chameleon-db

Presented by
Güneş Aluç

Joint work with
M. Tamer Özsu, Khuzaima Daudjee and Olaf Hartig

What is ***chameleon-db***?

- A ***native*** RDF data management system that is **workload-aware**,
  - which means that it ***automatically*** and ***periodically*** adjusts its physical layout to optimize for queries so that they can be executed efficiently;
  - which sets it apart from any of the existing RDF data management systems.

What is ***chameleon-db***?

Q: Why is it necessary/important to have a workload-aware system as such?

- First, we need to
  - characterize ***emerging*** SPARQL workloads, and
  - understand how ***real*** RDF data on the Web look like.

Characterization of SPARQL Workloads

- Emerging SPARQL workloads are ***diverse***
  - Sources of diversity:
    - Triple pattern composition
    - Structural diversity

- Emerging SPARQL workloads are ***dynamic***

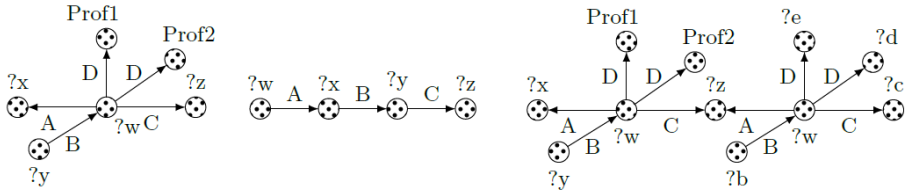
Characterization of SPARQL Workloads

- A single triple pattern can be composed in 8 different ways:

<s>	<p>	<o>
<s>	<p>	?o
<s>	?p	<o>
?s	<p>	<o>
?s	?p	<o>
?s	<p>	?o
<s>	?p	?o
?s	?p	?o

Characterization of SPARQL Workloads

- Multiple triple patterns can be combined in various ways to form
  - Linear
  - Star-shaped
  - Snowflake-shaped, or
  - Complex structures



(a) Star (b) Linear (c) Snowflake

Figure 1: Structural diversity in SPARQL.

Characterization of SPARQL Workloads

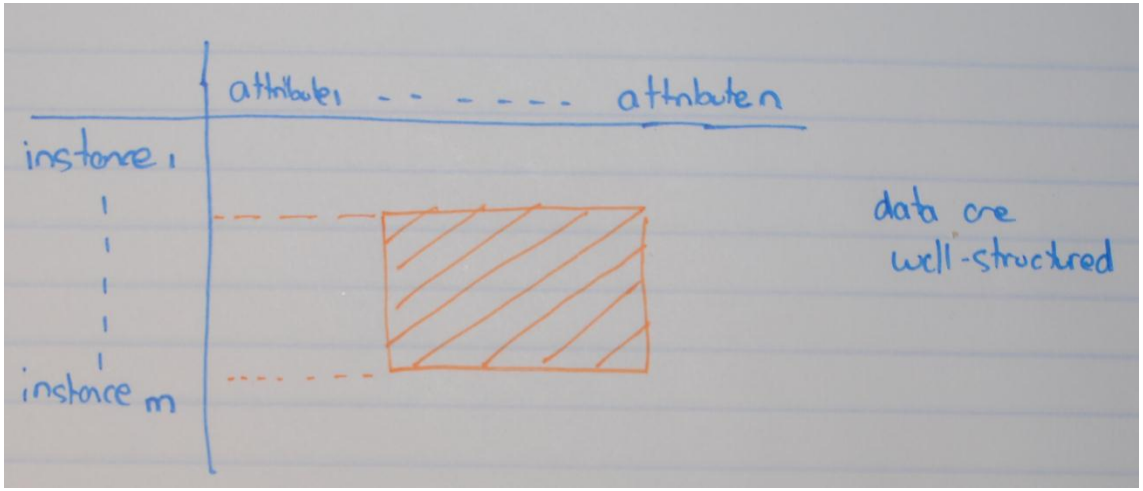
- Emerging SPARQL workloads are dynamic:
  - set of frequently queried structures change, and
  - frequently queried resources change.

