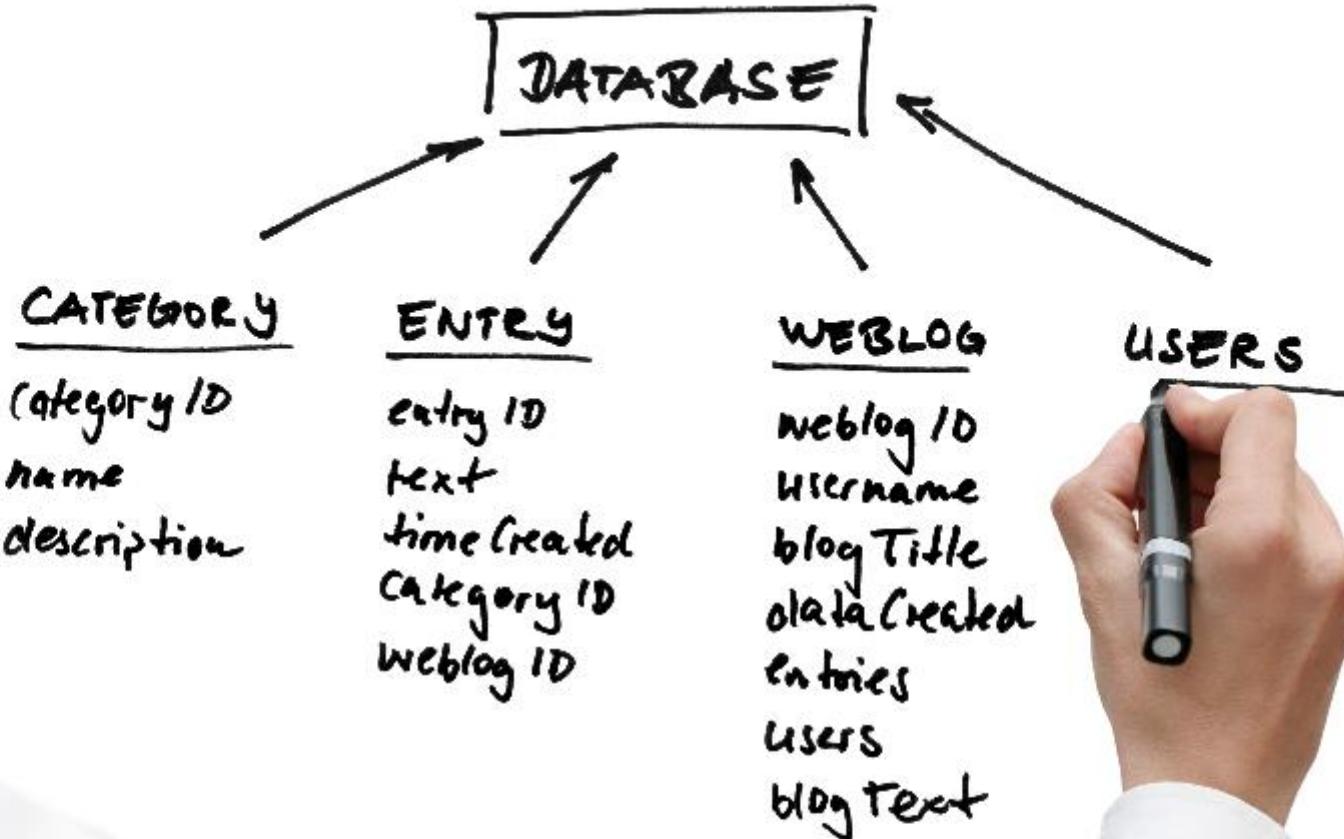


Understanding Schema Evolution in Schema-less NoSQL Data Stores

Loup Meurice and Anthony Cleve

Faculty of Informatics
University of Namur
Belgium

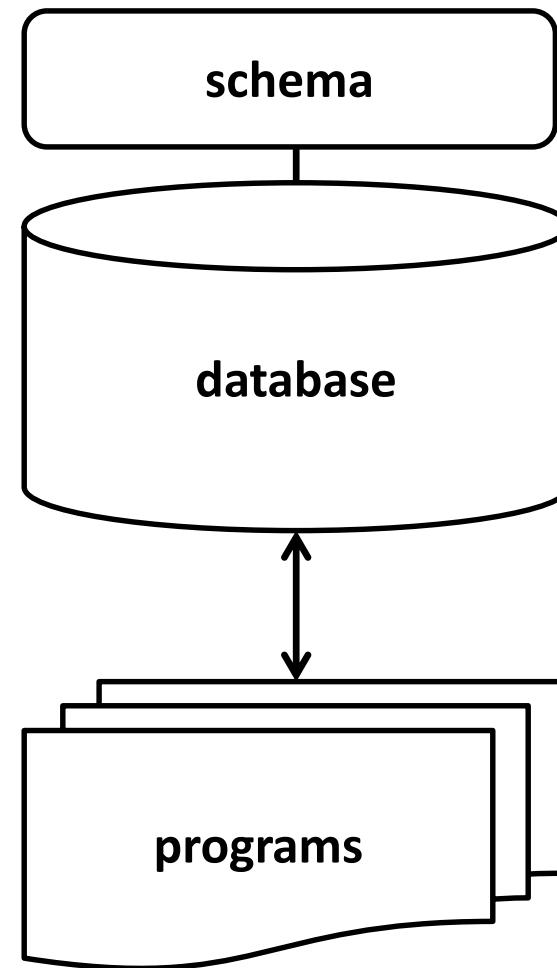




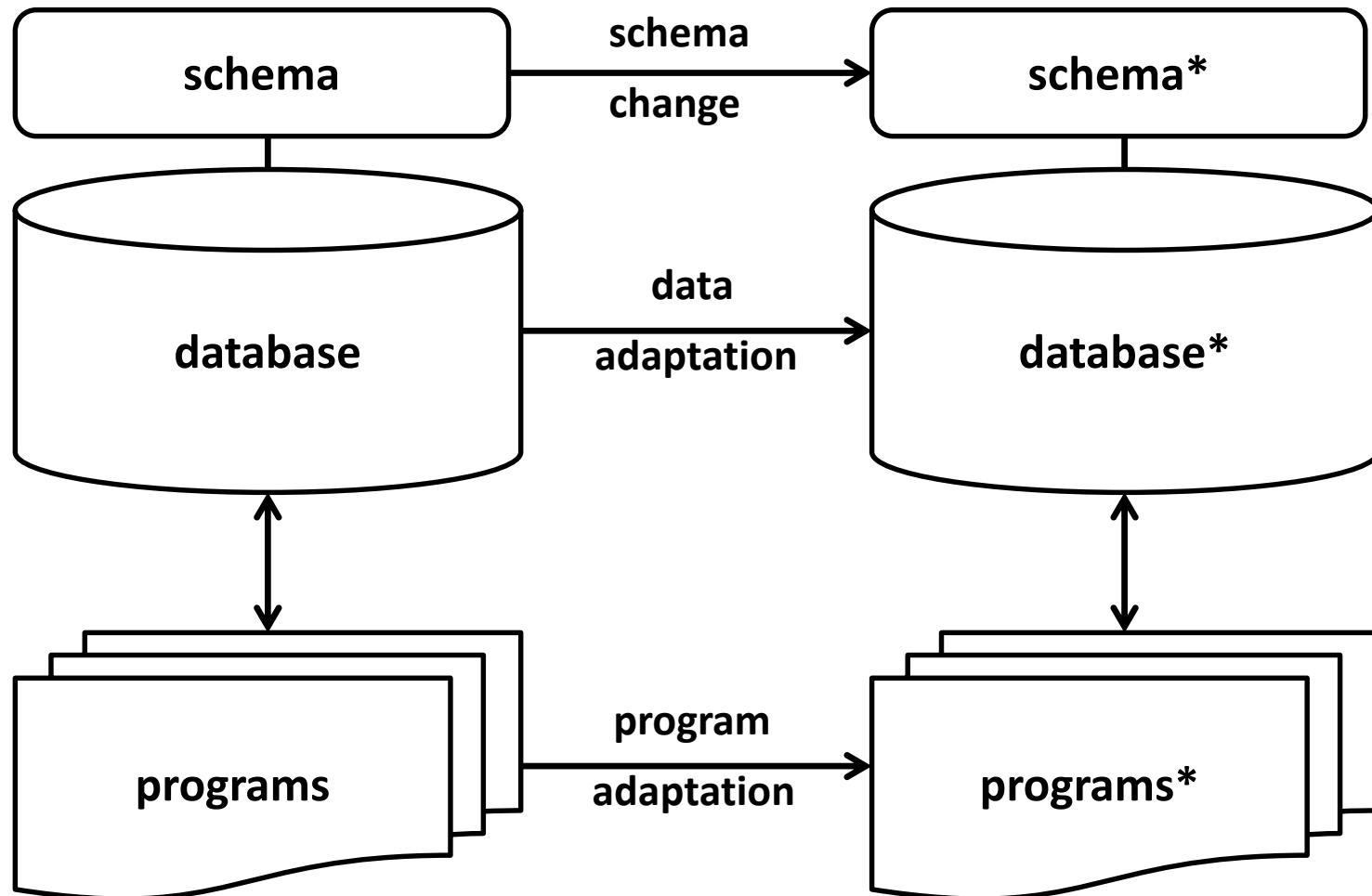
Part I

Schema-less NoSQL Database Evolution

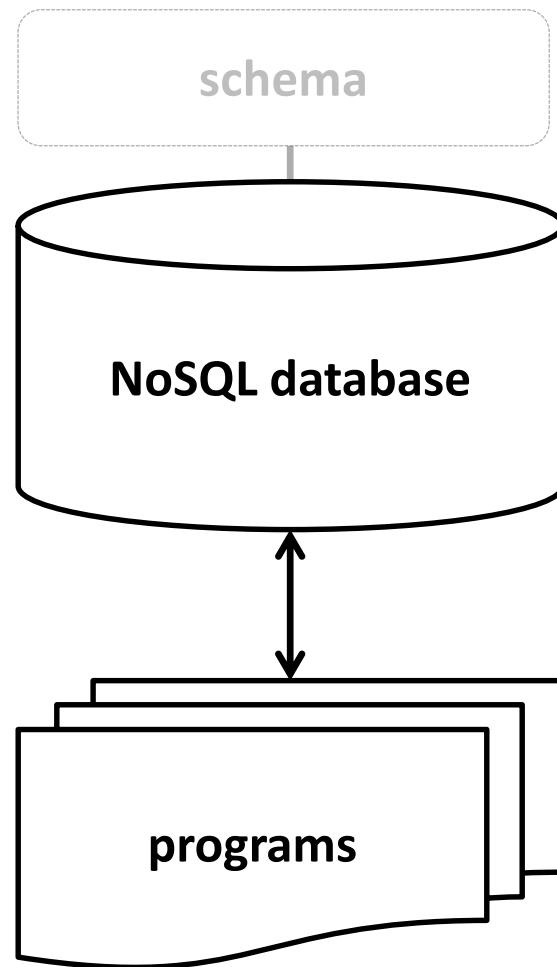
