• Good morning everyone
• Welcome to my PhD defence
• Today, I am going to talk about a system
• ... that I implemented called chameleon-db
• In particular, I am going to discuss
  – why chameleon-db was implemented in the first place,
  – ... i.e., what sets it apart from existing solutions
  – the challenges that I encountered along the road, and
  – how I addressed these challenges
Background

Motivation

Proposal & Challenges

Evaluation

Conclusions & Future Work
Graph-Structured Data

- chameleon-db is a system for managing
- ... graph-structured data
- There are numerous use cases for graph-structured data:
  - ... from social-networks to biological networks
  - ... from representing Web data/metadata to the Internet of Things
  - ... you name them
For example, consider the Linked Open Data (LOD) cloud diagram. The LOD cloud connects hundreds of datasets. There are more than 50 billion facts stored on the LOD cloud and it is still growing. As shown in the figure, the LOD cloud is being utilized across a wide spectrum of domains.
Graph-Structured Data

- We have been able to achieve
- ... data integration
- ... at such scale
- ... thanks to data models like RDF and Property Graphs
- From now on, my presentation will focus on RDF

http://lod-cloud.net/
RDF Triple: Subject (s), Predicate (p), Object (o)

- Subject: Entity that is described
- Predicate: Feature of the entity
- Object: Value of the feature

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdb:film/2014</td>
<td>movie:director</td>
<td>mdb:director/8476</td>
</tr>
<tr>
<td>mdb:film/2685</td>
<td>movie:director</td>
<td>mdb:director/8476</td>
</tr>
<tr>
<td>mdb:film/2685</td>
<td>rdfs:label</td>
<td>&quot;A Clockwork Orange&quot;</td>
</tr>
</tbody>
</table>

- RDF consists of 3-attribute tuples called RDF triples
- Each RDF triple describes or asserts a fact about an entity in the RDF world, entities are referred to as resources
- The attributes of RDF triples are subject / predicate / and object
- or SPO
- Subject denotes...
**RDF Triple:** Subject (s), Predicate (p), Object (o)
- Subject: Entity that is described
- Predicate: Feature of the entity
- Object: Value of the feature

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdb:film/2014</td>
<td>movie:director</td>
<td>mdb:director/8476</td>
</tr>
<tr>
<td>mdb:film/2685</td>
<td>movie:director</td>
<td>mdb:director/8476</td>
</tr>
<tr>
<td>mdb:film/2685</td>
<td>rdfs:label</td>
<td>&quot;A Clockwork Orange&quot;</td>
</tr>
</tbody>
</table>

Consider a dataset

... that describes movies

Each movie is identified by a URI

URLs can be used for identifying

... resources (i.e., subjects/objects)

... as well as features (i.e., predicates)

There are also literals

... and blank nodes

... but blank nodes are ignored in this thesis
• It is natural to represent RDF data
• ... as a graph
• For sake of presentation,
• ... I will use abbreviations for
• ... URIs and literals
• ... instead of fully typing them up
Background on RDF/SPARQL

SPARQL Triple Pattern: Atomic unit in a SPARQL query
- subject/predicate/object patterns
- can be variables
- variables get instantiated w/ values from the triples in the RDF dataset

Dataset: (Google, locatedAt, NYC)
(Google, locatedAt, Waterloo)
(UWaterloo, locatedAt, Waterloo)

Query: (?x, locatedAt, Waterloo)

- SPARQL is the standard query language for RDF
- SPARQL queries are made up of triple patterns
- A SPARQL triple pattern is the atomic ...
- Read the slide
- For example, ...
Basic Graph Patterns (BGP):

- SPARQL Triple patterns can be combined in various ways to form basic graph patterns
- Mainly on S-S, S-O, O-S, O-O

- Read the slide
- SPARQL queries can be made more complex
  - ... with FILTER expressions or
  - ... by combining BGPs using
  - ... the UNION operator or
  - ... OPTIONAL graph patterns
  - ... ... (which is analogous to left outer joins)
  - ... etc.
- In this thesis, the focus
- ... is on BGPs
1. Applications that rely on RDF data are diverse [VHM+14]