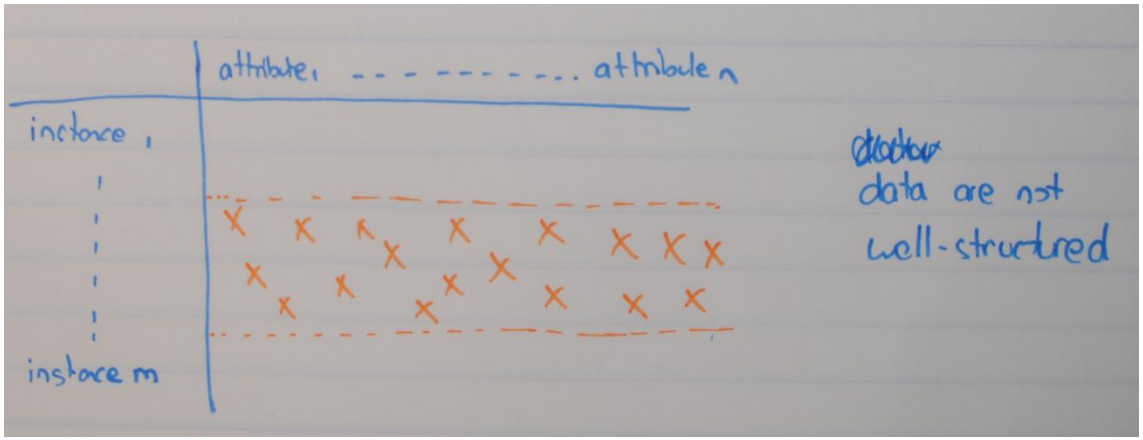
M. Arias, J. D. Fernandez, M. A. Martinez-Prieto, and P. de la Fuente. An empirical study of real-world SPARQL queries. In Proc. 1st Int. Workshop on Usage Analysis and the Web of Data, 2011.

M. Kirchberg, R. K. L. Ko, and B. S. Lee. From linked data to relevant data---time is the essence. In Proc. 1st Int. Workshop on Usage Analysis and the Web of Data, 2011.

S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. In SIGMOD Conference, pages 145--156, 2011.

RDF Data on the Web



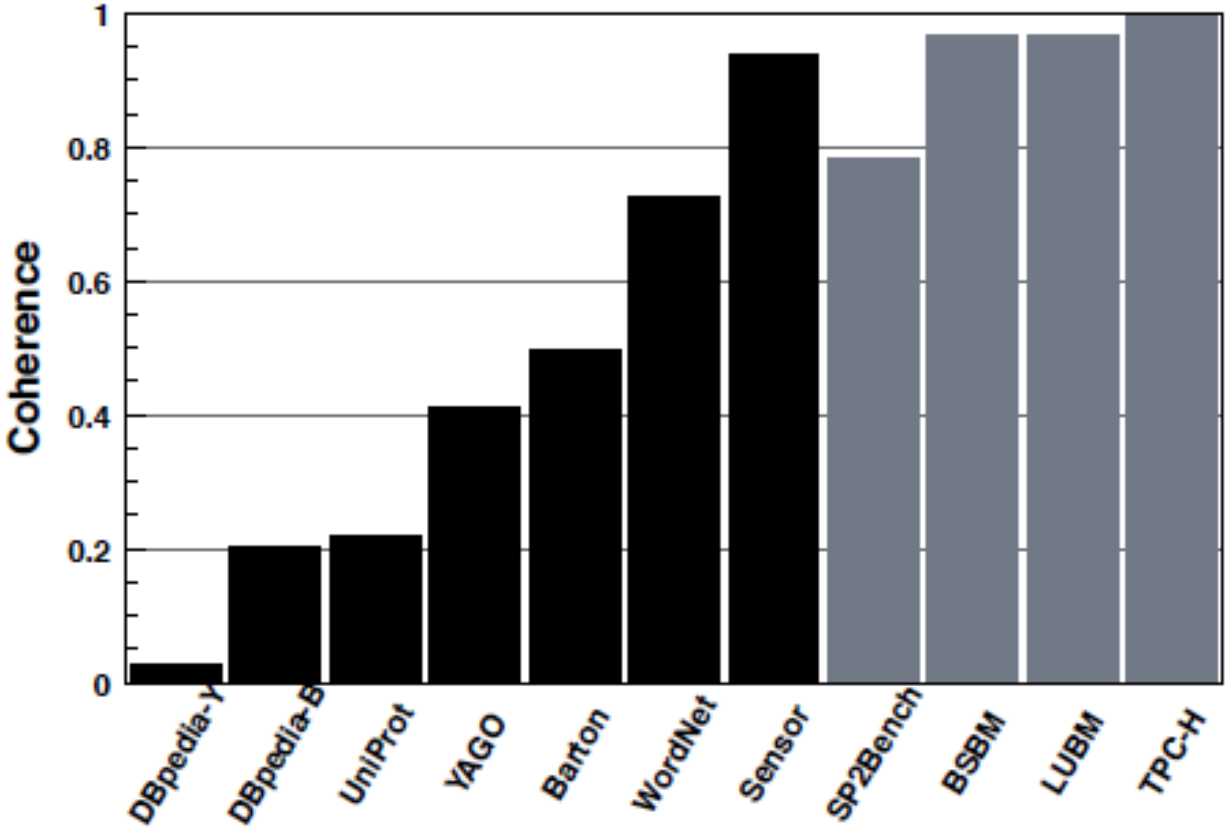


What is ***chameleon-db***?

Q: … but does this matter? After all, according to the papers that I have read, it seems like existing systems are doing a pretty good job on SPARQL benchmarks.

- **Problem**: Existing benchmarks are truly unrepresentative of the real RDF data and workloads!

Truth is …



S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. In SIGMOD Conference, pages 145--156, 2011.