Relational Database



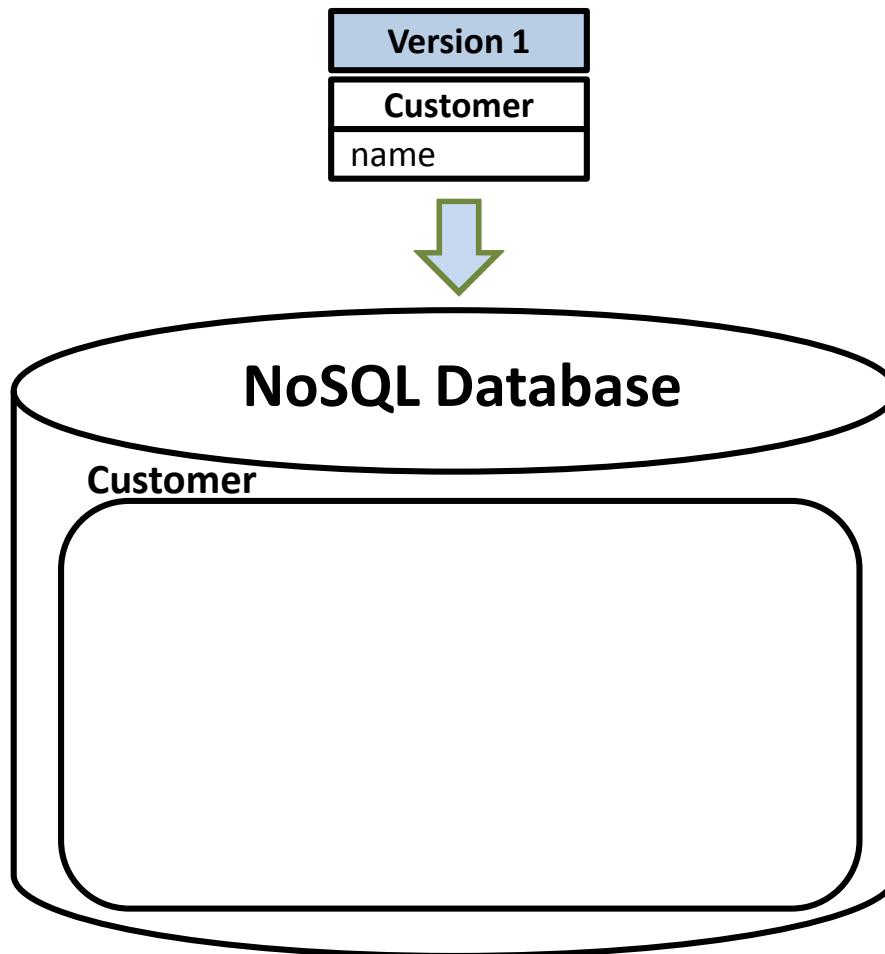
Relational Database



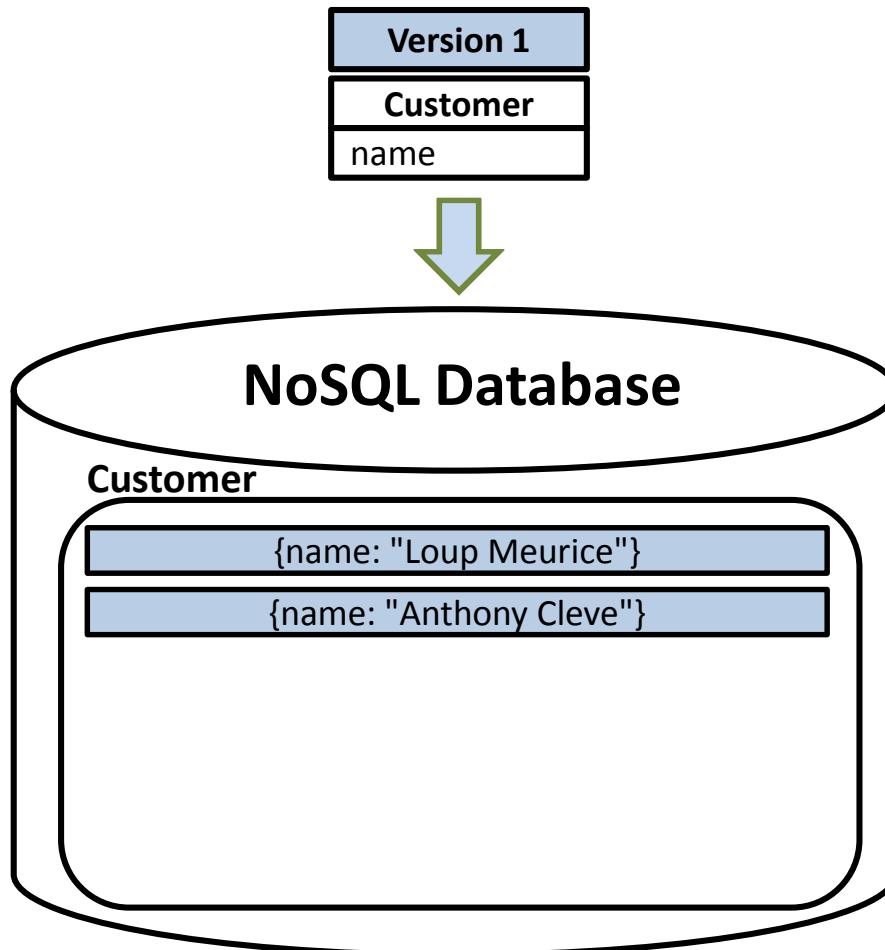
Schema-less NoSQL Database



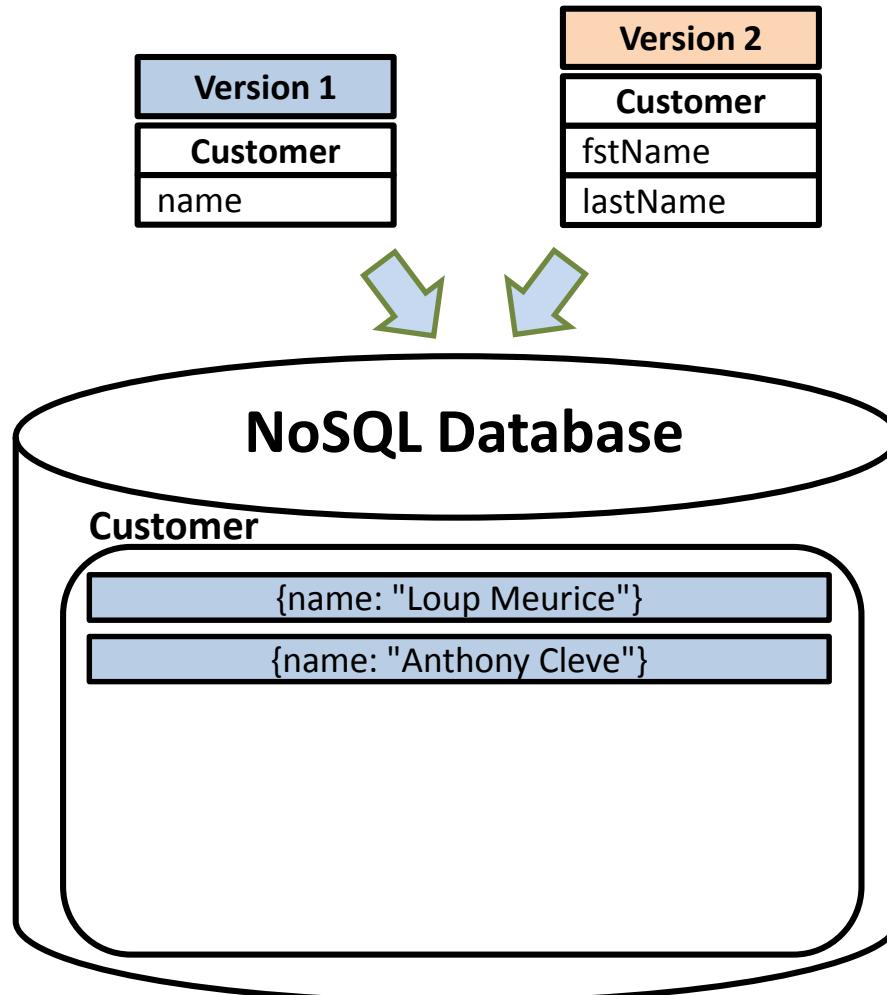
Schema-less NoSQL Database



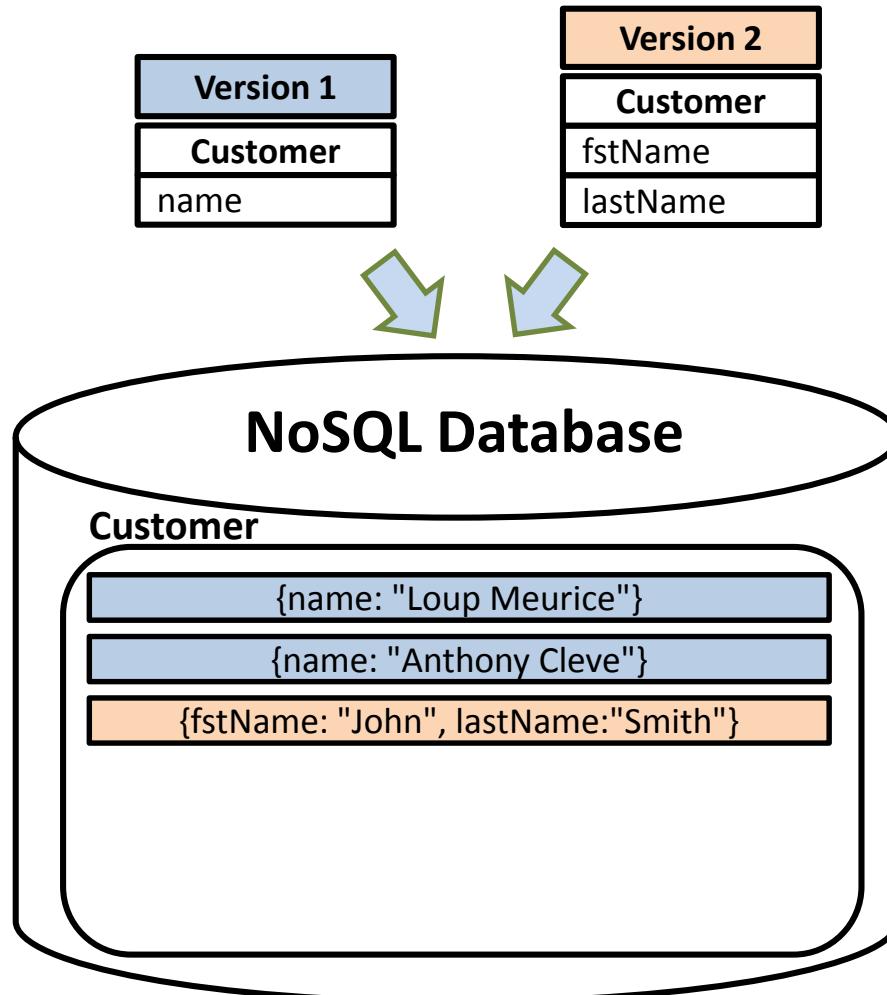
Schema-less NoSQL Database



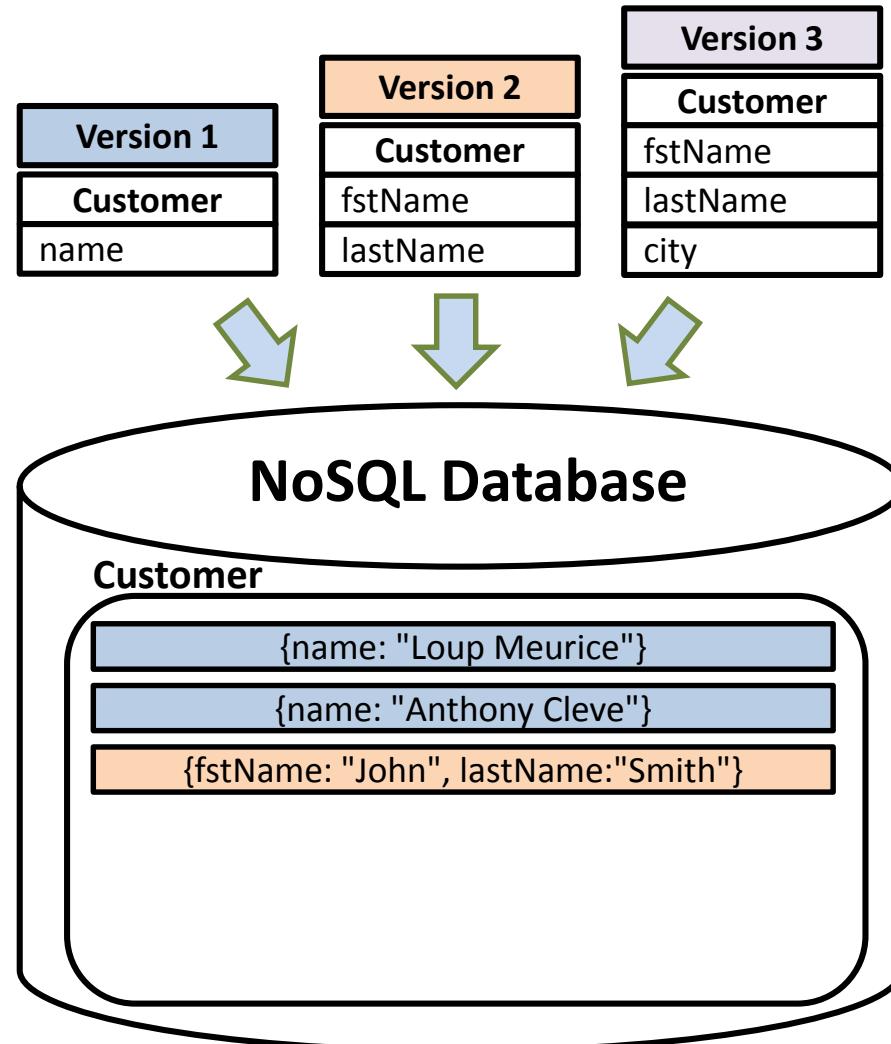