• Before I motivate my thesis
• ... I would like you to get a feel for
• ... how RDF and SPARQL are being utilized
• ... in real-life
• First of all, we need to appreciate the fact that
• ... use cases of RDF are very diverse
1. Applications that rely on RDF data are diverse [VHM+14]

- The slide shows just a few
- ... of the many businesses
- ... that work with RDF/SPARQL
- Some are consumers
- ... while others are producers of
- ... RDF/SPARQL technology
1. Applications that rely on RDF data are diverse [VHM+14]

- Long story short ...
- ... RDF applications are diverse
- ... with very different business requirements
- ... thus, potentially requiring
- ... custom-tailored
- ... RDF systems/engines
2. RDF datasets are heterogeneous [DKSU11]

- Read slide
2. RDF datasets are heterogeneous [DKSU11]

The LOD cloud diagram should be
... convincing enough
... to appreciate the heterogeneity
... in real RDF datasets
For more concrete examples,
... you may read the Apples and Oranges paper
... by Tasos et al.
... which quantifies
... diversity in the LOD cloud using a measure
... called structuredness
2. RDF datasets are heterogeneous [DKSU11]

- In practice,
  - ... diversity in the structuredness of RDF datasets
  - translates to
  - ... diversity in the selectivities of queries
  - ... executed over these datasets
2. RDF datasets are heterogeneous [DKSU11]

• ... and of course
• ... selectivity matters
• ... when it comes to
• ... physical design of databases
3.1 SPARQL queries are diverse [AFMPdlF11]

- SPARQL enables triple patterns
- ... to be combined in numerous ways
3.1 SPARQL queries are diverse [AFMPdlF11]

- Resulting in
- ... linear
3.1 SPARQL queries are diverse [AFMPdlF11]
3.1 SPARQL queries are diverse [AFMPdlF11]

- ... snowflake-shaped
3.1 SPARQL queries are diverse [AFMPdLF11]

- ... and star-shaped patterns
3.1 SPARQL queries are diverse [AFMPdlF11]

- Experience shows that
- ... each type of pattern
- ... may require
- ... a completely different
- ... type of physical layout and/or
- ... indexing strategy
- ... for optimal performance
3.2 SPARQL workloads are unpredictable [KKL11]

- To make matters worse,
- ... we need to realize
- ... that SPARQL workloads
- ... are unpredictable
3.2 SPARQL workloads are unpredictable [KKL11]

- This chart
- ... which is based on
- ... information extracted from
- ... real SPARQL query logs
- ... reveals that

Extracted from query logs [BDH+15]
3.2 SPARQL workloads are unpredictable [KKL11]

- While resources such as...
- ... the Wikipedia entry for Italy
- ... receive
- ... more or less
- ... steady number of hits

Extracted from query logs [BDH+15]
3.2 SPARQL workloads are unpredictable [KKL11]

- The number of hits
- ... corresponding to other resources
- ... may fluctuate over time
Applications that rely on RDF data are diverse [VHM+14], RDF datasets are heterogeneous [DKSU11], SPARQL queries are diverse [AFMPdlF11], and SPARQL workloads are unpredictable [KKL11]
Summary so far:

- Applications that rely on RDF data are diverse [VHM+14],
- RDF datasets are heterogeneous [DKSU11],
- SPARQL queries are diverse [AFMPdlF11], and
- SPARQL workloads are unpredictable [KKL11]

Question

Are existing RDF data management systems capable of handling the diversity and unpredictability in workloads?
1 Background

2 Motivation

3 Proposal & Challenges

4 Evaluation

5 Conclusions & Future Work
Motivation

Problems with Existing Approaches

- To quantify how well existing systems perform
- ... across SPARQL workloads
- ... with diverse selection of queries
- experiments were conducted
Motivation

Problems with Existing Approaches

- For these experiments,
- ... I developed and used the WatDiv benchmark
- ... because
- existing SPARQL benchmarks
- ... even those that *claim* to be based on real query logs
- ... are not truly diverse
- A thorough comparison and analysis can be found in my thesis