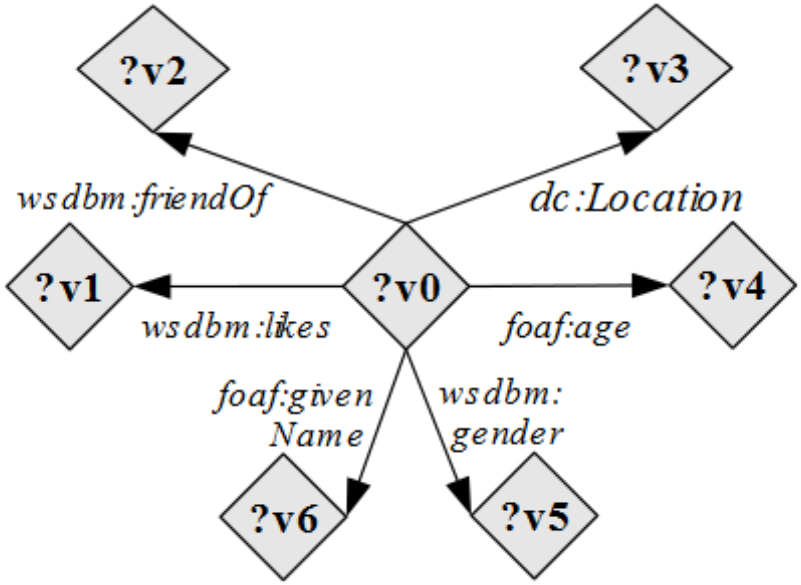
Truth is …

Q: Still… I do not really see an issue here.

- Consider the following query

C3:

```
SELECT ?v0 WHERE {
        ?v0     wsdbm:likes     ?v1 .
        ?v0     wsdbm:friendOf  ?v2 .
        ?v0     dc:Location     ?v3 .
        ?v0     foaf:age        ?v4 .
        ?v0     wsdbm:gender    ?v5 .
        ?v0     foaf:givenName  ?v6 .
}
```

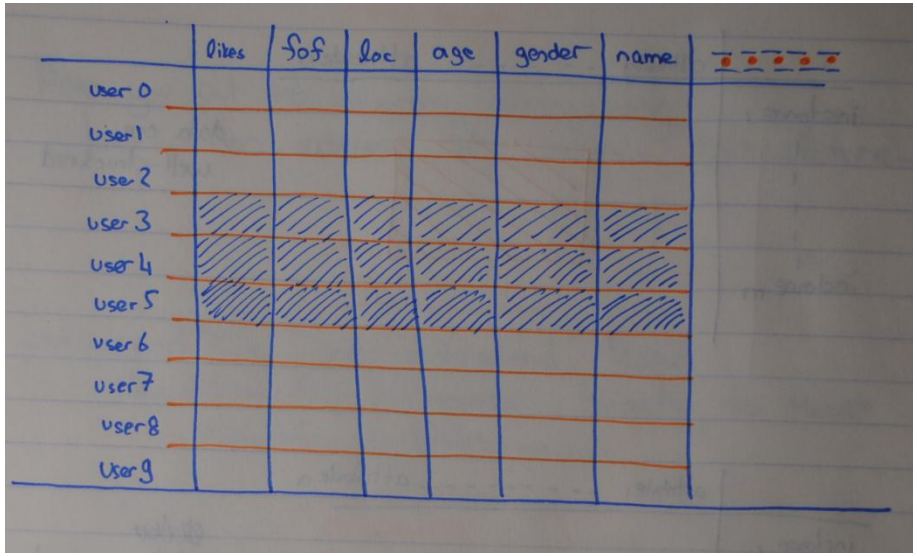


Truth is …

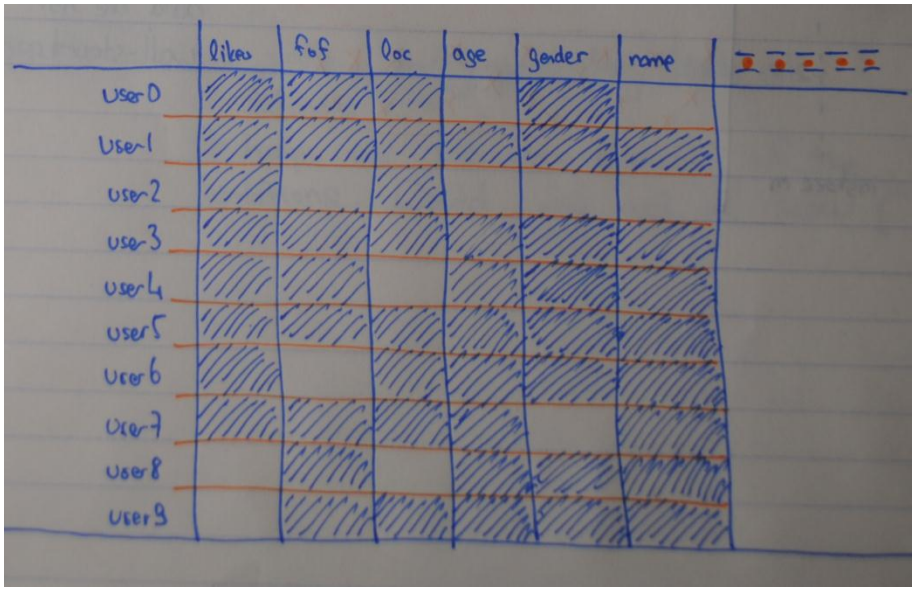
Q: Still… I do not really see an issue here.