Schema-less NoSQL Database



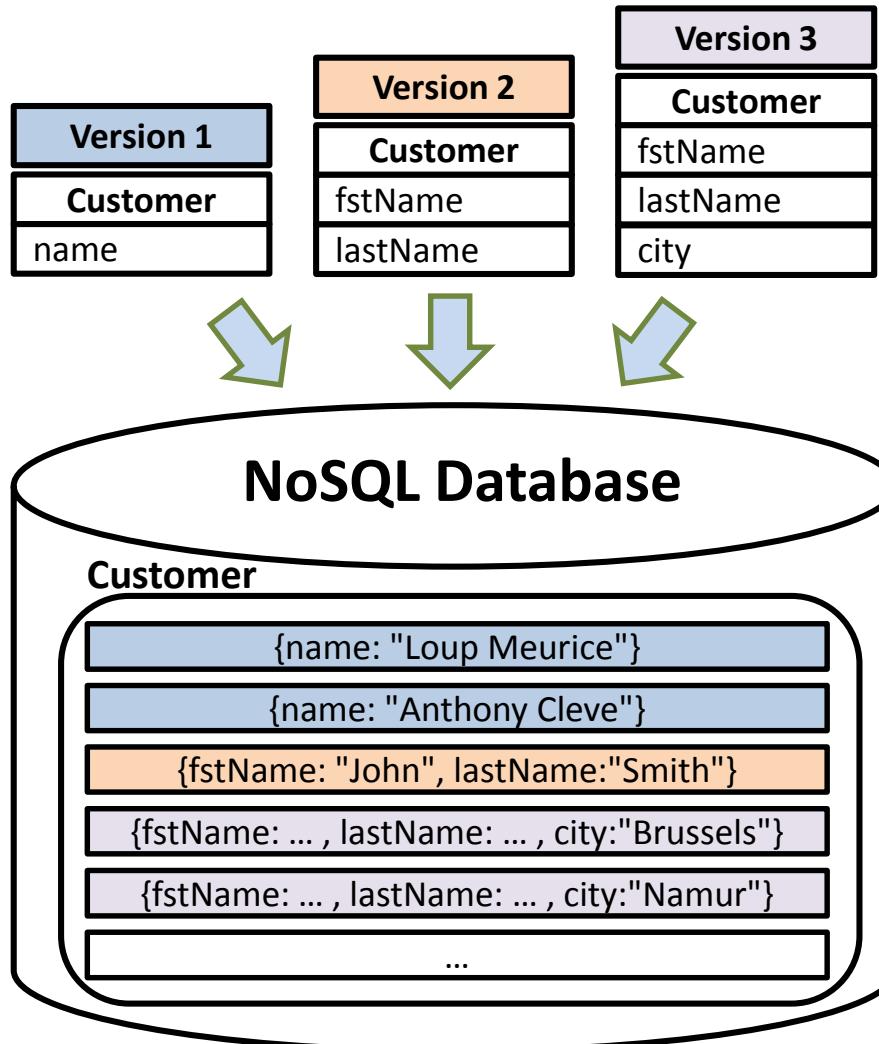
Schema-less NoSQL Database



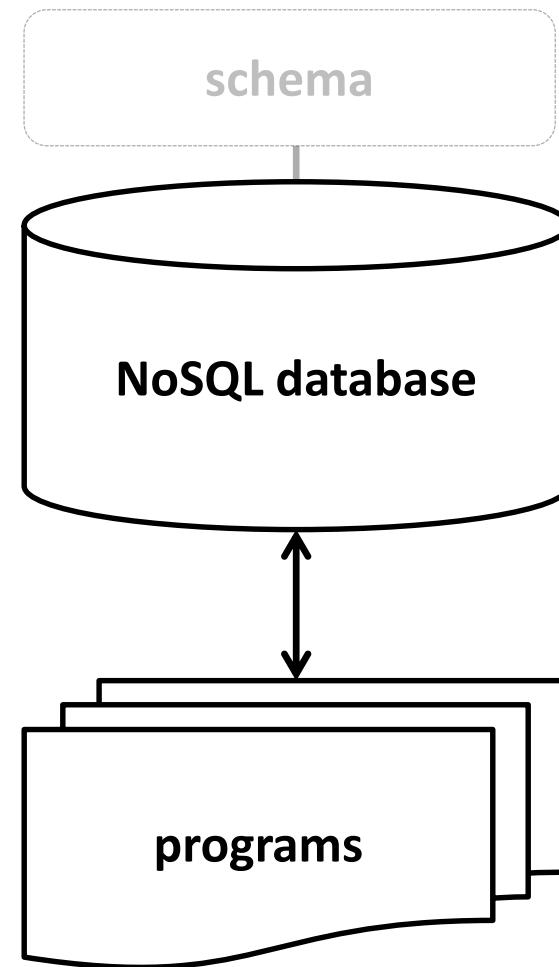
Schema-less NoSQL Database



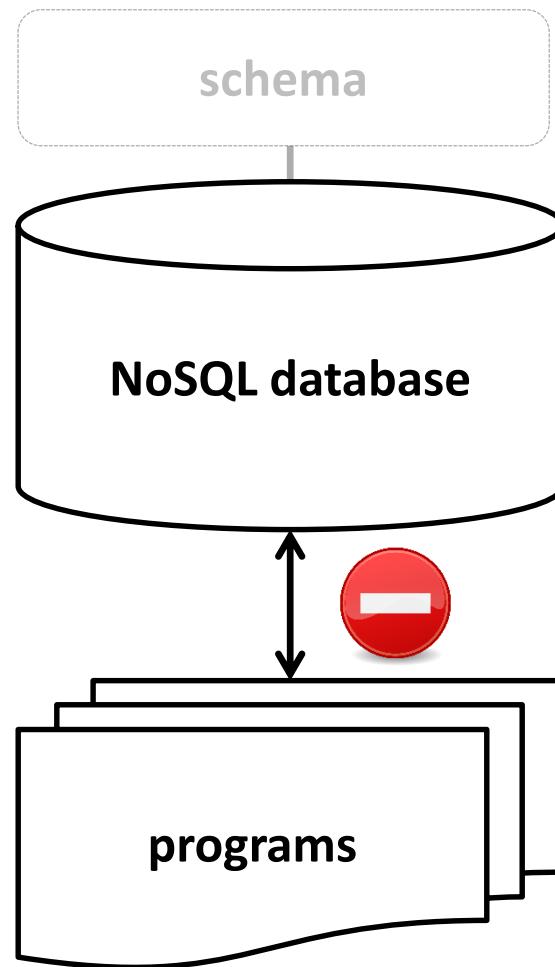
Schema-less NoSQL Database



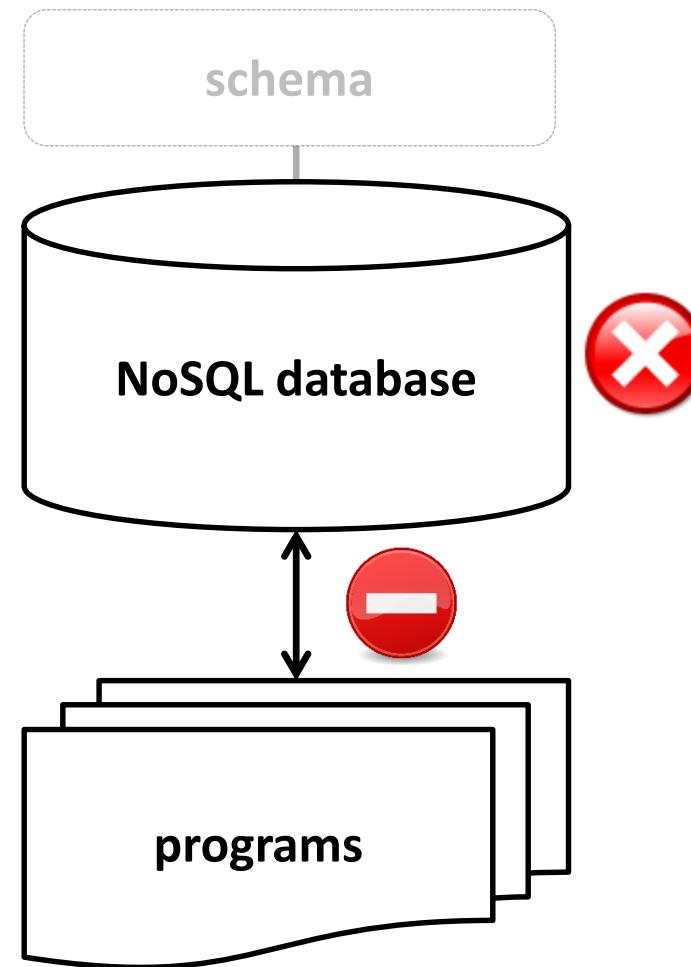
Schema-less NoSQL Database



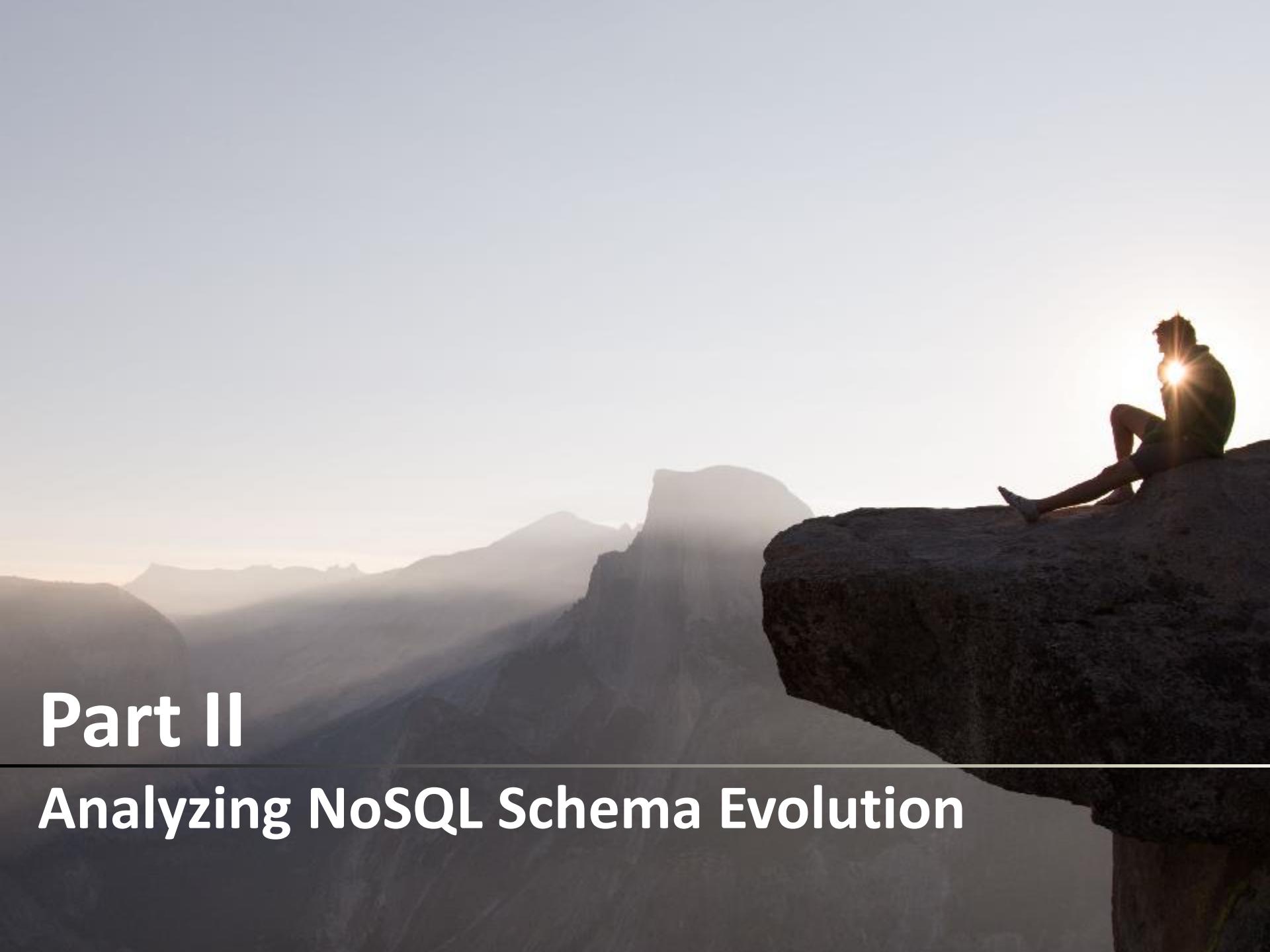
Schema-less NoSQL Database



Schema-less NoSQL Database





The background of the slide features a scenic landscape of mountains at sunset or sunrise. A person is sitting on a large, dark rock on the right side, their back to the viewer, looking out over the mountains. The sky is a soft, warm color, and the mountains are silhouetted against it.