WatDiv 100M triples, stress testing workload [AHOD14]
Motivation

Problems with Existing Approaches

• Five systems have been evaluated in these experiments
• RDF-3x is the de-facto research prototype
• Virtuoso Open Source (VOS) is the open source version
• ... of a popular commercial engine
• VOS 6.1 is a row-store, and
• VOS 7.1 is a hybrid column-store / graph engine
• MonetDB is a state-of-the-art column-store,
• ... which was incorporated with Abadi et al.’s
• ... techniques for RDF layout
• Lastly, 4Store is used,
• ... which is another popular commercial RDF store
Motivation

Problems with Existing Approaches

The experiments revealed some important results:

- Each system is optimal
- ... only for a small percentage of queries
- Most systems time-out on
- ... a significant number of queries
- There can be orders of magnitude difference
- ... between the best and worst performance of a system

WatDiv 100M triples, stress testing workload [AHOD14]
Motivation

Problems with Existing Approaches

WatDiv 100M triples, stress testing workload \cite{AHOD14}

- There can be orders of magnitude difference
- ... between the best and worst system for each query
Motivation

Problems with Existing Approaches

Existing systems are built on top of one of the following physical layouts, but each layout is optimized for a different type of query, and there is little overlap:

- **Single Table Representation**: [CDD$^+$04, ACZH10, HD05, NW10, WKB08, HLS09]
- **Group-by-Predicates**: [AMMH09, ACZH10, Erl12, IGN$^+$12]
- **Group-by-Entities**: [BDK$^+$13, Wil06, EM07]
- **Group-by-Vertices**: [BPS09, HS13a, HAR11, ZÖC$^+$14]

- The problem is ...
Motivation

Problems with Existing Approaches: Single Table

- In the single-table representation
- ... data are stored in a 3-attribute table
- In variants of this representation
- ... multiple indexes and/or copies of the table
- ... can be maintained for better performance
Motivation

Problems with Existing Approaches: Single Table

<table>
<thead>
<tr>
<th>S</th>
<th>P</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>A</td>
<td>v21</td>
</tr>
<tr>
<td>v2</td>
<td>A</td>
<td>v98</td>
</tr>
<tr>
<td>v3</td>
<td>A</td>
<td>v250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v21</td>
<td>B</td>
<td>v20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v98</td>
<td>C</td>
<td>v30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v250</td>
<td>D</td>
<td>v40</td>
</tr>
</tbody>
</table>

Ideal for: Queries with few number of joins and where intermediate results are not fragmented

- Read slide

2015-09-21 PhD Defence

Motivation

Problems with Existing Approaches: Single Table
Motivation

Problems with Existing Approaches: Single Table

Not suitable for: Queries with many joins and where intermediate results are fragmented
Motivation

Problems with Existing Approaches: Group-by-Predicates

- It has been argued that grouping data can improve performance.
- One possible grouping is based on the predicates in the dataset.
- Where...
- A 2-attribute table is created...
- For each predicate in the dataset...
- And these tables are stored...
- In a column-store...
Problems with Existing Approaches: Group-by-Predicates

- Read slide
- ... that is, the predicate patterns
- ... do not contain variables

Ideal for: Queries with aggregation and where all predicate patterns are bound
Problems with Existing Approaches: Group-by-Predicates

- Not suitable for: Queries with at least one unbound predicate pattern

Read slide
Problems with Existing Approaches: Group-by-Entities

- Data can be grouped
- ... also based on the “entity sets”
- ... in the dataset
- There are multiple ways of determining
- ... what constitutes an “entity set”
- One way is to bring a database administrator
- ... into the picture
- DB2-RDF relies on automatic techniques
- ... to determine the entity sets
Problems with Existing Approaches: Group-by-Entities

Ideal for: Queries over well-structured datasets that do not involve too many entity sets

• Read slide
Problems with Existing Approaches: Group-by-Entities

Not suitable for: Queries over not well-structured datasets that involve many entity sets

- Read slide
Motivation

Problems with Existing Approaches: Group-by-Vertices

- RDF can be naturally represented as a graph
- ... and an adjacency-list is one way of
- ... physically storing a graph
- RDF data management systems like
- ... gStore follow this approach
Motivation

Problems with Existing Approaches: Group-by-Vertices

• Since in the group-by-vertices representation
• ... RDF triples that share the same vertex
• ... are grouped-together
• ... star-shaped queries are easy to answer

Ideal for: Star-shaped queries
Problems with Existing Approaches: Group-by-Vertices

