NoSQL Schema Evolution and Big Data Migration at Scale

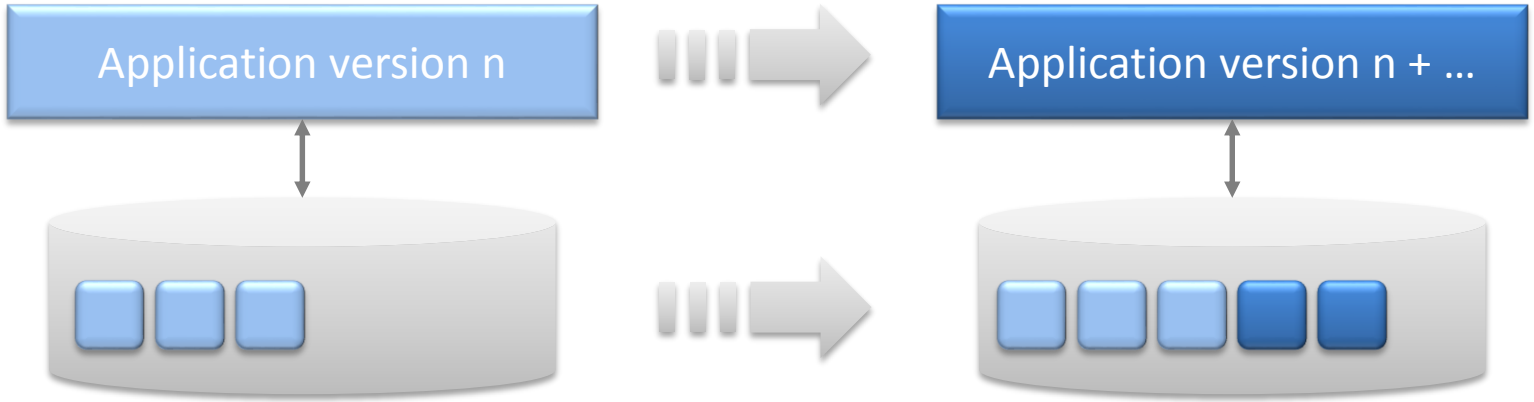
Uta Störl
Darmstadt University of Applied Sciences, Germany

April 2018

Joint work with Meike Klettke (University of Rostock, Germany)
and Stefanie Scherzinger (OTH Regensburg, Germany)

Motivation

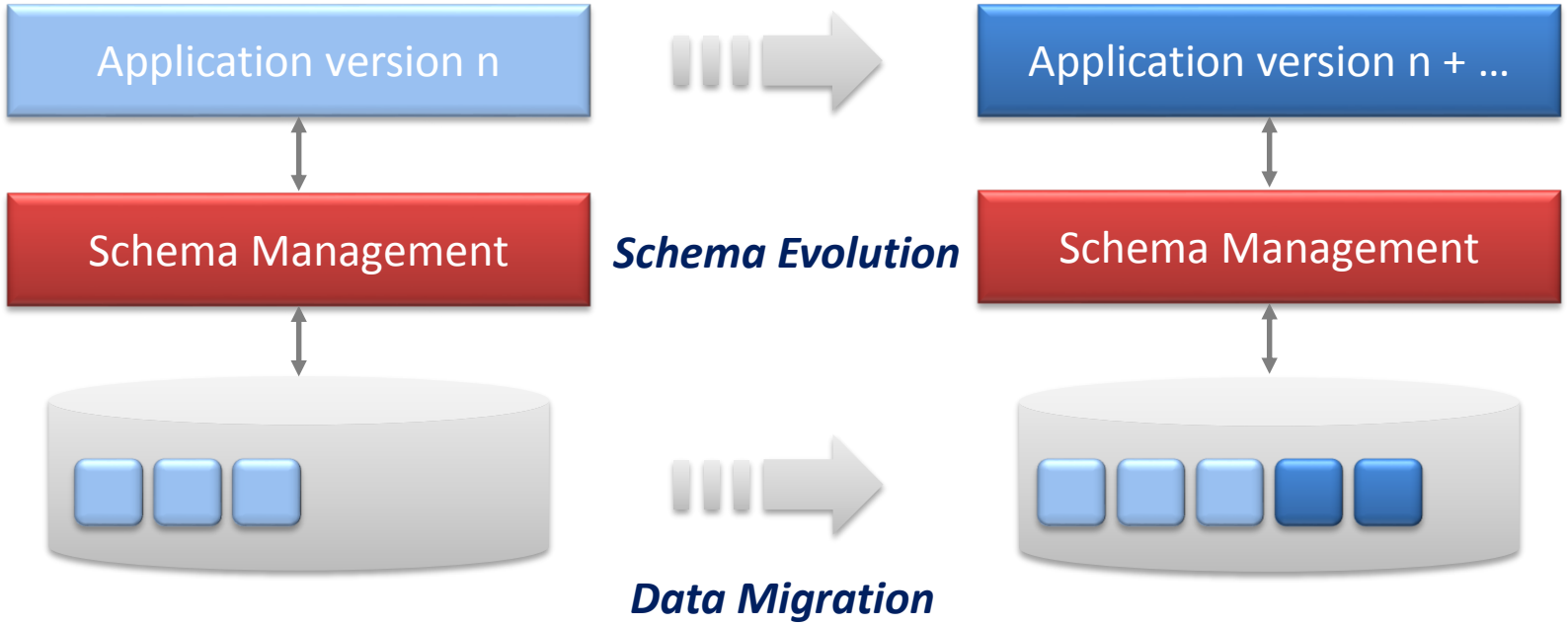
- Agile software development with frequently schema changes (weekly up to daily!) ⇨ Schema-flexible NoSQL databases ☺



- However, how to migrate variational data in the productive database?
  - State of the art: Within the application code ☹
  - **Schema management for NoSQL database systems necessary!**

Schema Management for NoSQL Databases

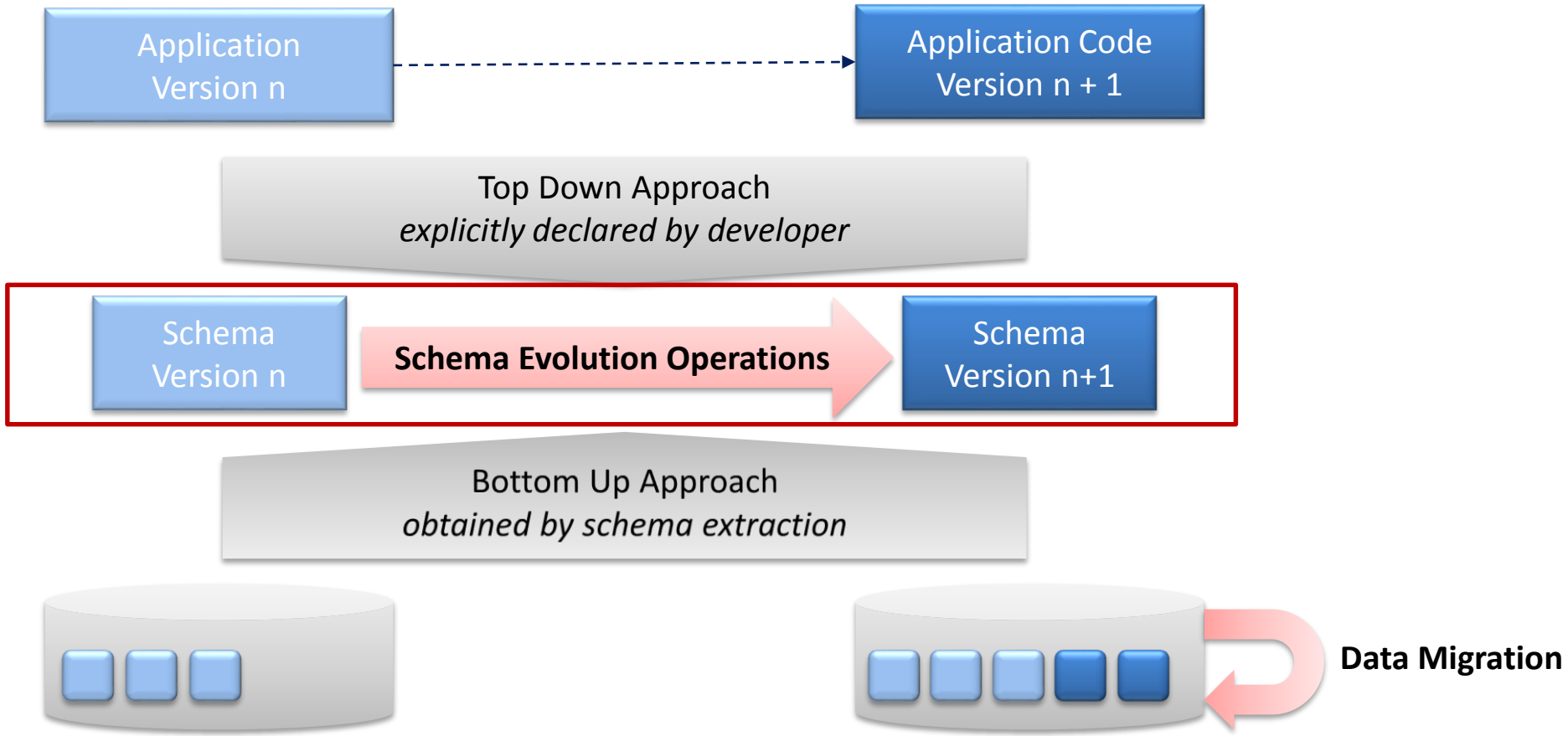
- (Optional) **schema management for NoSQL databases**