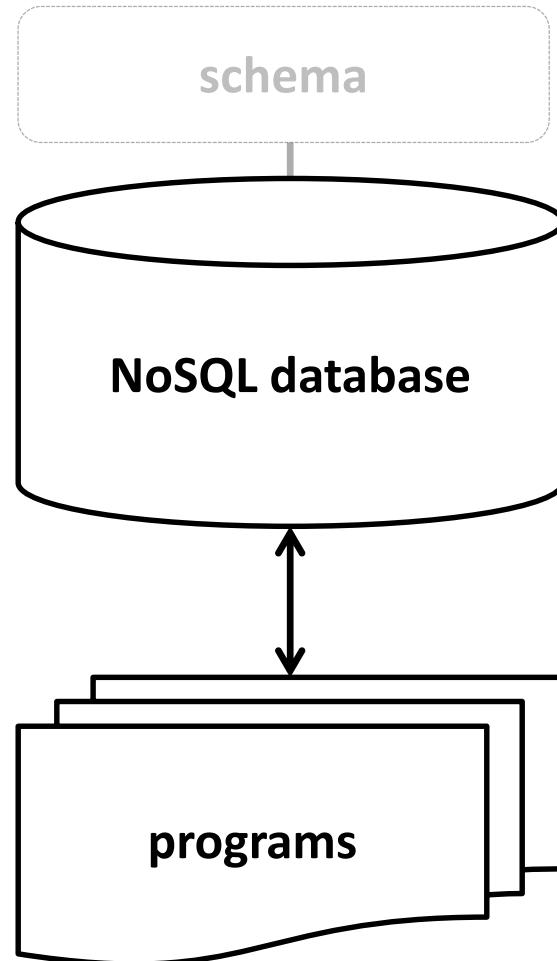
Part II

Analyzing NoSQL Schema Evolution

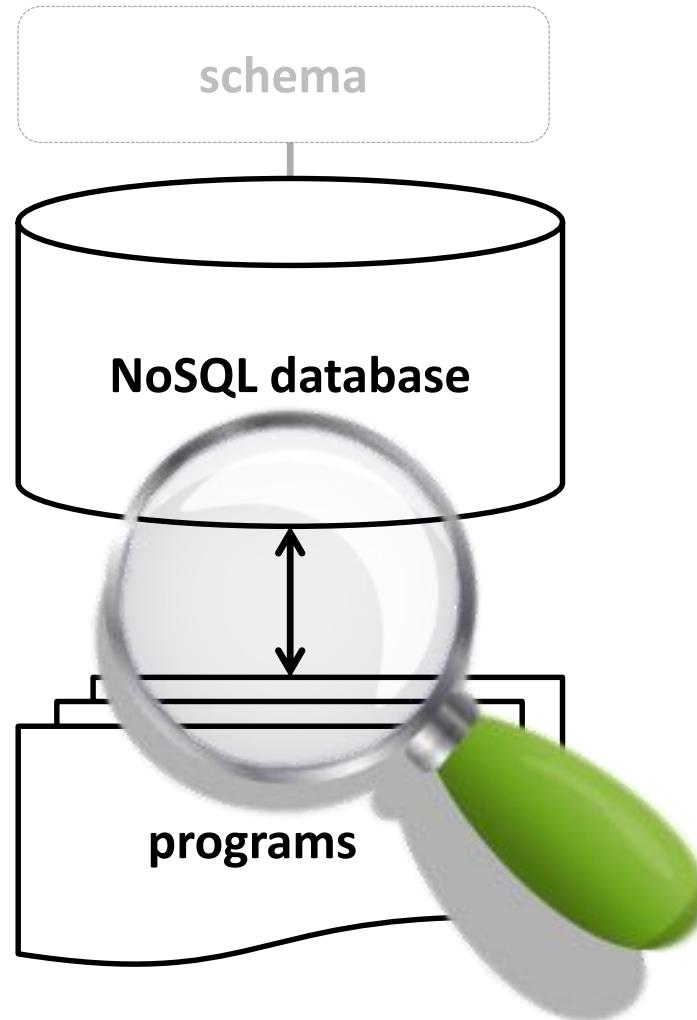
A photograph of a person meditating in a lotus position on a beach at sunset. The person's silhouette is visible against the bright sky. The ocean waves are in the foreground, and their reflection is clearly visible in the water.

The Present is
Reflection of the Past

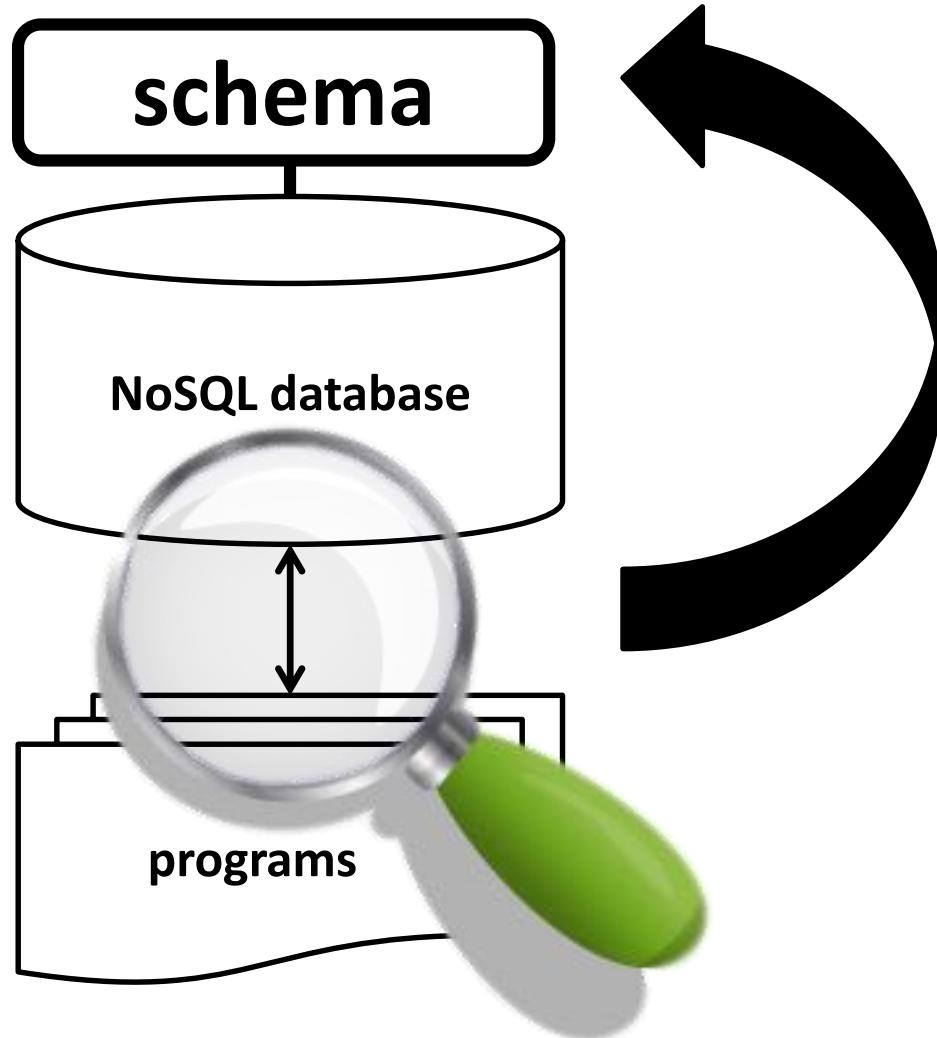
Extracting NoSQL Database Schema



Extracting NoSQL Database Schema



Extracting NoSQL Database Schema



Extracting NoSQL Database Schema

```
1  public String save(ContributionToSave contributionToSave) {
2      BasicDBObject authorQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getAuthor().getId()));
3      DBObject author = db.getCollection("author").findOne(authorQuery);
4      BasicDBObject showQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getShow().getId()));
5      DBObject show = db.getCollection("show").findOne(showQuery);
6      addContributionToAuthor(contributionToSave, authorQuery, author, show);
7      return "ok";
8  }
9
10 private void addContributionToAuthor(ContributionToSave contributionToSave, BasicDBObject authorQuery, DBObject
11     author, DBObject show) {
12     BasicDBList contributions = (BasicDBList) author.get("contributions");
13     if (contributions == null) {
14         contributions = new BasicDBList();
15         author.put("contributions", contributions);
16     }
17     BasicDBObject contribution = new BasicDBObject();
18     contribution.put("nick", contributionToSave.getNick());
19     BasicDBObject contributionShow = new BasicDBObject();
20     contributionShow.put("alias", show.get("alias"));
21     contributionShow.put("name", show.get("name"));
22     contributionShow.put("ref", new DBRef(db, "show", show.get("_id")));
23     contribution.put("show", contributionShow);
24     contributions.add(contribution);
25     db.getCollection("author").update(authorQuery, author);
}
```