- On the other hand, ...
- Read slide

Not suitable for: Linear queries
Motivation

Problems with Existing Approaches

• In short,
  • ... no one size fits all
  • Existing systems are not
  • ... robust across workloads
  • ... that have diverse queries

WatDiv 100M triples, stress testing workload [AHOD14]
Motivation

Problems with Existing Approaches

- Furthermore, it is not possible to...
- ...update the physical layout of...
- ...existing systems at runtime
- ...should the workloads change
- In cases, it might be necessary...
- ...to switch from a row-oriented...
- ...representation to a columnar one
- ...at runtime
Motivation

Problems with Existing Approaches

Question

Is it possible to build a more robust RDF data management system?

WatDiv 100M triples, stress testing workload [AHOD14]

• Then, the question is ...
• Read slide
Motivation

Problems with Existing Approaches

Question

If so, how?

WatDiv 100M triples, stress testing workload [AHOD14]

- Read slide
- The answers to these questions is what my thesis is about
1 Background

2 Motivation

3 Proposal & Challenges

4 Evaluation

5 Conclusions & Future Work
Proposal & Challenges

Space of Solutions Explored

Conventional RDF Data Management Systems

RDF View Materialization, Modern Distributed RDF Systems

Solutions Explored in this Thesis

- Schema fixed at the time the DBMS code is implemented, e.g., RDF-3x [NW10], gStore [ZOC+14], vertical partitioning [AMMH09]
- Schema fixed once data are loaded, e.g., Jena2 [Wil06], DB2-RDF [BDK+13]

Workload-Awareness

- It must be evident
- ... from my previous slides that
- ... conventional methods are not
- ... suitable for real
- ... RDF applications
- These methods are neither
- ... workload-aware nor
- ... runtime-adaptive
### Space of Solutions Explored

#### Conventional RDF Data Management Systems
- **Schema fixed at the time the DBMS code is implemented**, e.g., RDF-3x [NW10], gStore [ZÖC’14], vertical partitioning [AMMH09]
- **Schema fixed once data are loaded**, e.g., Jena2 [Wil06], DB2-RDF [BDK’13]

#### Solutions Explored in this Thesis
- **Workload-Oblivious**
- **Workload-Aware**

- **Runtime-Adaptiveness**
  - **Static**
  - **Adaptive**

**PhD Defence Proposal & Challenges**

- In fact, ...
- ... in systems like RDF-3x and gStore
- ... the schema is fixed already when ...
- In systems like Jena2 and DB2-RDF, ...
- ... the schema is fixed once ...
- ... and only minor changes are allowed at runtime
- Otherwise, implementing the schema changes ...
- ... can take hours

Conventional RDF Data Management Systems

- RDF View Materialization, Modern Distributed RDF Systems

Solutions Explored in this Thesis

Workload-Oblivious

Workload-Aware

Runtime-Adaptiveness

Static

Adaptive

- Materialized views for RDF [CL10, GKL11]
- Workload-aware distribution techniques [HS13b, GHS14]

- While view materialization techniques offer some form of relief
- ... they are not truly runtime-adaptive
- In fact, computing the materialized views alone can take more than half an hour
This thesis is the first to explore this solution space for RDF systems.

What about relational techniques? See next slide.
Proposal & Challenges

Space of Solutions Explored

Conventional Techniques in Self-Tuning Databases [CN07]

Database Cracking [IKM07]

Solutions Explored in this Thesis

Infeasible

Aggressive

Local

Conventional Techniques in Self-Tuning Databases [CN07]

Lazy

Global

Tuning Method

• More specifically,
• ... can we rely on conventional techniques
• ... in self-tuning databases?
Proposition & Challenges

Space of Solutions Explored

- Conventional Techniques in Self-Tuning Databases [CN07]
- Database Cracking [IKM07]

Infeasible Solutions Explored in this Thesis

Aggressive
- Global
- Local
  - Conventional Techniques in Self-Tuning Databases [CN07]

Lazy
- Global
- Local
  - Database Cracking [IKM07]

- Not really, because
- First of all,
  - ... these techniques are scalable only if
  - ... the schema changes are minor

PhD Defence

Proposal & Challenges

Space of Solutions Explored

2015-09-21
Proposal & Challenges

Space of Solutions Explored

Conventional Techniques in Self-Tuning Databases [CN07]

Database Cracking [IKM07]

Infeasible Solutions Explored in this Thesis

Second, these techniques try to tune...