- … and two datasets, namely, D1 and D2

D1: data are *well-structured*



D2: data are ***less*** *well-structured*



Truth is …

- Let us try to emulate how *RDF-3x* would answer this query … on D1

T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. VLDB J., 19(1):91--113, 2010.

Truth is …

- Let us try to emulate how *RDF-3x* would answer this query … on D2

- *There are lots of intermediate tuples, which do not en up in the final query result!*

Truth is …

- Now let us take a look at *gStore*
  - *gStore* creates an index over the vertices in the RDF graph such that for each vertex edges that are incident on that vertex are stored
  - Hence, given a set of edge labels, *gStore* can more easily pinpoint those vertices that have incident edges with those labels
  - As we will show in our experiments, *gStore* does a much better job for this query than other systems
  - However, for linear queries, it runs into the same problem as RDF-3x

L. Zou, J. Mo, D. Zhao, L. Chen, and M. T. Özsu. gStore: Answering SPARQL queries via subgraph matching. Proc. VLDB, 4(1):482--493, 2011.

Waterloo SPARQL Diversity Test Suite

- Designed a dataset such that
  - some entities are well-structured, while
  - others are less well-structured.

- Generated queries in 4 different categories
  - Linear
  - Star-shaped
  - Snowflake-shaped
  - Complex

https://cs.uwaterloo.ca/~galuc/wsdts/

Waterloo SPARQL Diversity Test Suite

- … while making sure that query selectivity varies
  - at the two extremes we have

	?v0
?v0 wsdbm:likes ?v1	23922
?v0 wsdbm:friendOf ?v2	39782
?v0 dc:Location ?v3	40298
?v0 foaf:age ?v4	50096
?v0 wsdbm:gender ?v5	59785
?v0 foaf:givenName ?v6	69971
overall	806

	?v0
?v0 rdf:type ?v1	125146
?v0 sorg:text ?v2	7477
%v3% wsdbm:likes ?v0	2
overall	2

https://cs.uwaterloo.ca/~galuc/wsdts/

Waterloo SPARQL Diversity Test Suite

- We generated 20 query skeletons (templates) which look like

Query Template

```
#mapping v1 wsdbm:Website uniform
SELECT ?v0 ?v2 ?v3 WHERE {
L1:     ?v0     wsdbm:subscribes        %v1% .
        ?v2     sorg:caption    ?v3 .
        ?v0     wsdbm:likes     ?v2 .
}

#mapping v0 wsdbm:City uniform
SELECT ?v1 ?v2 WHERE {
L2:     %v0%    gn:parentCountry        ?v1 .
        ?v2     wsdbm:likes     wsdbm:Product0
        ?v2     sorg:nationality        ?v1 .
}

#mapping v2 wsdbm:Website uniform
SELECT ?v0 ?v1 WHERE {
L3:     ?v0     wsdbm:likes     ?v1 .
        ?v0     wsdbm:subscribes        %v2% .
}
```

https://cs.uwaterloo.ca/~galuc/wsdts/

Waterloo SPARQL Diversity Test Suite

- A snapshot of our results

	linear:L4	star:S3	snowflake:F5	complex:C3
x-RDF-3x	**7.6**ms	8.9ms	56.4ms	timeout
gStore	53.9ms	**6.2**ms	timeout	**162.7**ms
Virtuoso	282.8ms	347.2ms	**9.4**ms	103623.7ms

Table 1: Snapshot from our experimental results (timeout is issued if query evaluation does not terminate in 15 minutes).

What is ***chameleon-db***?

Q: Okay, I understand the issue here, but cannot we choose the system that performs best for a given workload?

What is ***chameleon-db***?

- chameleon-db does not have a fixed physical design
- On the contrary,
  - the workload dictates its physical design, and
  - this physical design changes as the workload changes.

What is ***chameleon-db***?

Q: What do you mean by physical design?