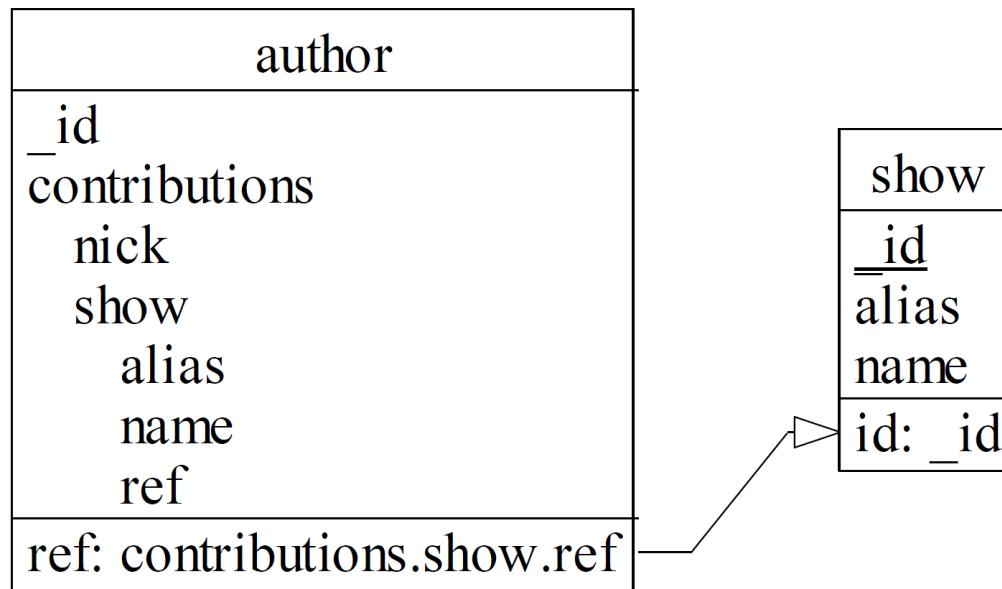
Extracting NoSQL Database Schema

```
1  public String save(ContributionToSave contributionToSave) {
2      BasicDBObject authorQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getAuthor().getId()));
3      DBObject author = db.getCollection("author").findOne(authorQuery);
4      BasicDBObject showQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getShow().getId()));
5      DBObject show = db.getCollection("show").findOne(showQuery);
6      addContributionToAuthor(contributionToSave, authorQuery, author, show);
7      return "ok";
8  }
9
10 private void addContributionToAuthor(ContributionToSave contributionToSave, BasicDBObject authorQuery, DBObject
11     author, DBObject show) {
12     BasicDBList contributions = (BasicDBList) author.get("contributions");
13     if (contributions == null) {
14         contributions = new BasicDBList();
15         author.put("contributions", contributions);
16     }
17     BasicDBObject contribution = new BasicDBObject();
18     contribution.put("nick", contributionToSave.getNick());
19     BasicDBObject contributionShow = new BasicDBObject();
20     contributionShow.put("alias", show.get("alias"));
21     contributionShow.put("name", show.get("name"));
22     contributionShow.put("ref", new DBRef(db, "show", show.get("_id")));
23     contribution.put("show", contributionShow);
24     contributions.add(contribution);
25     db.getCollection("author").update(authorQuery, author);
}
```

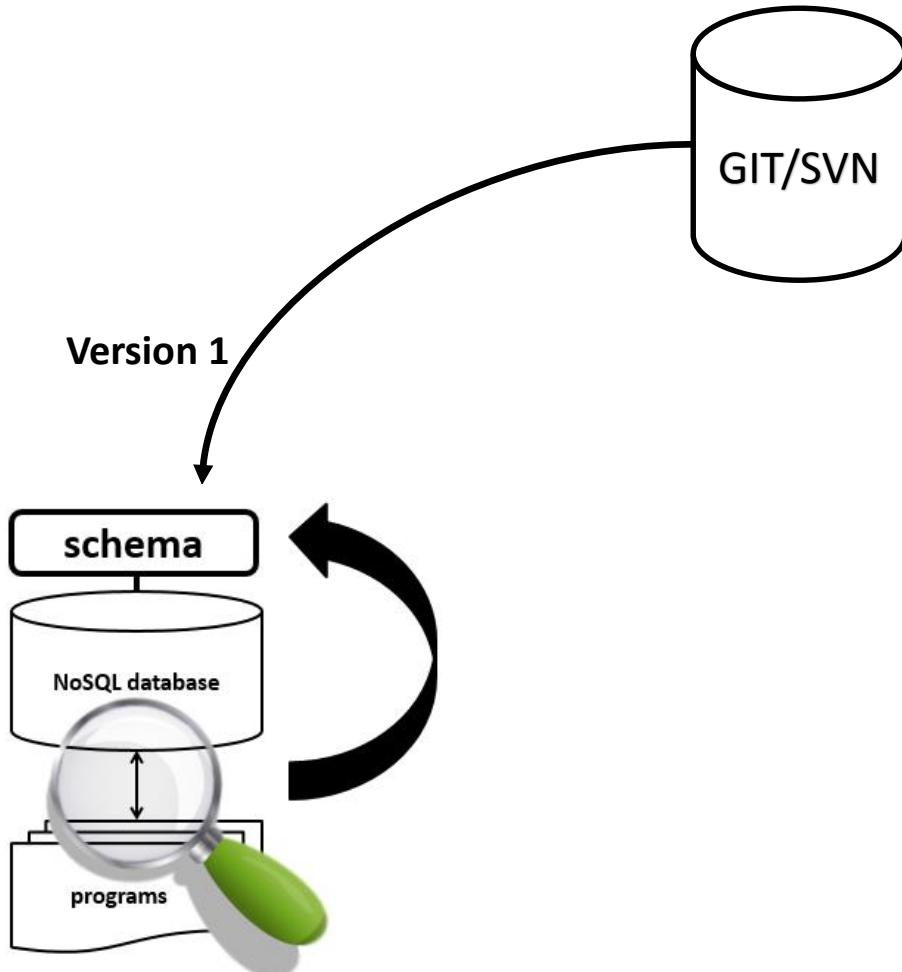
Extracting NoSQL Database Schema

```
1  public String save(ContributionToSave contributionToSave) {
2      BasicDBObject authorQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getAuthor().getId()));
3      DBObject author = db.getCollection("author").findOne(authorQuery);
4      BasicDBObject showQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getShow().getId()));
5      DBObject show = db.getCollection("show").findOne(showQuery);
6      addContributionToAuthor(contributionToSave, authorQuery, author, show);
7      return "ok";
8  }
9
10 private void addContributionToAuthor(ContributionToSave contributionToSave, BasicDBObject authorQuery, DBObject
11     author, DBObject show) {
12     BasicDBList contributions = (BasicDBList) author.get("contributions");
13     if (contributions == null) {
14         contributions = new BasicDBList();
15         author.put("contributions", contributions);
16     }
17     BasicDBObject contribution = new BasicDBObject();
18     contribution.put("nick", contributionToSave.getNick());
19     BasicDBObject contributionShow = new BasicDBObject();
20     contributionShow.put("alias", show.get("alias"));
21     contributionShow.put("name", show.get("name"));
22     contributionShow.put("ref", new DBRef(db, "show", show.get("_id")));
23     contribution.put("show", contributionShow);
24     contributions.add(contribution);
25     db.getCollection("author").update(authorQuery, author);
}
```