... the database aggressively

That is,

... if an index needs to be built,

... it will be built over the whole table

or

... if an index needs to be dropped,

... the whole index will be dropped
In that respect, database cracking offers... a more scalable solution... because tuning is lazy. That is, ... only very small chunks... of the database are tuned... at each tuning step... where tuning might take place... during/after the execution of every query.
Proposal & Challenges

Space of Solutions Explored

Conventional Techniques in Self-Tuning Databases [CN07]

Database Cracking [IKM07]

Infeasible

Solutions Explored in this Thesis

Aggressive

Lazy

Global

Local

Tuning Method

• On the other hand,
• ... database cracking is applicable only
• ... to in-memory arrays
• ... in a column-store
Proposal & Challenges

Space of Solutions Explored

Conventional Techniques in Self-Tuning Databases [CN07]

Database Cracking [IKM07]

Infeasible

Solutions Explored in this Thesis

Aggressive

Local

Conventional Techniques in Self-Tuning Databases [CN07]

Database Cracking [IKM07]

Lazy

Global

Tuning Method

PhD Defence

Proposal & Challenges

Space of Solutions Explored

• Instead, techniques are needed
• ... that can handle more global
• ... changes in the schema
• ... such as dynamically
• ... switching from
• ... a row-oriented representation
• ... to a columnar representation
Proposal & Challenges

Space of Solutions Explored

Conventional Techniques in Self-Tuning Databases [CN07]

Database Cracking [IKM07]

Infeasible

Solutions Explored in this Thesis

Lazy

Global

Local

Aggressive

Tuning Method

which brings me to the

... space of solutions

... explored in this thesis

These solutions are

... ... not only lazy

... ... but also can support

... ... global schema changes
When designing and implementing an RDF data management system, assume nothing about the workload upfront. Organize data dynamically and purely based on the workload. Divide the database into very small chunks, tune each chunk independently and in a lazy-fashion based on the workload.

- In particular,
- ... my proposal is that ...
- When designing ...
- ... unconventional, but
- ... as discussed before, Web workloads are unpredictable
- ... and experiments show that fixing the layout upfront
- ... is not a good option
- Continue reading the slide
Characteristics:
- Records are not necessarily of fixed length
- Records are not grouped into tables
- Records do not necessarily share the same set of RDF predicates
- Each record represents a very tiny part of the RDF graph
My Proposal: Group-by-Query

- This way ...
My Proposal: Group-by-Query

• ... parts of the database
My Proposal: Group-by-Query

• ... can be tuned for linear queries
My Proposal: Group-by-Query

• ... while the remaining parts ...

Query 2
My Proposal: Group-by-Query

- ... for, say, star-shaped queries
My Proposal: Group-by-Query

Query 2
My Proposal: Group-by-Query

Advantages

- Data are physically clustered for most queries in the workload,
- Better pruning by the indexes,
- Fewer intermediate result tuples [AOD14]

- This representation,
- ... which I will call
- ... the group-by-query representation
- ... ... just to be consistent with the former notation
- ... has multiple advantages
- Read slide
- The details are discussed in the thesis
- ... and in our PVLDB 2014 paper
Determining a suitable physical layout for the current workload is not trivial.

Overhead of updating the physical layout can be high for online tuning.

Correct and efficient query evaluation is difficult when the underlying physical layout is frequently changing.

- Of course,
- ... there is no free-lunch
- ... and these ideas do not have a
- ... trivial implementation, either
- In particular, in this thesis,
- ... the following challenges are addressed
- Read slide
Determining a suitable physical layout for the current workload is not trivial
Overhead of updating the physical layout can be high for online tuning
Correct and efficient query evaluation is difficult when the underlying physical layout is frequently changing

BREADTH

While each of these three challenges ...
... receive adequate attention
... in the thesis
Determining a suitable physical layout for the current workload is not trivial.

Overhead of updating the physical layout can be high for online tuning.

Correct and efficient query evaluation is difficult when the underlying physical layout is frequently changing.