1. RDF graph is logically partitioned into edge-disjoint partitions (otherwise, partitions can be arbitrary)
2. Each partition is physically stored as a record of triples, sorted on their subject attributes
3. Whenever a record is retrieved from disk, it is stored in the buffer pool as an adjacency list (more complex indexes can be built; however, this is an orthogonal work)
4. An in-memory index is created across the partitions that can answer …

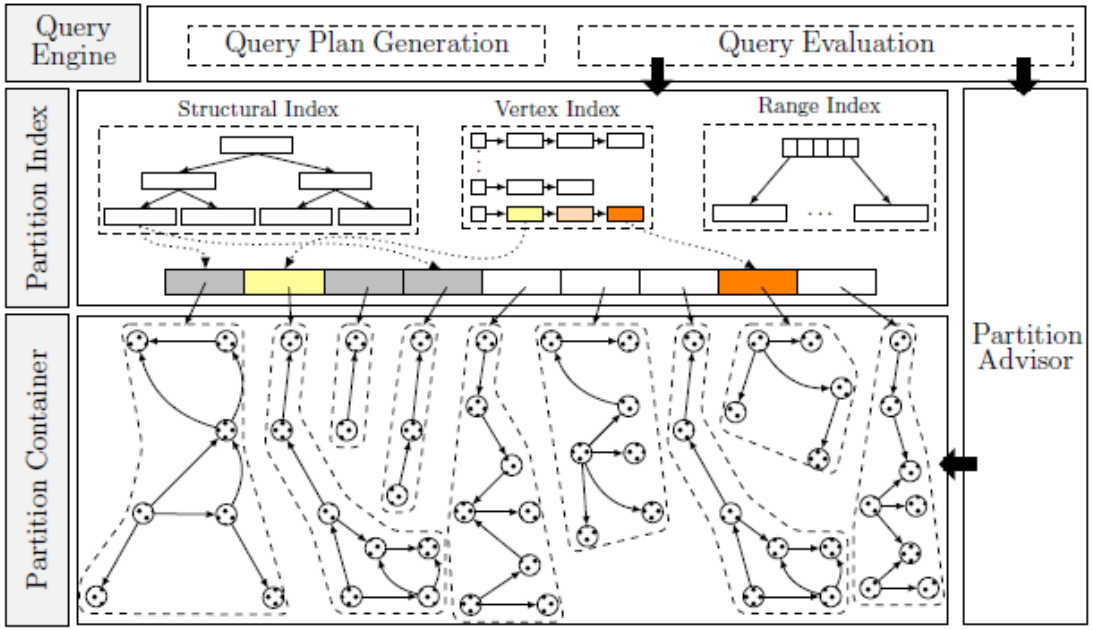


Figure 2: The architectural layout of *chameleon-db*.

Query Evaluation

- Before I step into
  i. how partitioning affects performance, and
  ii. how to compute a “good” partitioning,

  let me first explain how queries are evaluated in *chameleon-db.*

- *chameleon-db* relies on a query evaluation model that we call ***partition-restricted evaluation*** (PRE).

- In a nutshell, PRE depends on one major operation that we call ***partitioned-match***.

Query Evaluation

- Partitioned-match: $\lfloor\lfloor Q_i \rfloor\rfloor_{\boldsymbol{P}}$

  Given
  - a constrained-pattern graph (CPG) Q_i, and
  - a partitioning $\boldsymbol{P} = \{P_1, \ldots, P_n\}$ of an RDF graph G,

  we define partitioned-match as …

Query Evaluation

- Does the following statement hold?

$$\left[\left[Q\right]\right]_{G} = \left\lfloor\left\lfloor Q_i \right\rfloor\right\rfloor_{\boldsymbol{P}}$$

- This is a conscious design decision and I will explain why it is important... For now, just bear with me when I say it has important consequences on
  - indexing
  - the way partitions are updated

Query Evaluation

Given

- a CPG Q, and
- a partitioning $\boldsymbol{P} = \{P_1, \ldots, P_n\}$ of an RDF graph G,

we want to compute $\left[\left[Q\right]\right]_G$ but using partitioned-match

1. An ***oracle*** decomposes Q into a set of smaller CPGs $Q_1, \ldots, Q_k$
2. Each $Q_i \in \{Q_1, \ldots, Q_k\}$ is evaluated independently over the partitioning using partitioned-match, i.e., $\left\lfloor\left[Q_i\right]\right\rfloor_{\boldsymbol{P}}$
3. Results from Step 2 are joined

Query Evaluation

- It turns out that for
  - any RDF graph G,
  - any partitioning $\boldsymbol{P} = \{P_1, \ldots, P_n\}$ of G, and
  - any CPG Q,

  There exists a decomposition of Q into a set of CPGs $\{Q_1, \ldots, Q_k\}$ such that

$$\left[\left[Q\right]\right]_G = \left\lfloor\left\lfloor Q_1 \right\rfloor\right\rfloor_{\boldsymbol{P}} \bowtie \ \ldots \bowtie \left\lfloor\left\lfloor Q_k \right\rfloor\right\rfloor_{\boldsymbol{P}}$$

Query Evaluation

- If our stars are aligned (that is, depending on the partitioning), it may also hold that

$$\left[\left[Q\right]\right]_G = \left\lfloor\left\lfloor Q \right\rfloor\right\rfloor_{\boldsymbol{P}}$$

  which is what we want to achieve for ***most*** of the queries in the workload.

  In this presentation, I will not discuss how queries are decomposed and how query plans are generated. For a discussion about correctness and efficiency please refer to our technical report.

G. Aluç, M. T. Özsu, K. Daudjee, and O. Hartig. *chameleon-db*: a workload-aware robust RDF data management system. Technical Report CS-2013-10, University of Waterloo, 2013.