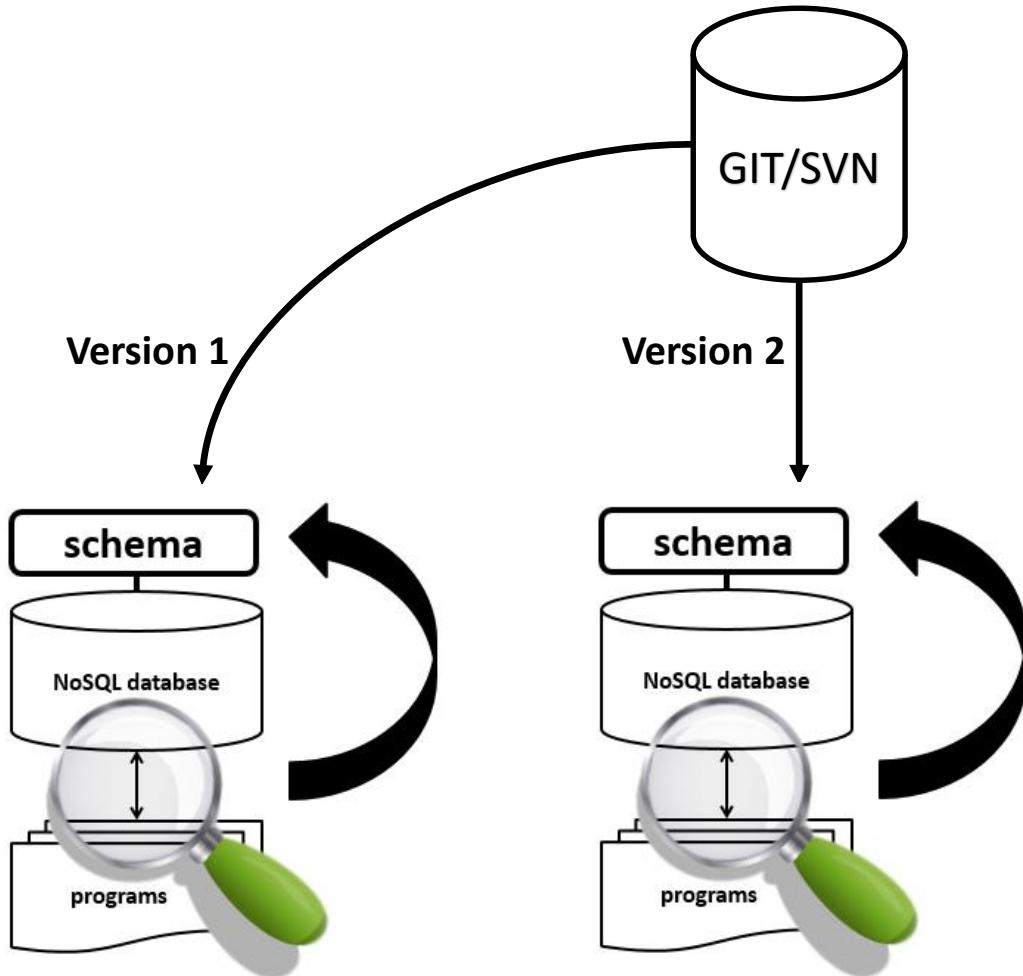
Extracting NoSQL Database Schema



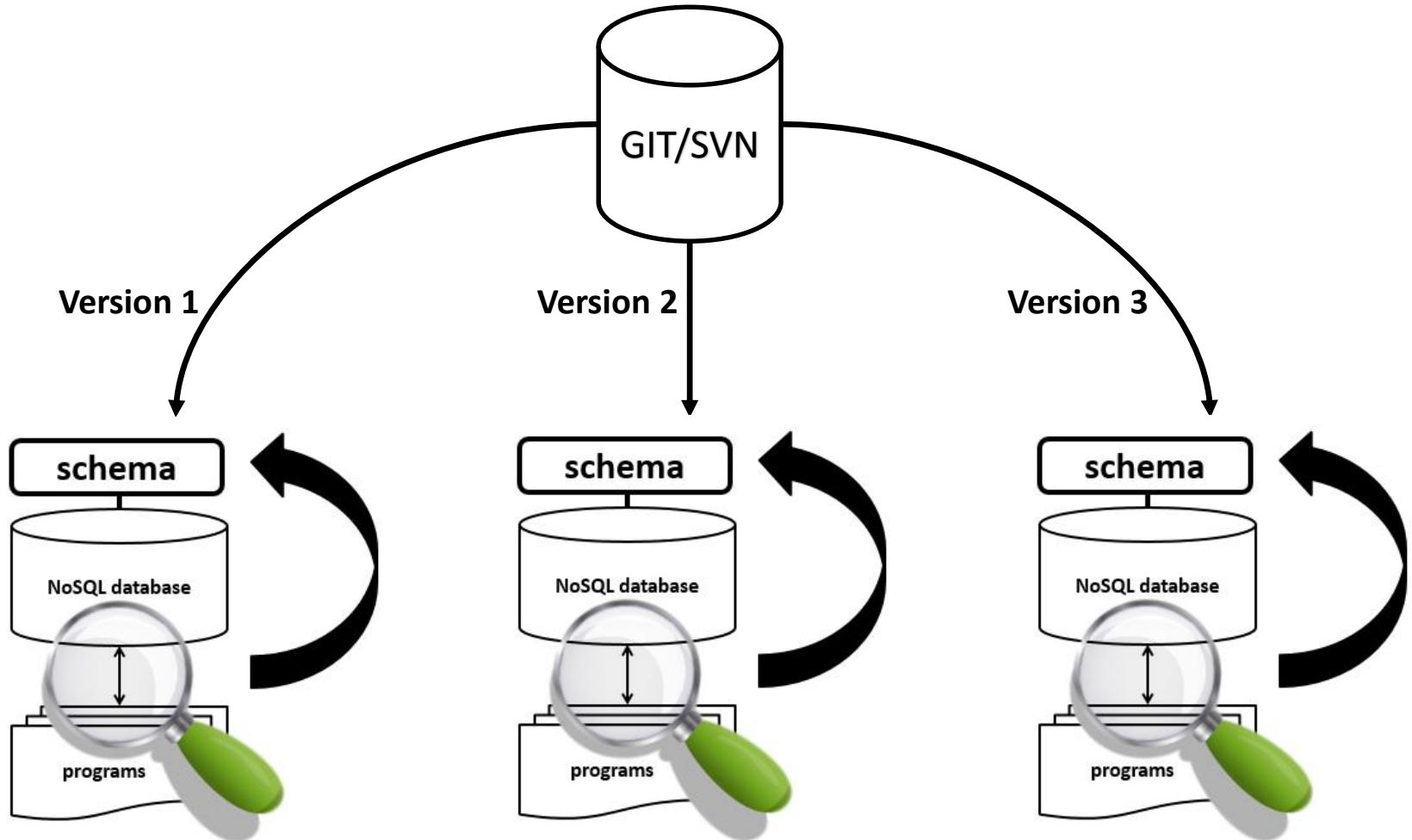
Extracting NoSQL Database SchemaS



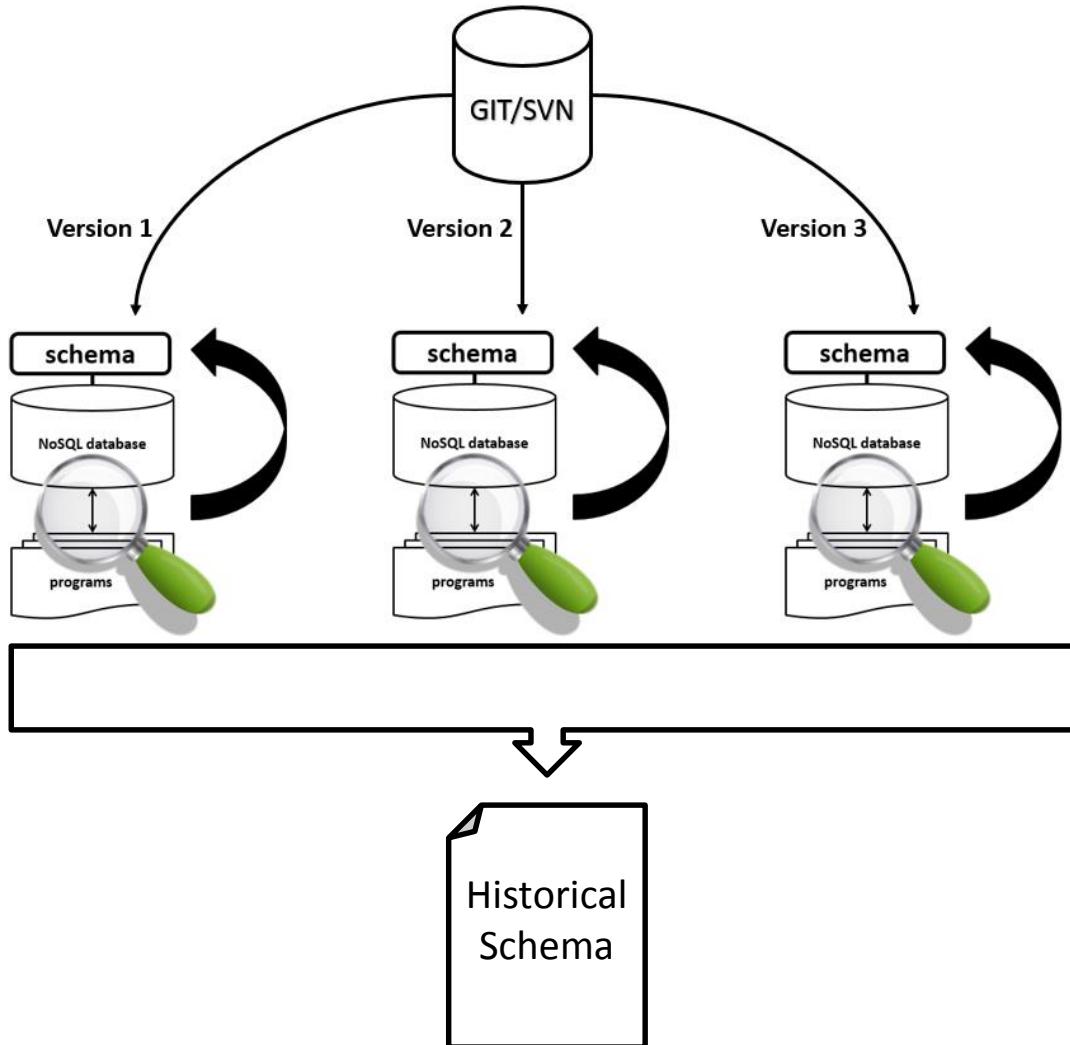
Extracting NoSQL Database SchemaS



Extracting NoSQL Database SchemaS



Extracting NoSQL Database SchemaS



A Subject System

Titos 23

A Subject System

Non-Profit Hungarian Radio

Titos 23

A Subject System

Non-Profit Hungarian Radio
Java + MongoDB



A Subject System

Non-Profit Hungarian Radio

Java + MongoDB

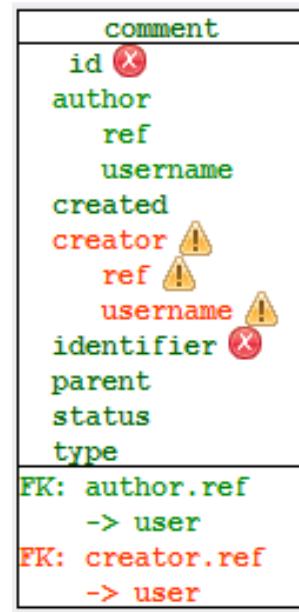
Two-Year History (303 versions)



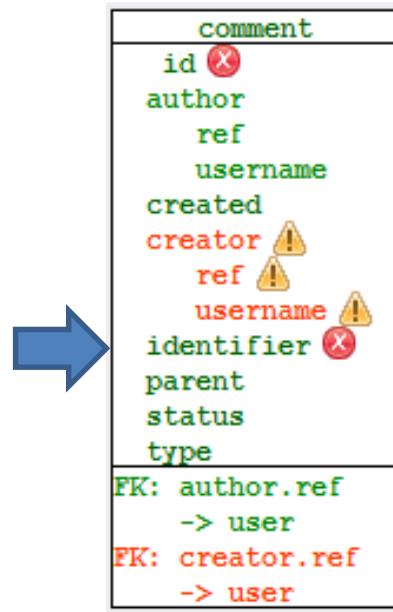
Extracting Historical Schema

author	bookmark	comment	episode	mix	page	show	stat download	stat icecast	tags	user
id alias contributions nick show alias name ref email introduction name FK: contributions.show.ref -> show	created creator ref username from title to FK: creator.ref -> user	id X author ref username created creator ref username identifier X parent status type FK: author.ref -> user FK: creator.ref -> user	id alias bookmarks created creator ref username from title to created extra plannedFrom plannedTo show alias name ref tags name type text content title FK: bookmarks.creator.ref -> user FK: show.ref -> show	alias category date id show ref FK: show.ref -> show	id alias content format title type	id alias contributors author alias ref nick description name schedulings validFrom validTo status type FK: contributors.author.ref -> author	id bytes endDate position realStartDate startDate time token	id tilos 128 mp3 tilos 32 mp3	id name type value	id author email facebook link password passwordChangeToken passwordChangeTokenCreated X role role id salt username FK: author -> author