⇨ **Two main tasks**

- **Schema evolution management**
- **Data migration** as safe process based on schema evolution management

Vision: Curating Variational Data End-to-End

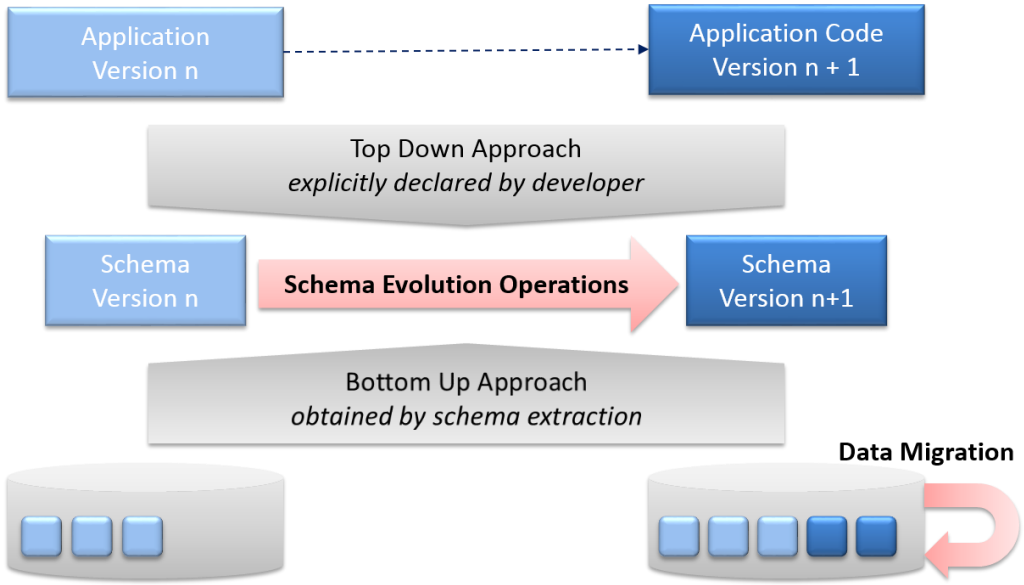


NoSQL Schema Evolution Language

- Schema evolution language for describing structural changes of the data

- **Single type** operations:
  - **add** property
  - **rename** property
  - **delete** property

- **Multi-type** operations:
  - **copy** property (denormalization)
  - **move** property (refactoring)



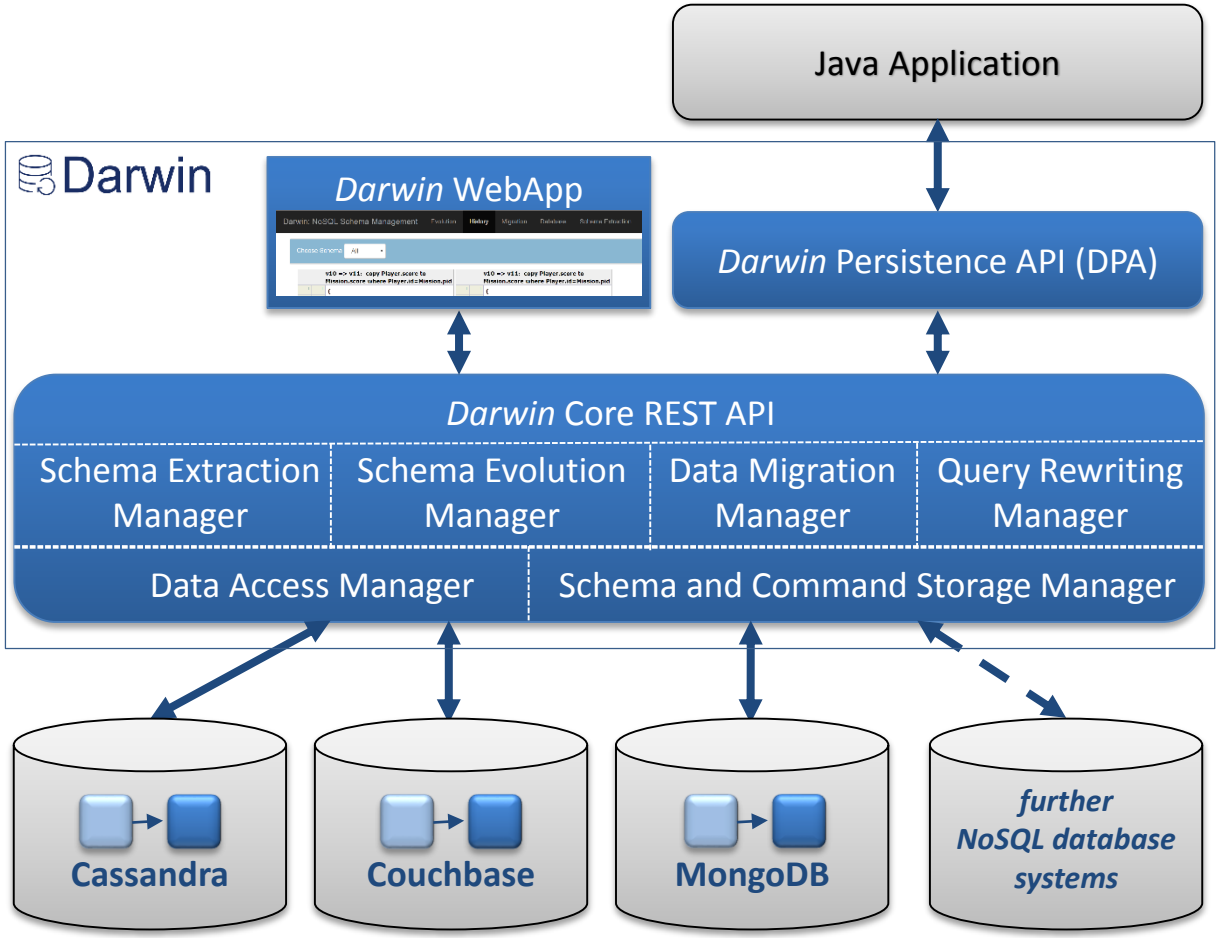
Introduced in: S. Scherzinger, M. Klettke, U. Störl: Managing Schema Evolution in NoSQL Data Store, DBPL@VLDB, Italy, 2013

Our Approach: Supporting Schema Management and Data Migration with *Darwin*

***Darwin* supports the complete schema management life cycle:**

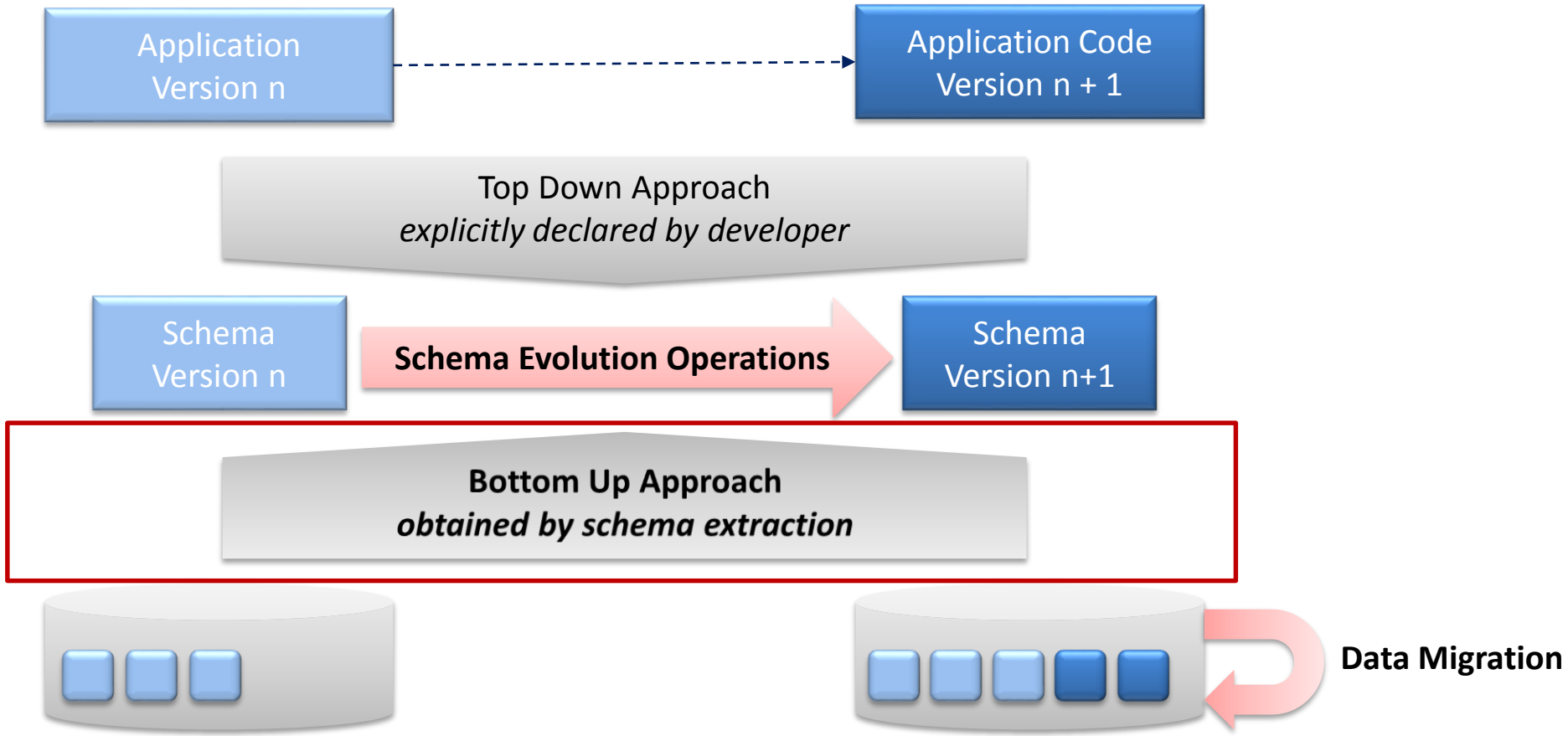
- declaring or extracting an initial schema,
- repeatedly applying schema changes,
- proposing mappings between legacy schema versions,
- and migrating legacy data.

