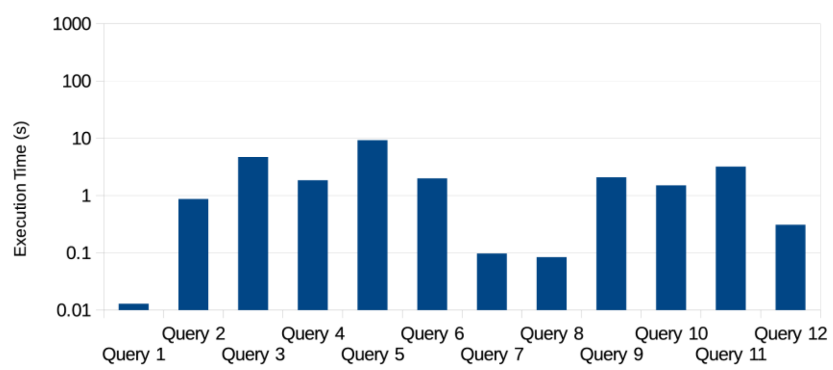
Exploration in Scale

- 3.3 bn RDF triples per 100K users, 24G in triples, 36G in literals
- 2/3 of data in interactive working set, 1/4 in BI working set
- scale out becomes increasingly necessary after 1M
- 10-100M users are data center scales
  - as in real social networks
  - larger scales will favor space efficient data models, e.g. column store with a schema, but
  - larger scales also have greater need for schema-last features

DEX Interactive Workload

- Query validation (no SPARQL)
- Identified some of implementation choke points
- New optimizations implemented and tested





Keynote Roadmap

- LDBC and its benchmarks
- Benchmark Design → "choke points"
- Correlated Graph Generation
- SNB Details & Results
- Conclusion

Status

- First Draft Release of SNB & SPB
  - Data generators
  - Query Drivers
  - Documentation

- Launch of user-facing LDBC website

Expected April/May 2014

Pointers

- **Code&Queries**: github.com/ldbc
  - ldbc_socialnet_bm
    - ldbc_socialnet_dbgen
    - ldbc_socialnet_qgen
- **Wiki**: ldbc.eu:8090/display/TUC
  - Background & Discussions + Detailed report

    "November 213 SNB Task Force Report"
- LDBC Technical User Community (TUC) **meeting**:
  - Thursday April 3, CWI Amsterdam

Conclusion

- LDBC: a new graph/RDF benchmarking initiative
  - EU initatiated, Industry supported
  - benchmarks under development (SNB, SPB)
    - more to follow
- Choke-point based benchmark development
  - SNB: querying and analyzing Correlated graphs

Thank you very much!!
Questions?