The emphasis is on the first item

... which constitutes the depth

... of my thesis
## Proposal & Challenges

### Challenges

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Proposed Solution</th>
</tr>
</thead>
</table>
| Determining a suitable physical layout for the current workload is not trivial | Database divided into very small chunks where each chunk can be tuned for a different workload. To determine a suitable layout for each chunk, two algorithms are proposed:  
  - Hierarchical clustering algorithm (periodic)  
  - Clustering algorithm based on a new locality-sensitive hashing scheme (online) |

- Read slide and elaborate
- ... getting closer to fully automated
- ... and dynamic updating of the physical layout
- ... but not quite there yet
<table>
<thead>
<tr>
<th>Challenge</th>
<th>Proposed Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overhead of updating the physical layout can be high for online tuning</td>
<td>Lazy updating of the storage layout</td>
</tr>
<tr>
<td></td>
<td>Lazy updating of the indexes</td>
</tr>
</tbody>
</table>
### Challenges

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Proposed Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query evaluation is difficult when the underlying physical layout is frequently changing</td>
<td></td>
</tr>
<tr>
<td>• Returning correct query results</td>
<td></td>
</tr>
<tr>
<td>• Trade-off between efficiently maintaining schema information vs. efficiently generating query plans</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Schema-oblivious query evaluation (as a fallback mechanism)</td>
</tr>
<tr>
<td></td>
<td>• Query rewrite rules that enable efficiently generating efficient query plans while using as little schema information as possible</td>
</tr>
</tbody>
</table>

- Read slide and elaborate
• Prototype system [AODH13]
• More than 35,000 lines of code in C++ in Linux (excluding SPARQL parser and 3rd party tools)

• These ideas have been
• ... implemented and tested
• ... in a proof-of-concept prototype ...

• Explain system stack
1. Background

2. Motivation

3. Proposal & Challenges

4. Evaluation

5. Conclusions & Future Work
Evaluation

Note:

- Numerous experiments are performed in the thesis; this presentation focuses on just one experiment
  - Quantified analysis of existing SPARQL benchmarks
  - Comparison of WatDiv with existing SPARQL benchmarks
  - End-to-end evaluation of CDB and comparison with state-of-the-art systems
    - Hierarchical clustering algorithm
    - Clustering using TunableLSH (i.e., the new LSH function)
  - Evaluation of individual components of CDB, in particular,
    - Physical clustering,
    - Indexing and
    - Query evaluation techniques
  - Evaluation of techniques such as TunableLSH in applications beyond CDB
Setup:
- WatDiv, stress testing workloads (125 query templates)
- For each query template, generate warmup (100) + test (500) queries
- Execute first 100 queries using completely fragmented clustering in CDB, then let Storage Advisor compute a better clustering
- Results for hierarchical clustering algorithm are reported
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>RDF-3x</th>
<th>VOS [6.1]</th>
<th>VOS [7.1]</th>
<th>MonetDB</th>
<th>4Store</th>
<th>CDB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WatDiv 10M triples</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query execution time (ms)</td>
<td>18.8</td>
<td>44.0</td>
<td>24.5</td>
<td>17.0</td>
<td>93.0</td>
<td>4.7</td>
</tr>
<tr>
<td>(geometric mean)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of query templates where system is fastest</td>
<td>1.1%</td>
<td>0.0%</td>
<td>2.2%</td>
<td>16.3%</td>
<td>0.0%</td>
<td><strong>80.4%</strong></td>
</tr>
</tbody>
</table>

92 / 125 query templates

- Emphasize CDB is fastest across
- ... the largest percentage of queries
- ... and it has the smallest
- ... mean query execution time
As you can see, CDB is more robust.
### WatDiv 100M triples

<table>
<thead>
<tr>
<th>RDF-3x</th>
<th>VOS [6.1]</th>
<th>VOS [7.1]</th>
<th>MonetDB</th>
<th>4Store</th>
<th>CDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query execution time (ms) (geometric mean)</td>
<td>71.4</td>
<td>210.3</td>
<td>96.4</td>
<td>62.7</td>
<td>767.2</td>
</tr>
<tr>
<td>% of query templates where system is fastest</td>
<td>9.7%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>40.3%</td>
<td>1.4%</td>
</tr>
</tbody>
</table>