Query Evaluation

$$[\![Q]\!]_G = \lfloor\!\lfloor Q \rfloor\!\rfloor_{\boldsymbol{P}}$$

Question 1: Why do you want the property above to hold for most of the queries in the workload?

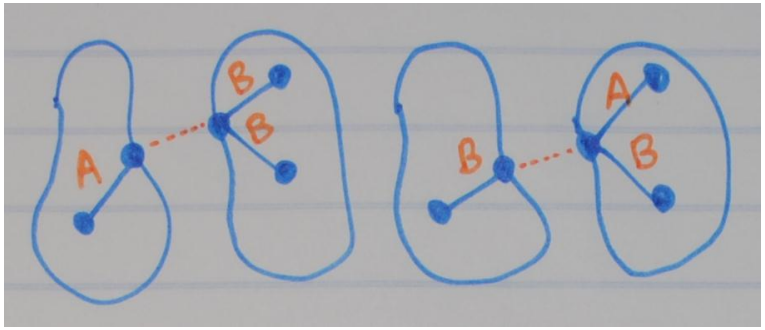
Question 2: How do you make it happen for most of the queries in the workload?

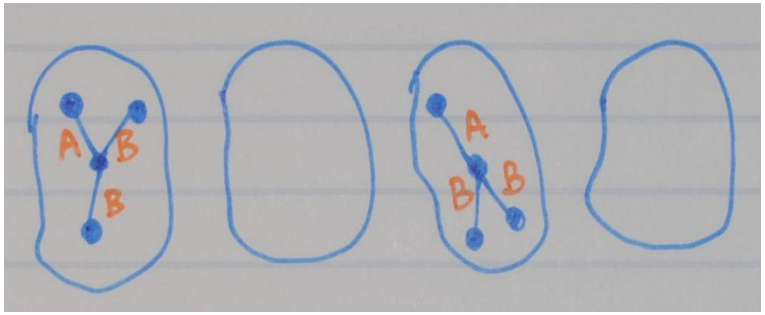
Query Evaluation

- Answer to Q1: Consider our earlier query example and (dataset) D2.
  - What happens with a decomposed evaluation?
    - Same problem as RDF-3x
  - What happens otherwise?
    - We can exploit an idea similar to that used in gStore to select only the relevant partitions (more on this later on!)

Partitioning

- Answer to Q2:
  - The way the graph is partitioned determines whether the property holds or not
- Segmentation:

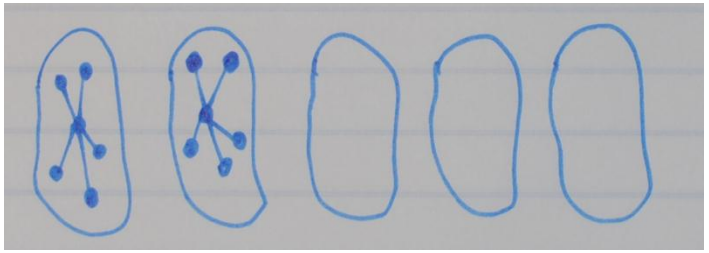


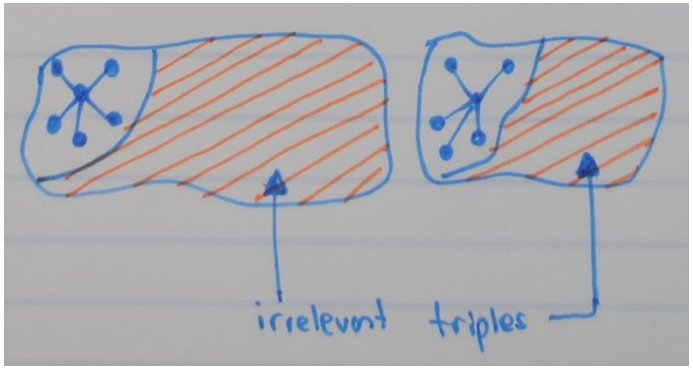


Partitioning

Q: What constitutes a “good” partitioning?

- Minimality





- Ideally, we want to find a partitioning of the RDF graph whose *segmentation* is minimal and *minimality* is maximal with respect to a given workload

Partitioning

Q: What happens when minimality is low?