The system is implemented for different types of NoSQL database systems.



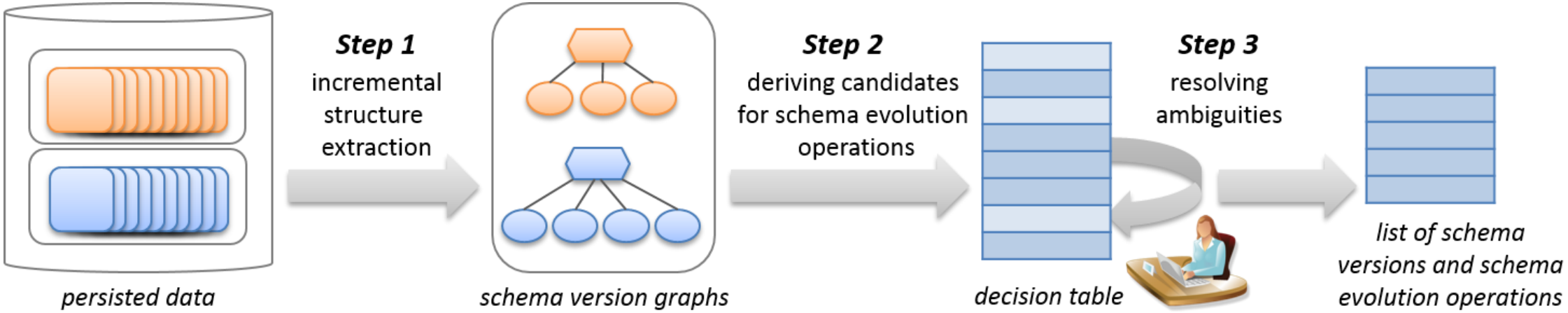
U. Störl, D. Müller, J. Stenzel, A. Tekleab, S. Tolale, M. Klettke, S. Scherzinger. Curating Variational Data in Application Development. Demo@ICDE, Paris, April, 2018.

Vision: Curating Variational Data End-to-End



Schema Extraction Approach

- Process of extracting schema versions and evolution operations by reverse engineering



Introduced in: M. Klettke, H. Awolin, U. Störl, D. Müller, and S. Scherzinger. Uncovering the Evolution History of Data Lakes. Proc. of Scalable Cloud Data Management Workshop (SCDM)@IEEE Big Data 2017, Boston, MA, USA, December 2017.

Running Example from Marine Biology

- JSON datasets for *Species* classification of the Baltic Sea and observation *Protocols*

entity type Species


```
{"id": 123,
 "name": "Mya arenaria",
 "ts": 1}
```

```
{"id": 124,
 "name": "Abra prismatica",
 "ts": 3,
 "category": 141436}
```

```
{"id": 125,
 "name": "Abra alba",
 "ts": 5,
 "WoRMS": 141433}
```

```
{"id": 126,
 "name": "Abra aequalis",
 "ts": 7,
 "WoRMS": 293683}
```


entity type Protocols


```
{"id": 900,
 "time": "2017-07-21",
 "location":
  {"x":19.863281,
   "y":58.487952,
   "z":-1400},
 "spec_id": 123,
 "ts": 2},
```

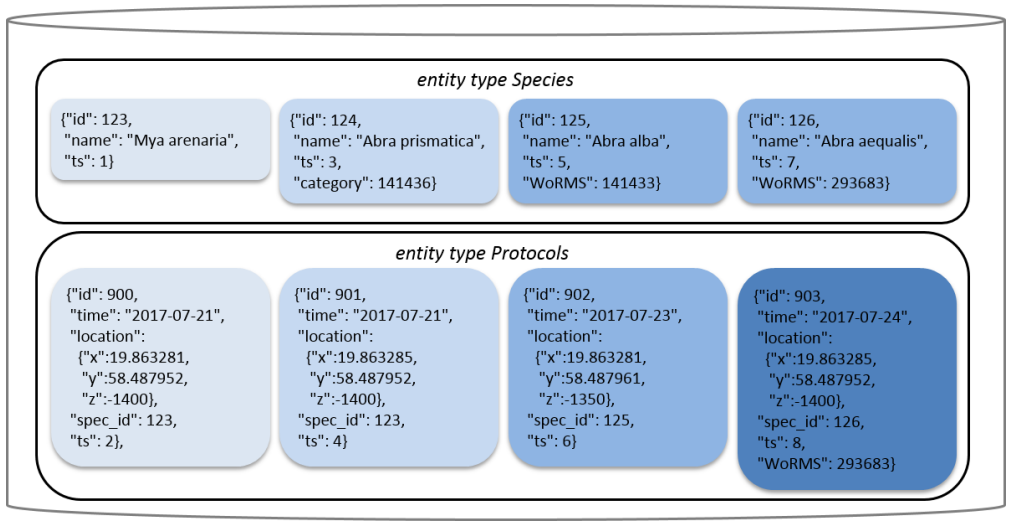
```
{"id": 901,
 "time": "2017-07-21",
 "location":
  {"x":19.863285,
   "y":58.487952,
   "z":-1400},
 "spec_id": 123,
 "ts": 4}
```

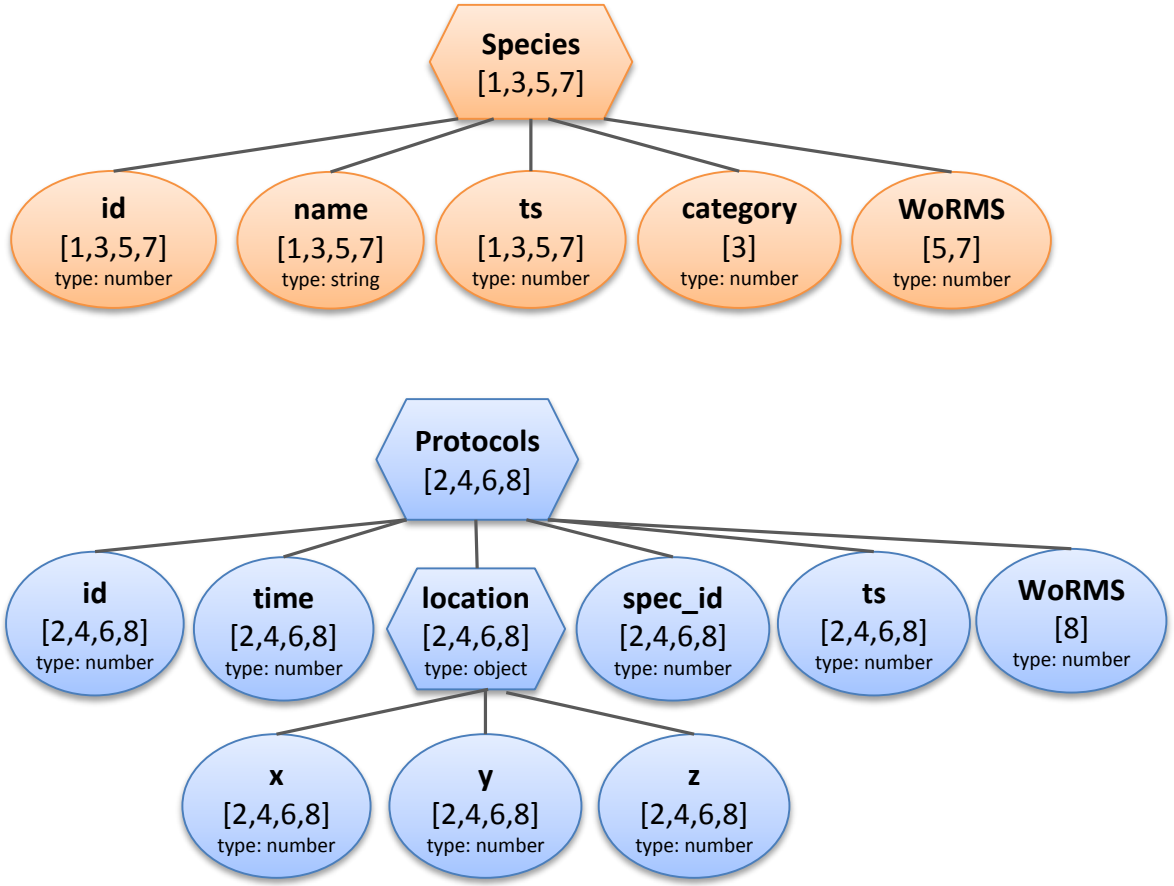
```
{"id": 902,
 "time": "2017-07-23",
 "location":
  {"x":19.863281,
   "y":58.487961,
   "z":-1350},
 "spec_id": 125,
 "ts": 6}
```

```
{"id": 903,
 "time": "2017-07-24",
 "location":
  {"x":19.863285,
   "y":58.487952,
   "z":-1400},
 "spec_id": 126,
 "ts": 8,
 "WoRMS": 293683}
```