76 / 125 query templates

- Similar statements hold
- ... for the larger dataset
- Except some relative
- ... reduction in performance
- If you are interested,
- ... ask me later and I will elaborate
Evaluation of systems on WatDiv 100M triples, stress testing workloads
Conclusions & Future Work
Conclusions & Future Work

Conclusions

It is possible to build runtime-adaptive RDF data management systems:

- Divide the database into very small chunks, tune each chunk independently and in a lazy-fashion based on the workload
- Develop workload-adaptive indexes for RDF (need to go beyond database cracking [IKM07])
- Develop query evaluation techniques for when little is known about the underlying schema and/or when the schema is frequently updated

• Read slide
Future Work

• Extending the Tuning Model
• Support for and Optimizations Beyond BGPs
• Improving Existing Query Evaluation/Optimization Techniques for BGPs
• Extending Indexing Techniques
• Extending Adaptive Techniques
• Distribution

- Read slide
Questions?
Medha Atre, Vineet Chaoji, Mohammed J. Zaki, and James A. Hendler.
Matrix “bit” loaded: a scalable lightweight join query processor for rdf data.

Mario Arias, Javier D. Fernández, Miguel A. Martínez-Prieto, and Pablo de la Fuente.
An empirical study of real-world SPARQL queries.
CoRR, abs/1103.5043, 2011.

Diversified stress testing of RDF data management systems.
Daniel J. Abadi, Adam Marcus, Samuel R. Madden, and Kate Hollenbach.
SW-Store: a vertically partitioned DBMS for semantic web data
management.

Workload matters: Why RDF databases need a new design.

Güneş Aluç, M. Tamer Özsu, Khuzaima Daudjee, and Olaf Hartig.
chameleon-db: a workload-aware robust RDF data management
system.
References IV

Matthias Bröcheler, Andrea Pugliese, and V. S. Subrahmanian. DOGMA: A disk-oriented graph matching algorithm for RDF databases.

Jena: implementing the semantic web recommendations.

Roger Castillo and Ulf Leser.
Selecting materialized views for RDF data.
Surajit Chaudhuri and Vivek Narasayya.
Self-tuning database systems: a decade of progress.
In Proc. 33rd Int. Conf. on Very Large Data Bases, pages 3–14, 2007.

Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea.
Apples and oranges: a comparison of RDF benchmarks and real RDF datasets.

Orri Erling and Ivan Mikhailov.
RDF support in the Virtuoso DBMS.


References VII

Andreas Harth and Stefan Decker.
Optimized index structures for querying RDF from the Web.

S. Harris, N. Lamb, and N. Shadbolt.
4store: The design and implementation of a clustered RDF store.

Katja Hose and Ralf Schenkel.
WARP: Workload-aware replication and partitioning for RDF.
In Proc. 4th Int. Workshop on Data Engineering Meets the Semantic Web, pages 1–6, 2013.
Katja Hose and Ralf Schenkel.
WARP: workload-aware replication and partitioning for RDF.
In Proc. Workshops of the 29th IEEE Int. Conf. on Data Engineering, pages 1–6, 2013.

Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, and Martin L. Kersten.
MonetDB: Two decades of research in column-oriented database architectures.

Stratos Idreos, Martin L. Kersten, and Stefan Manegold.
Database cracking.
Markus Kirchberg, Ryan K. L. Ko, and Bu-Sung Lee.
From linked data to relevant data – time is the essence.

Thomas Neumann and Gerhard Weikum.
The RDF-3X engine for scalable management of RDF data.

Ruben Verborgh, Olaf Hartig, Ben De Meester, Gerald Haesendonck,
Laurens De Vocht, Miel Vander Sande, Richard Cyganiak, Pieter Colpaert, Erik Mannens, and Rik Van de Walle.
Querying datasets on the web with high availability.
Kevin Wilkinson.
Jena property table implementation.

Cathrin Weiss, Panagiotis Karras, and Abraham Bernstein.
Hexastore: sextuple indexing for semantic web data management.

Lei Zou, M. Tamer Özsu, Lei Chen, Xuchuan Shen, Ruizhe Huang, and Dongyan Zhao.
gStore: a graph-based SPARQL query engine.