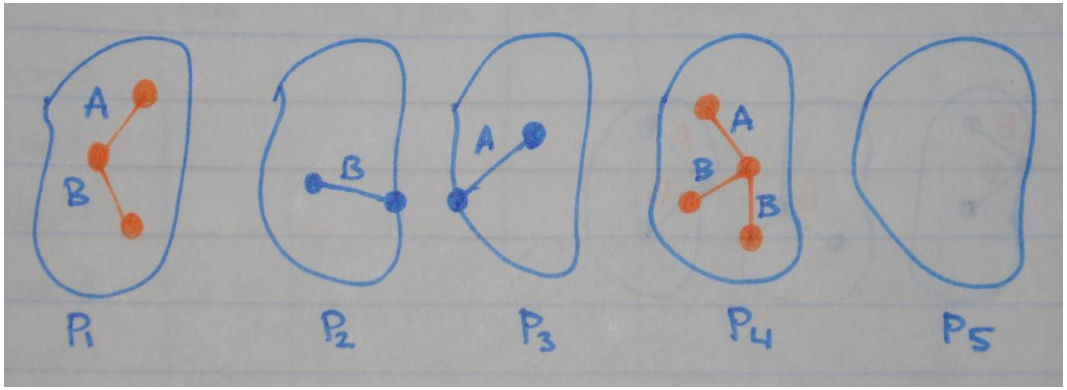
1. Bringing the records from disk (to buffer pool) is more costly.
2. When a record is retrieved from disk the first time, building the adjacency list will be more complex.
3. Search within a partition (i.e.., over the adjacency list) will be more costly.
4. You may build an index (other than an adjacency list) such that search is more efficient, however, still the overhead of building that index will be more costly when minimality is low.

Partitioning

- To compute a suitable partitioning that minimizes segmentation and maximizes minimality, we exploit a clustering algorithm whose details are in the paper.

Indexing

- Partitioned-match, $\left\lfloor [Q_i] \right\rfloor_{\boldsymbol{P}}$ has the property that without loss of generality one can prune-out (exclude) those partitions in $\boldsymbol{P}$ that do not have a match for Q_i



- What does this mean?
  - If a partition has only a partial-match to the query, then it can be safely pruned out.
  - Contrast this to $\left[[Q] \right]_G$

Indexing

Q: Why should this matter? I mean, what does it have to do with indexing?

A: This enables us to build the index adaptively (inspired by a paper work by Idreos et al. on database cracking/adaptive indexing)

- Instead of building the index across the partitions upfront, we build the partition index adaptively as each query is executed
- Initially, we assume nothing about the workload and the index is not selective at all
- However, as queries are executed, the index gets more and more selective

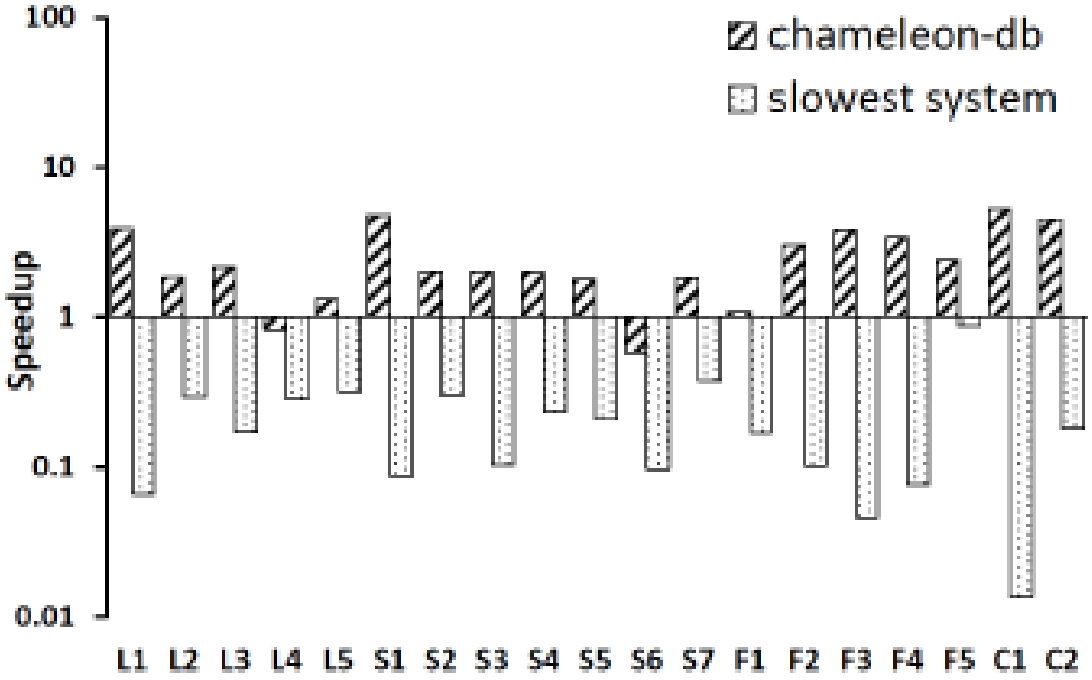
S. Idreos, M. L. Kersten, and S. Manegold. Self-organizing tuple reconstruction in column-stores. In Proc. ACM SIGMOD Int. Conf. on Management of Data, pages 297--308, 2009