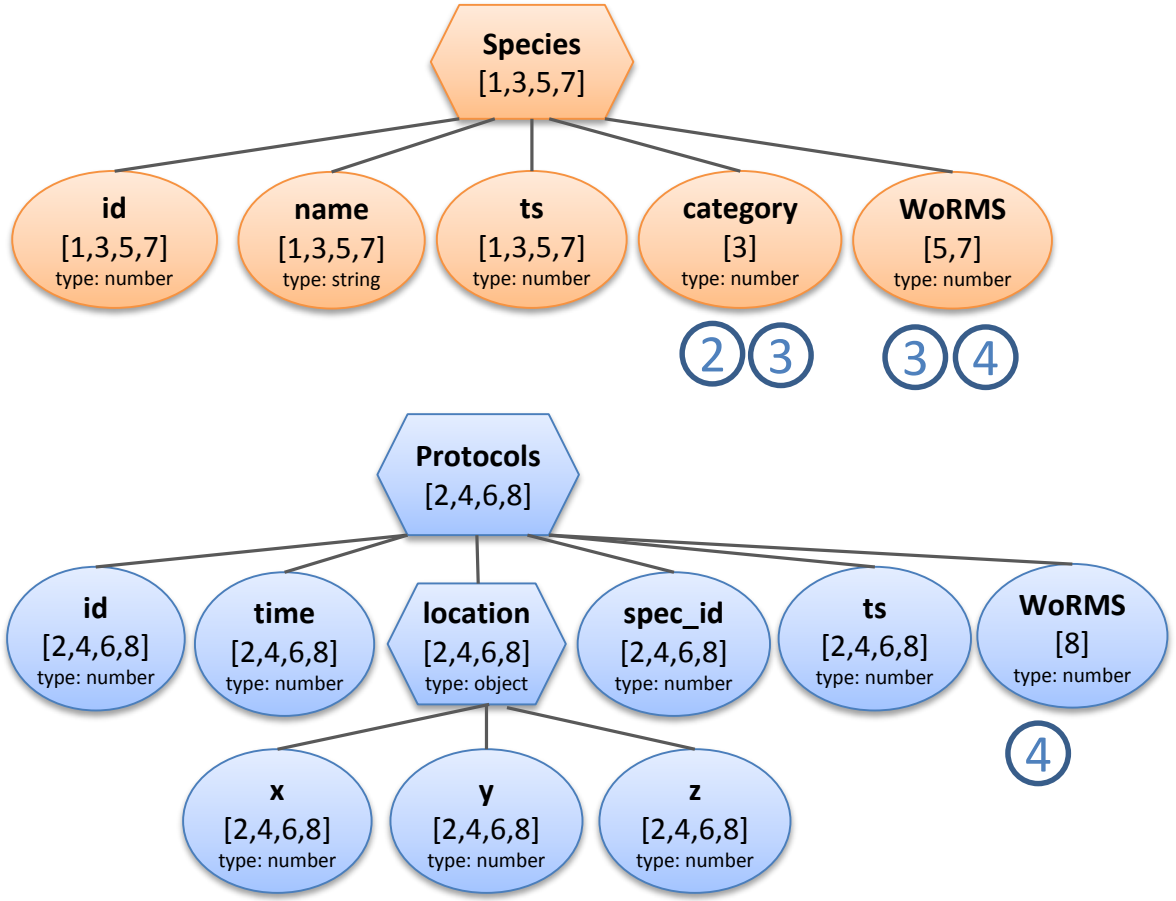

WoRMS: World Register of Marine Species

Step 1: Building the Schema Version Graphs





Step 2: Deriving Schema Evolution Operations



2

```
add integer Species.category
```

3

```
rename Species.category to WoRMS
```

or

```
delete Species.category
add Species.WoRMS
```

4

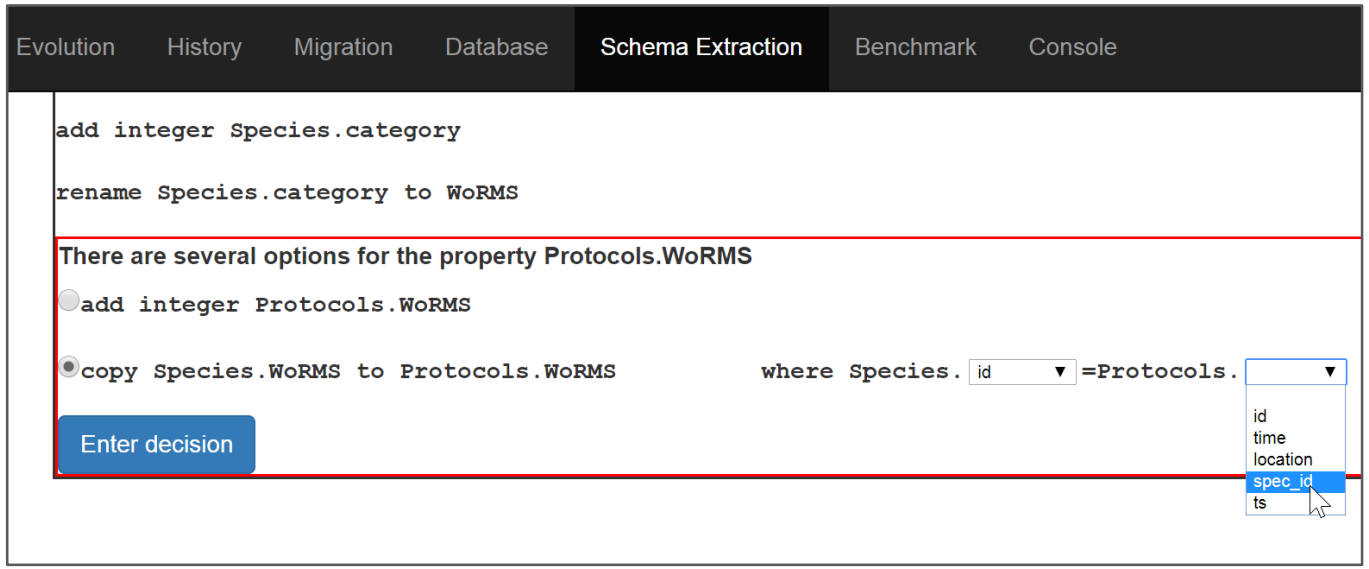
```
add integer Protocols.WoRMS
```

or

```
copy Species.WoRMS to Protocols.WoRMS
where Species.id = Protocols.spec_id
```

Step 3: Resolving Ambiguities

- **Interactively resolving ambiguous schema evolution operations:**
  - alternative schema evolution operations
  - specifying join conditions for move or copy operations

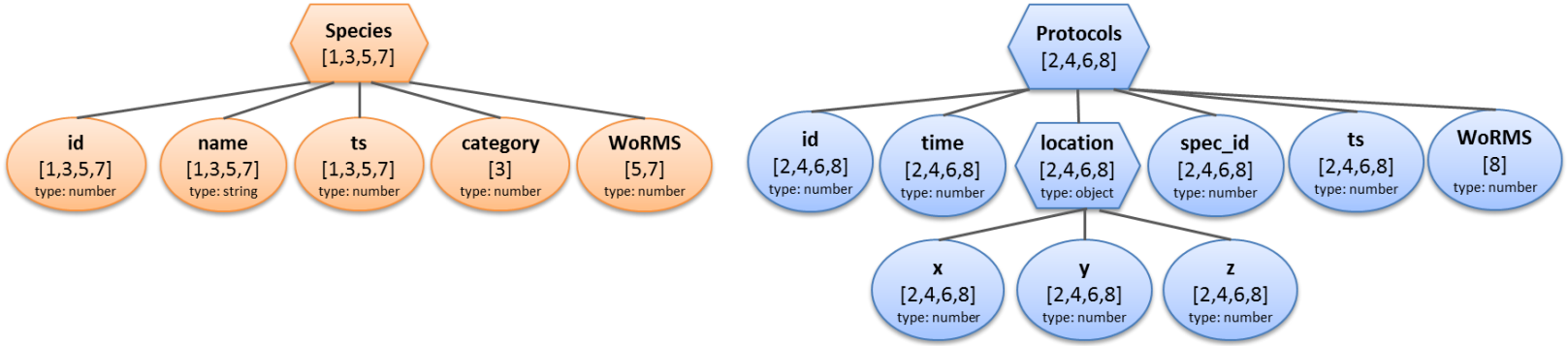


Resolving Ambiguities

- **Open questions**
  - Automated choice in case of ambiguities
  - Suggestion of meaningful join conditions

- **Solution approaches**
  - Algorithm for deriving inclusion dependencies from NoSQL datasets proposed in
    - M. Klettke, H. Awolin, U. Störl, D. Müller, and S. Scherzinger. Uncovering the Evolution History of Data Lakes. Proc. of SCDM 2017@IEEE Big Data 2017, Boston, MA, USA, December 2017.
  - Next steps:
    - Choice of a meaningful join condition for copy and move operations
    - Heuristics for search space reduction

Deriving Schema Versions

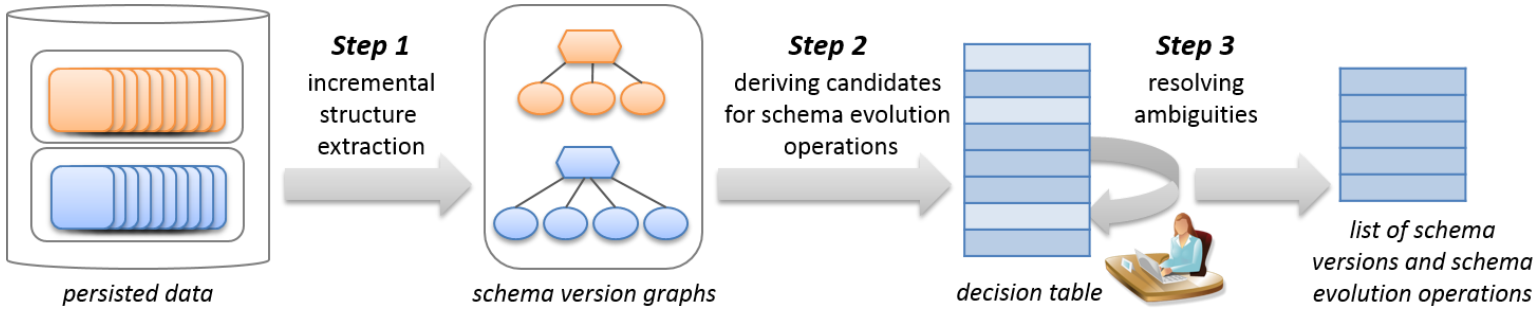


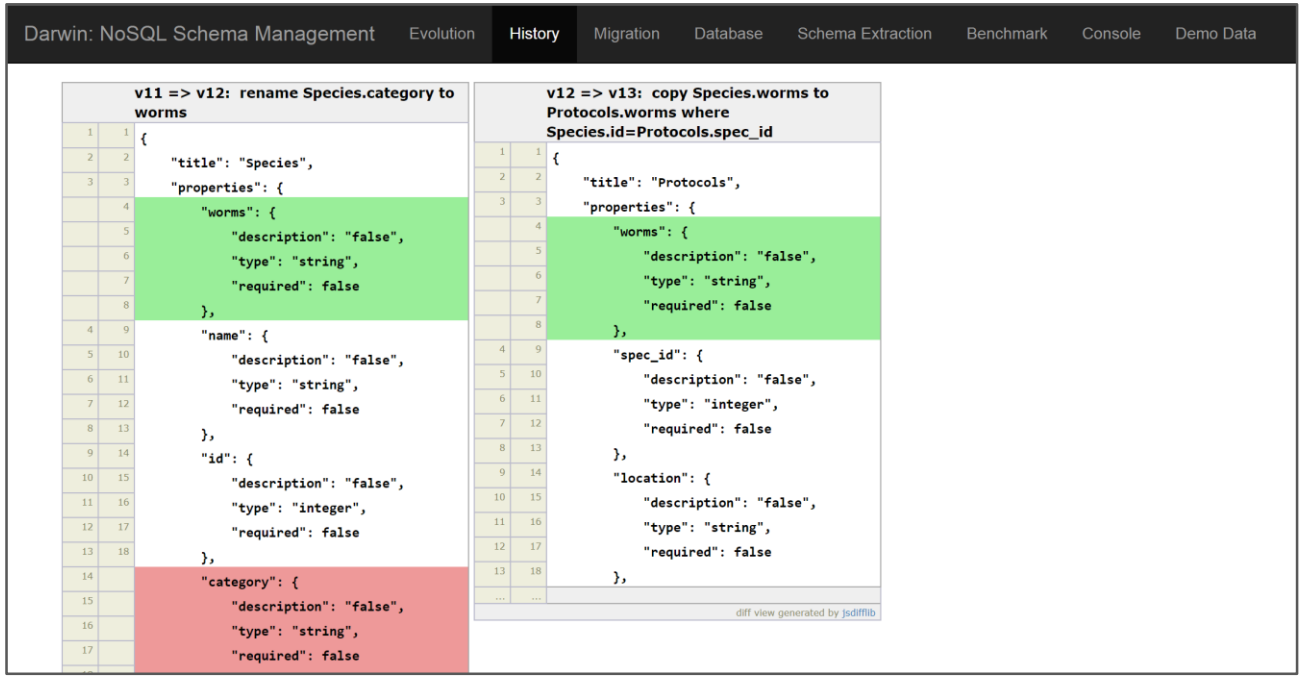

```
copy Species.WoRMS to Protocols.WoRMS
where Species.id = Protocols.spec_id
```

```
{ "title": "Protocols",
"description": "schema version 3",
"type": "object",
"properties": {
  "id": { "type": "number" },
  "location": { "type": "object",
    "properties": { ... } },
  "spec_id": { "type": "number" } }
}
```

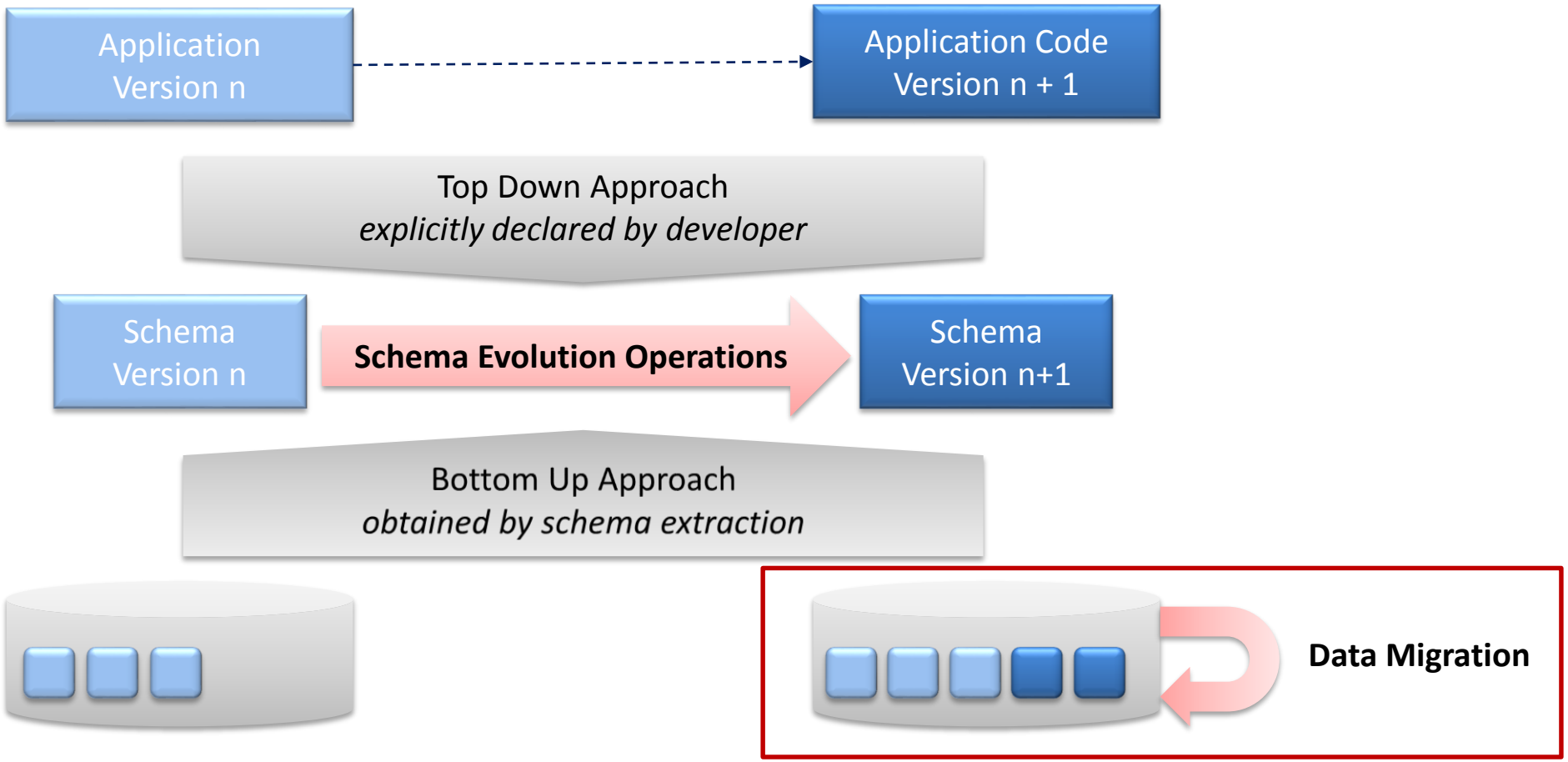
```
{ "title": "Protocols",
"description": "schema version 4",
"type": "object",
"properties": {
  "id": { "type": "number" },
  "location": {"type": "object",
    "properties": { ... } },
  "spec_id": { "type": "number" },
  "WoRMS": { "type": "number" } } }
```