Indexing

Updating the Partitioning

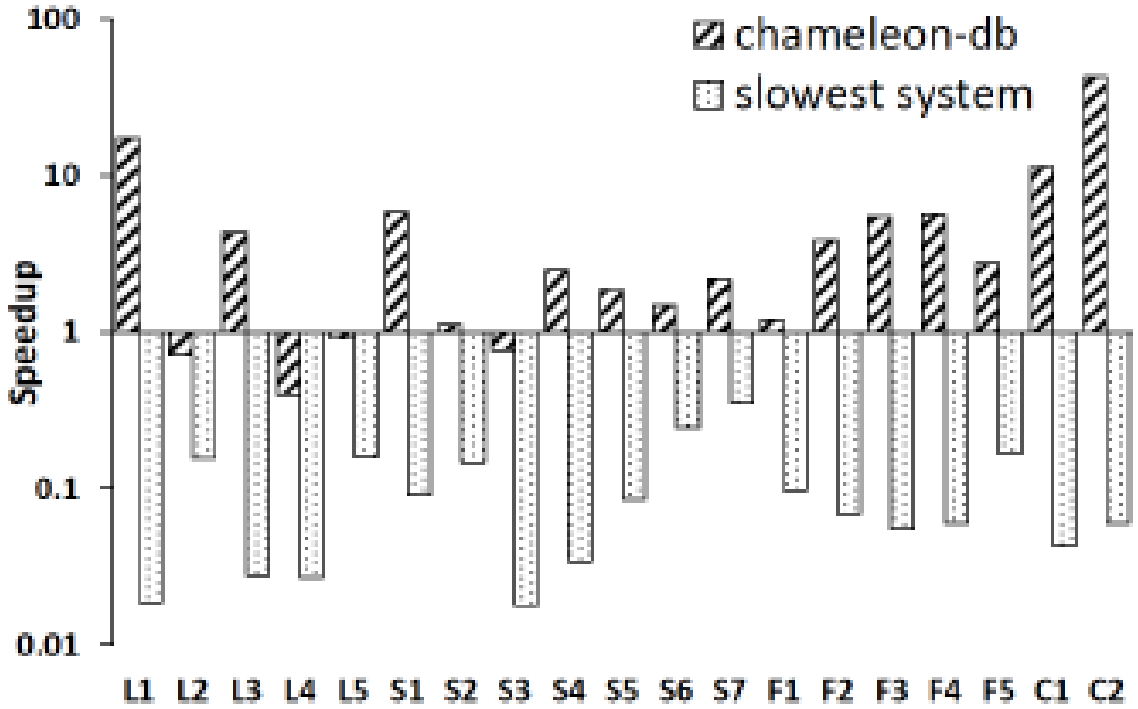
Q: You said earlier that partitioned-match also facilitates the updating of the partitions. Could you explain how?

Experimental Results



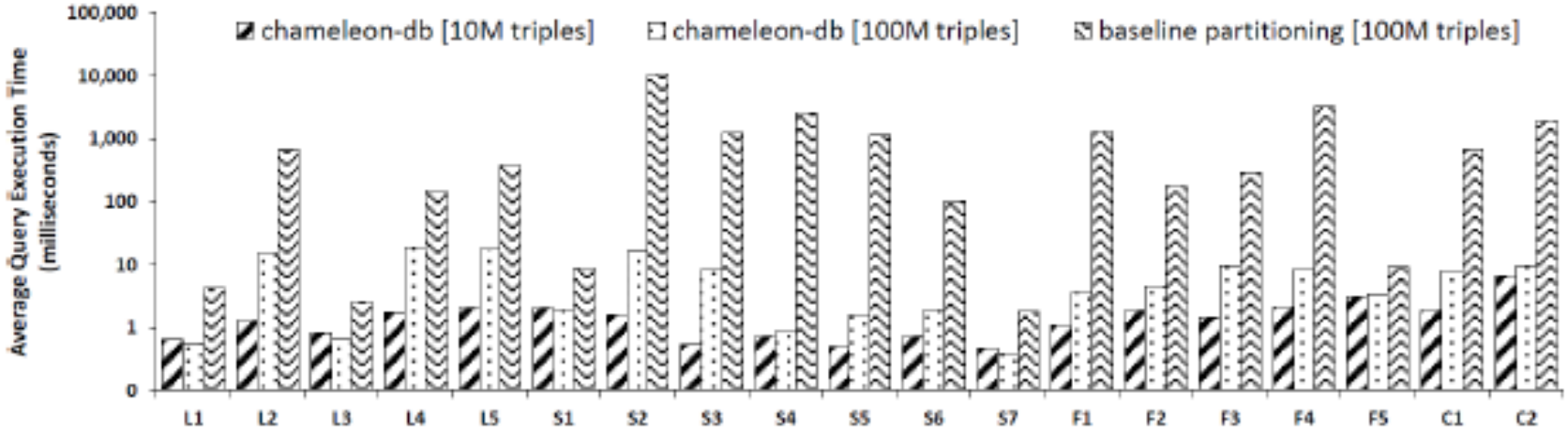
(a) Relative speedup at 10M triples

Experimental Results



(b) Relative speedup at 100M triples

Experimental Results



(d) Static performance of *chameleon-db*

Experimental Results

Figure 6: Adaptive behaviour of *chameleon-db*

Conclusions and Future Work