Extracted Schema Evolution Operations and Schema Versions



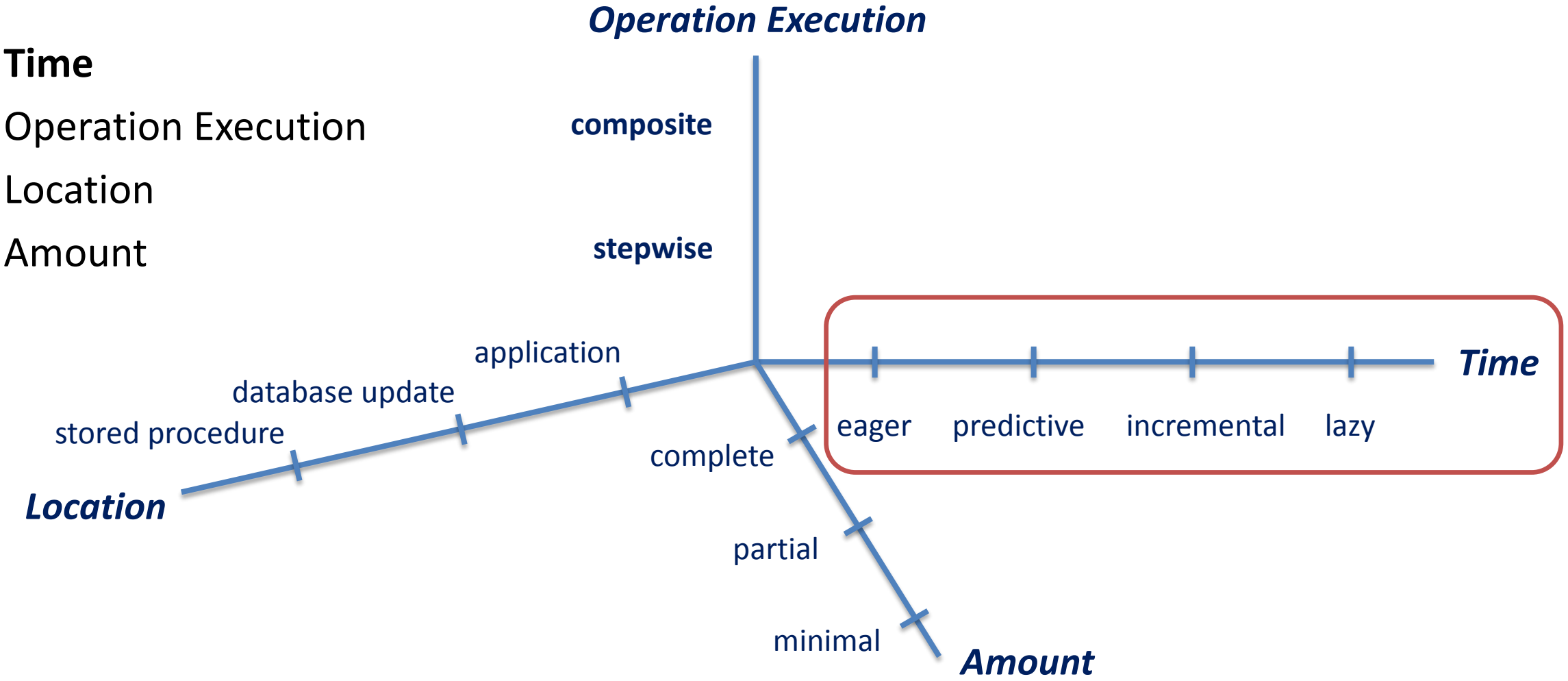


Vision: Curating Variational Data End-to-End



Data Migration – Dimensions

- **Time**
- Operation Execution
- Location
- Amount

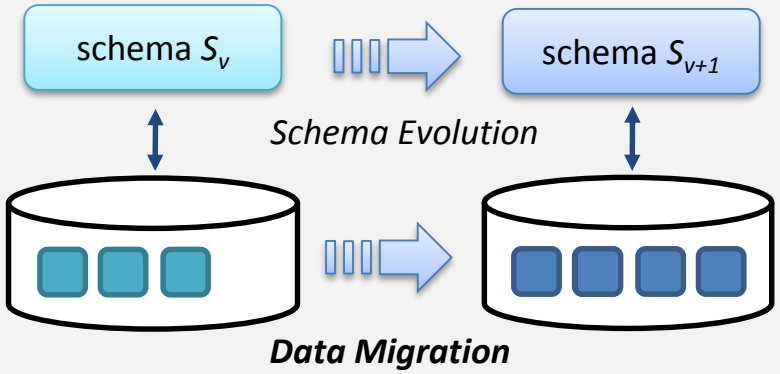


Introduced in: M. Klettke, U. Störl, M. Shenavai, S. Scherzinger: NoSQL Schema Evolution and Big Data Migration at Scale, Proc. of 4th Scalable Cloud Data Management Workshop (SCDM) @ IEEE Big Data Conference, Washington, D.C., 2016

Basic Strategies of Data Migration

Eager Migration

- after introduction of *a new schema version*, all entities are migrated



Advantages:

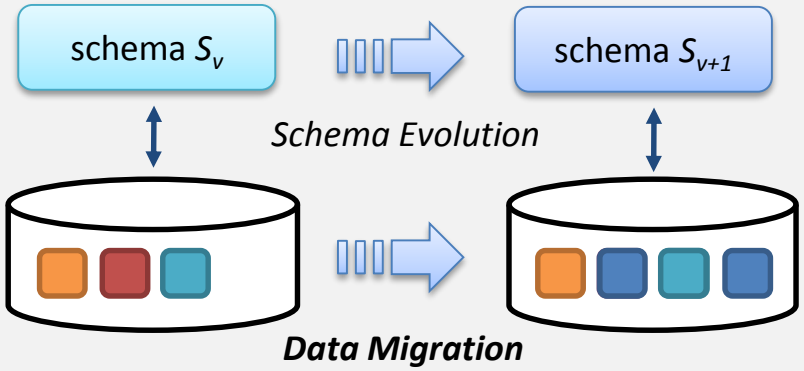
+ all entities are in the current version
+ **low latency** (if entities are accessed)

Disadvantages:

– even entities which are not in use are migrated
– **high number (and costs) of migration operations**

Lazy Migration

- evolution operations are stored,
- data migration is done on request



Advantages:

+ only entities which are in use are migrated
+ **no unnecessary** data migration **operations**
+ **composition of operations** is possible

Disadvantages:

– entities in the NoSQL database in different versions
– increased **latency**

How to reduce Costs of Data Migration?

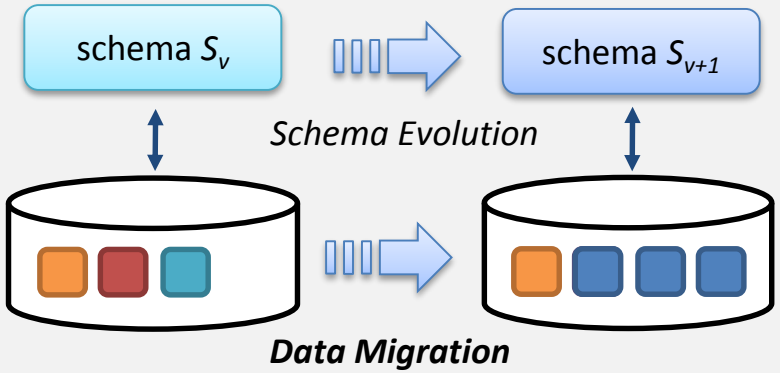
- In case of **large amount of datasets**
  - (system downtime)
  - update operations even for cold data (which are not in use)
- In case of **database as a service**
  - payable operations, monetary costs for all data migrations

- **How to reduce costs of data migration?**
  - ⇨**Optimize Data Migration**
  - ⇨**Hybrid Migration Approaches**
    - Predictive Migration
    - Incremental Migration

Hybrid Migration Strategies

Predictive Migration

- Forecast function, which entities are accessed in near future (bases on heuristics)
- Predictive migration of these entities



Advantages:

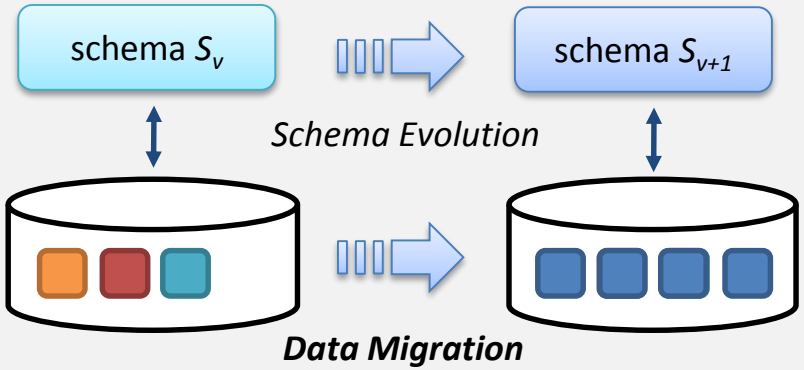
+ **decreased average latency**
+ **reduced** number of migration **operations**

Disadvantage:

– additional migration operations in case of wrong predictions

Incremental Migration

- in some version, an eager migration is applied



Advantage:

+ **composition of operations** is possible

Disadvantage:

– even entities which are not in use are migrated

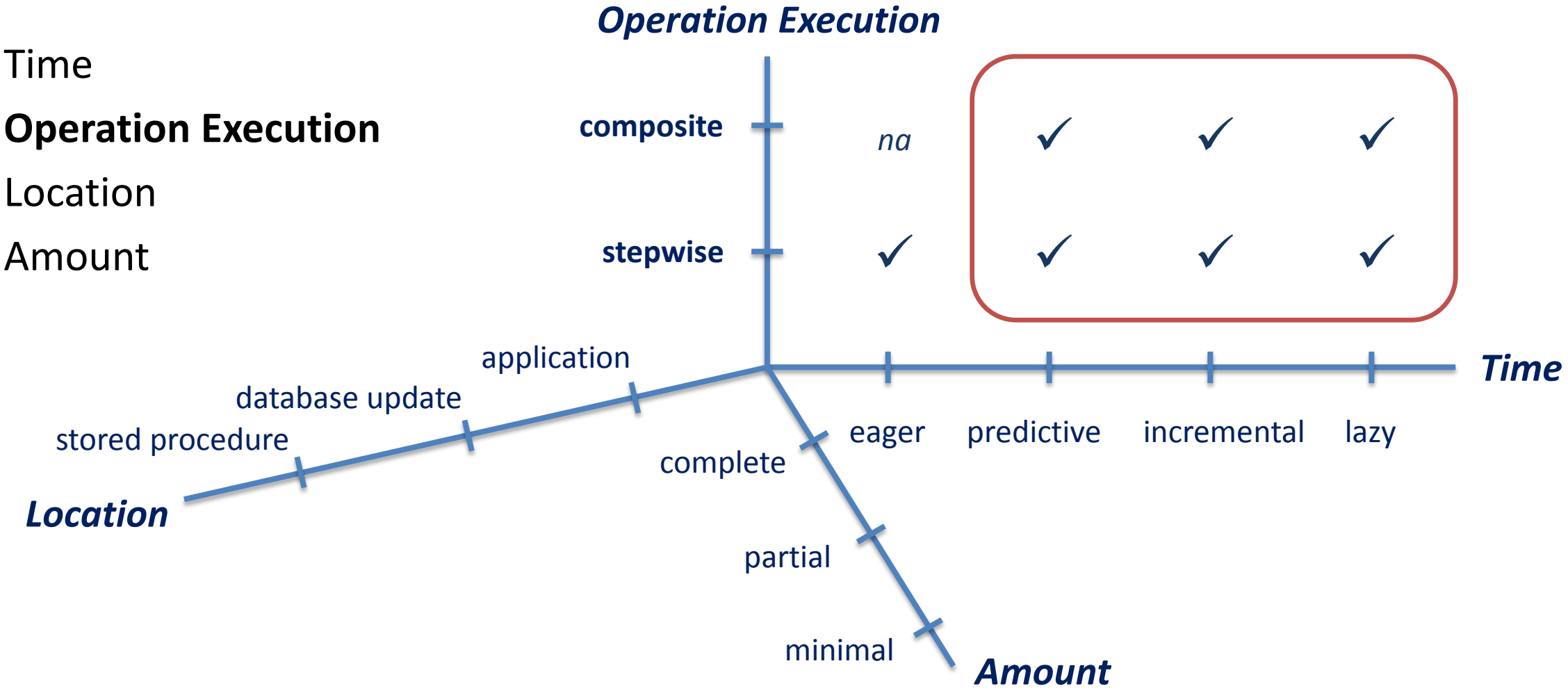
Data Migration: Comparison of different Approaches

	Eager Migration	Predictive Migration	Incremental Migration	Lazy Migration	*Versioning**
• Number of migration operations	**high**	**medium**	**medium**	**low**	***none***
• Average latency	**none**	**low**	**low**	**medium**	***high***
• Application downtime	**high**	**medium**	**medium**	**none**	***none***
• Effort for query rewriting	**none**	**medium**	**medium**	**medium**	***high***

**Versioning*: Does not migrate data – use query rewriting instead

Data Migration – Dimensions

- Time
- **Operation Execution**
- Location
- Amount



Introduced in: M. Klettke, U. Störl, M. Shenavai, S. Scherzinger: NoSQL Schema Evolution and Big Data Migration at Scale, 4th Scalable Cloud Data Management Workshop (SCDM) @ IEEE Big Data Conference, Washington, D.C., 2016

Optimization of Data Migration: Composite Migration

Theory: If entities are migrated from older versions

⇨ **Composition of evolution operations** is possible in some cases:

op1 \ op2	**rename** $E_B.y$ **to** z	**copy** $E_B.y$ **to** $E_C.z$ $cond_2$	**move** $E_B.y$ **to** $E_C.z$ $cond_2$	**delete** $E_B.y$
add $E_B.y$ = *default*	**add** $E_B.z$ = *default*	-	**add** $E_C.z$ = *default* $cond_2$ **add** $E_B.y$ = *default* $\neg cond_2$	ϵ (noop)
rename $E_B.x$ **to** y	**rename** $E_B.x$ **to** z	**copy** $E_B.x$ **to** $E_C.z$ $cond_2$	**move** $E_B.x$ **to** $E_C.z$ $cond_2$ **rename** $E_B.x$ **to** y $\neg cond_2$	**delete** $E_B.x$
copy $E_A.x$ **to** $E_B.y$ $cond_1$	**copy** $E_A.x$ **to** $E_B.z$ $cond_1$	-	**copy** $E_A.x$ **to** $E_C.z$ $cond_1 \wedge cond_2$ **copy** $E_A.x$ **to** $E_B.y$ $cond_1 \wedge \neg cond_2$	ϵ (noop)
copy $E_B.y$ **to** $E_D.u$ $cond_3$	-	-	-	**move** $E_B.y$ **to** $E_D.u$ $cond_3$
move $E_A.x$ **to** $E_B.y$ $cond_1$	**move** $E_A.x$ **to** $E_B.z$ $cond_1$	-	**move** $E_A.x$ **to** $E_C.z$ $cond_1 \wedge cond_2$ **move** $E_A.x$ **to** $E_B.y$ $cond_1 \wedge \neg cond_2$	**delete** $E_A.y$

Example:

```
copy A.y to B.y
   where A.id = B.aid
delete A.y
```

⇨

```
move A.y to B.y
where A.id = B.aid
```

Advantages

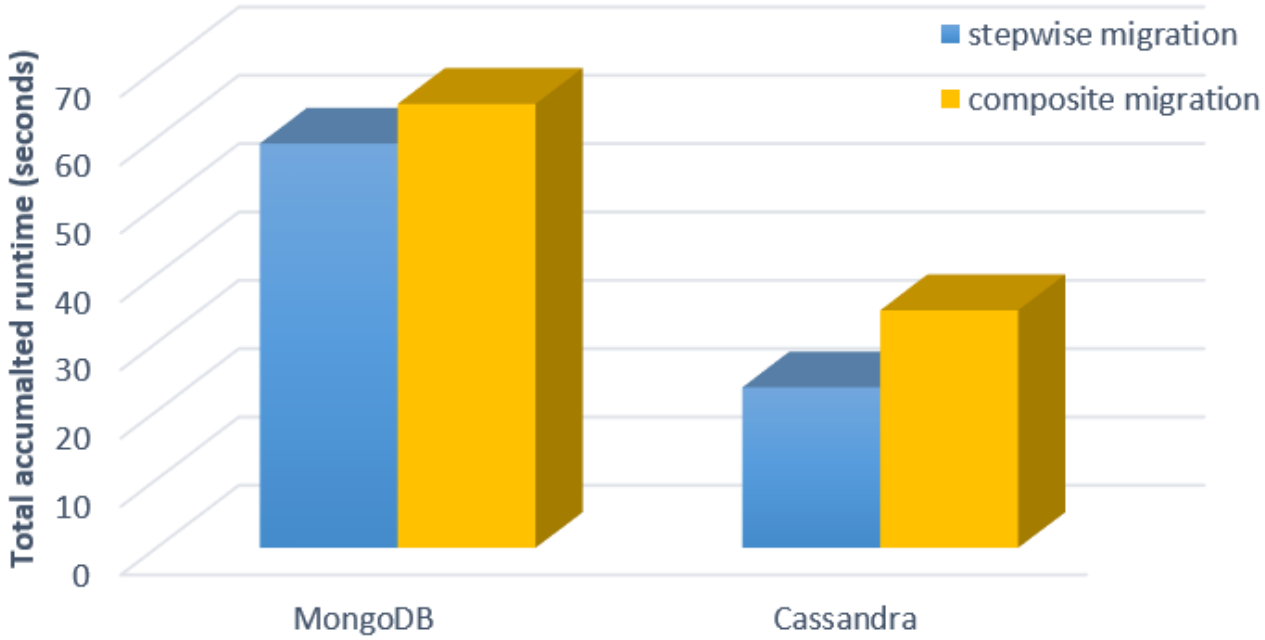
- ⇨ Smaller number of data migration operations
- ⇨ **Lower migration costs**

Introduced in: M. Klettke, U. Störl, M. Shenavai, S. Scherzinger: NoSQL Schema Evolution and Big Data Migration at Scale, 4th Scalable Cloud Data Management Workshop (SCDM) @ IEEE Big Data Conference, Washington, D.C., 2016

Optimization of Data Migration: Composite Migration

Practice

First (surprising) results:



Total accumulated runtime for applying five pending schema changes to 10,000 entities, realized as stepwise or composite migration

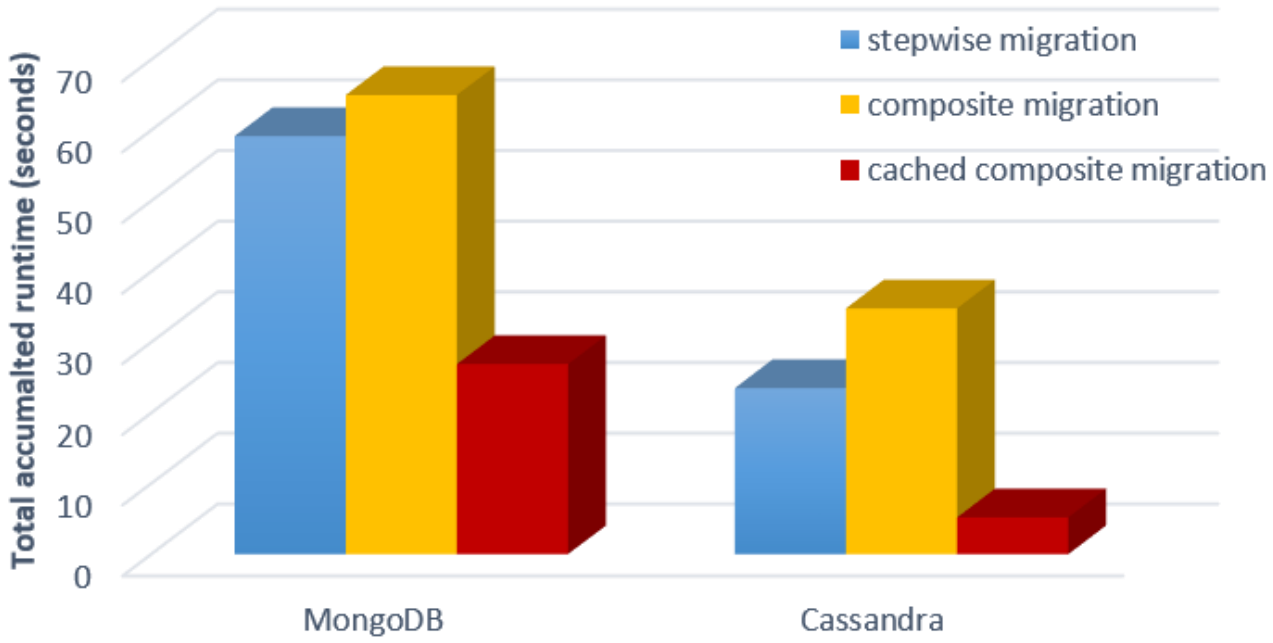
- Surprisingly, composite migration is not immediately faster.
- Closer analysis reveals that it is the **overhead of computing the compositions repeatedly**, which falsifies our hypothesis.

U. Störl, A. Tekleab, M. Klettke, S. Scherzinger. In for a Surprise when Migrating NoSQL Data. Lightning Talk@ICDE, Paris, April, 2018.

Optimization of Data Migration: Composite Migration

Practice

After introducing a cache
for computed composition formulae:



Total accumulated runtime for applying five pending schema changes to 10,000 entities, realized as stepwise or composite migration

- **By introducing a cache and storing the computed composition formulae**, we can effectively reduce the runtime when compared to the stepwise approach.
- **We achieve a reduction by about 50% for MongoDB, and by nearly 80% for Cassandra.**

U. Störl, A. Tekleab, M. Klettke, S. Scherzinger. In for a Surprise when Migrating NoSQL Data. Lightning Talk@ICDE, Paris, April, 2018.

NoSQL Schema Evolution and Big Data Migration at Scale

Conclusion

- **Objective: Support for schema-management end-to-end including**
  - **Schema evolution management**
    - Schema evolution language
    - Schema extraction
  - **Data migration** as safe process based on schema evolution management
    - Data migrations strategies may be classified and optimized by 4 dimensions
    - Lessons learned:
      - In architecting a scalable data migration tool for NoSQL data stores, careful engineering is called for.
      - In particular, a thorough experimental analysis is required.

NoSQL Schema Evolution and Big Data Migration at Scale

Future Work

- Investigate optimizations so that **data migration** can be implemented **at scale**
- Determine the major cost factors for the migration strategies to compile from them a generic **cost model**
- Craft a dedicated NoSQL **migration benchmark**

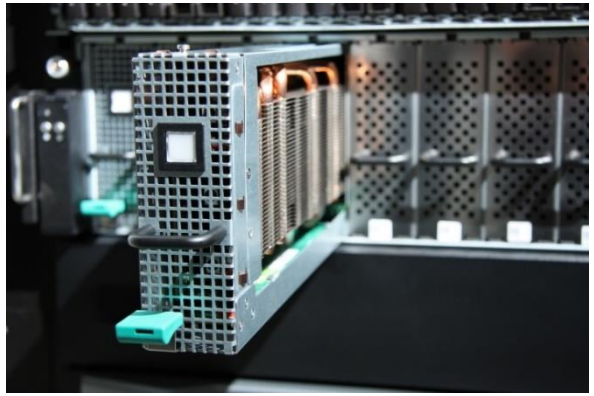
⇨ Design and develop an **automated data migration advisor** that supports software development teams in managing schema evolution in NoSQL data stores

Acknowledgement
This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant #385808805, since December 2017.

Backup

Supporting Schema Management and Data Migration with *Darwin*: Measurement Environment

- **Big Data Cluster @ Darmstadt University of Applied Sciences**
  - 40 Nodes
    - 64 GB / 128 GB RAM; 4 TB storage each
  - MongoDB / Couchbase / Cassandra / …
  - Apache Hadoop
  - Apache Spark
  - Big Data Competence Centre: http://fbi.h-da.de/bdcc



NoSQL Schema Management and Data Migration: Selected Publications

- U. Störl, D. Müller, J. Stenzel, A. Tekleab, S. Tolale, M. Klettke, S. Scherzinger. Curating Variational Data in Application Development. Demo@ 34th IEEE International Conference on Data Engineering (ICDE), Paris, April, 2018.
- U. Störl, A. Tekleab, M. Klettke, S. Scherzinger. In for a Surprise when Migrating NoSQL Data. Lightning Talk @ 34th IEEE International Conference on Data Engineering (ICDE), Paris, April, 2018.
- M. Klettke, H. Awolin, U. Störl, D. Müller, and S. Scherzinger. Uncovering the Evolution History of Data Lakes. Proc. of SCDM 2017@IEEE Big Data 2017, Boston, MA, USA, December 2017.
- U. Störl, D. Müller, M. Klettke, and S. Scherzinger. Enabling Efficient Agile Software Development of NoSQL-backed Applications. Demo@BTW 2017, Stuttgart, March 2017
- M. Klettke, U. Störl, S. Scherzinger. Schema Evolution in Scalable Cloud Data Management. SCDM 2017@BTW 2017, Stuttgart, March 2017 (Keynote)
- M. Klettke, U. Störl, M. Shenavai, and S. Scherzinger. NoSQL Schema Evolution and Big Data Migration at Scale. Proc. of SCDM 2016@IEEE Big Data 2016, Washington D.C., USA, December 2016.
- M. Klettke, U. Störl, S. Scherzinger, S. Sombach, and K. Wiech. Challenges with Continuous Deployment of NoSQL-backed Database Applications. Proc. of FG-DB 2016@LWDA 2016, September 2016.
- S. Scherzinger, M. Klettke, and U. Störl. A Datalog-based Protocol for Lazy Data Migration in Agile NoSQL Application Development. Proc. of DBPL 2015 @ SPLASH 2015 (Systems, Programming, Languages and Applications: Software for Humanity), Pittsburgh, Pennsylvania, United States, October 2015
- U. Störl, T. Hauf, M. Klettke und S. Scherzinger: Schemaless NoSQL Data Stores – Object-NoSQL Mappers to the Rescue? BTW 2015, Hamburg, March 2015
- M. Klettke, U. Störl und S. Scherzinger: Schema Extraction and Structural Outlier Detection for NoSQL Data Stores.BTW 2015, Hamburg, March 2015
- S. Scherzinger, M. Klettke und U. Störl: Cleager: Eager Schema Evolution in NoSQL Document Stores. Demo@BTW 2015, Hamburg, March 2015
- M. Klettke, S. Scherzinger und U. Störl: Datenbanken ohne Schema? Herausforderungen und Lösungs-Strategien in der agilen Anwendungsentwicklung mit schema-flexiblen NoSQL-Datenbanksystemen. Datenbank-Spektrum, 14(2): 119-129, July 2014
- S. Scherzinger, M. Klettke, and U. Störl. Managing Schema Evolution in NoSQL Data Stores. Proc. of the 14th International Symposium on Database Programming Languages (DBPL 2013)@VLDB 2013, Riva del Garda, Trento, Italy, August 2013