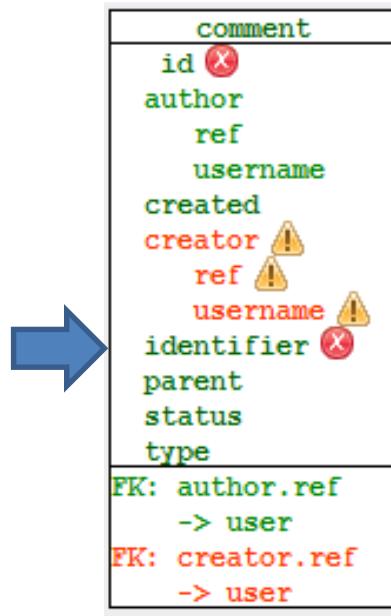
Type Mismatch Detection



Type Mismatch Detection



Type Mismatch Detection

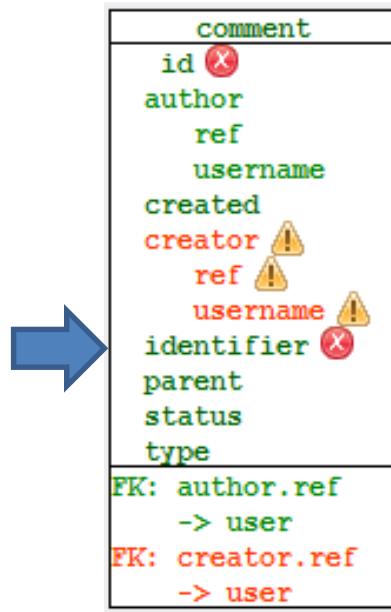


```
public List<CommentData> list(CommentType type, int id) {  
    BasicDBObject query = new BasicDBObject();  
    query.put("identifier", id);  
    DBCursor comments = db.getCollection("comment").find(query);  
  
    public List<CommentData> list(CommentType type, String id) {  
        BasicDBObject query = new BasicDBObject();  
        query.put("identifier", id);  
        DBCursor comments = db.getCollection("comment").find(query);  
    }  
}
```

(a)

(b)

Type Mismatch Detection



```
public List<CommentData> list(CommentType type, int id) {  
    BasicDBObject query = new BasicDBObject();  
    query.put("identifier", id);  
    DBCursor comments = db.getCollection("comment").find(query);  
  
public List<CommentData> list(CommentType type, String id) {  
    BasicDBObject query = new BasicDBObject();  
    query.put("identifier", id);  
    DBCursor comments = db.getCollection("comment").find(query);
```

(a)

(b)

Renaming Detection

bookmark	comment	episode
created	id	alias
creator	author	bookmarks
ref	ref	created
username	username	creator
from	ref	ref
title	username	username
to	identifier	from
FK: creator.ref -> user		title
		to
		created
		extra
		plannedFrom
		plannedTo
		show
		alias
		name
		ref
		tags
		name
		type
		text
		content
		title
FK: bookmarks.creator.ref -> user		
FK: show.ref -> show		

Renaming Detection

bookmark	comment	episode
created		
creator		
ref		
username		
from		
title		
to		
FK: creator.ref -> user		
	id	
	author	
	ref	
	username	
	created	
	creator	
	ref	
	username	
	identifier	
	parent	
	status	
	type	
	FK: author.ref -> user	
	FK: creator.ref -> user	
		id
		alias
		bookmarks
		created
		creator
		ref
		username
		from
		title
		to
		created
		extra
		plannedFrom
		plannedTo
		show
		alias
		name
		ref
		tags
		name
		type
		text
		content
		title
		FK: bookmarks.creator.ref -> user
		FK: show.ref -> show

Renaming Detection



bookmark
created
creator
ref
username
from
title
to
FK: creator.ref -> user

comment
id (X)
author
ref
username
created
creator
ref
username
identifier (X)
parent
status
type
FK: author.ref -> user
FK: creator.ref -----

episode
id
alias
bookmarks
created
creator
ref
username
from
title
to
created
extra
plannedFrom
plannedTo
show

```
BasicDBObject bookmark = new BasicDBObject();
bookmark.put("created", new Date());
...
db.getCollection("bookmark").insert(bookmark);
```

(a)

```
DBObject episode = db.getCollection("episode").findOne(q);
BasicDBObject bookmark = new BasicDBObject();
bookmark.put("created", new Date());
...
if (episode.get("bookmarks") == null)
    episode.put("bookmarks", new BasicDBList());
((BasicDBList) episode.get("bookmarks")).add(bookmark);
```

(b)

Data Loss Detection

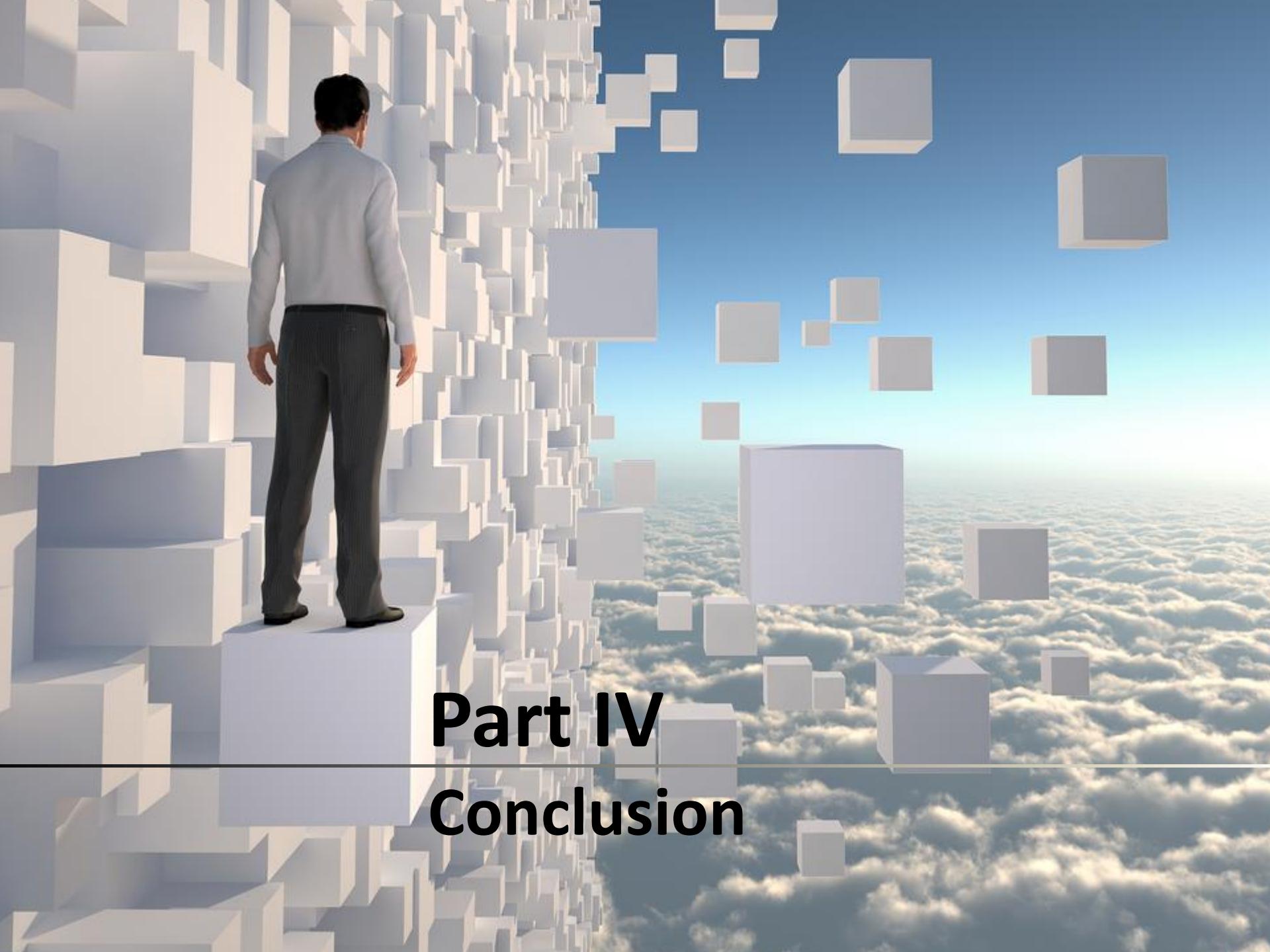


user	
<code>id</code>	
<code>author</code>	
<code>email</code>	
<code>facebook</code>	
<code>link</code>	
<code>password</code>	
<code>passwordChangeToken</code>	
<code>passwordChangeTokenCreated</code>	✖
<code>role</code>	
<code>role id</code>	
<code>salt</code>	
<code>username</code>	
<code>FK: author</code>	
<code>-> author</code>	

Data Loss Detection



```
DBObject user = db.getCollection("user").findOne(  
    new BasicDBObject("email", passwordReset.getEmail()));  
String token = authUtil.generateSalt();  
user.put("passwordChangeTokenCreated", new Date());  
user.put("passwordChangeTokenCreated", token);  
db.getCollection("user").update(  
    new BasicDBObject("username", user.get("username")), user);
```



A 3D rendering of a man in a white shirt and dark trousers standing on a large white cube. He is looking out over a vast landscape filled with numerous other white cubes of various sizes, some floating in the air and others resting on the ground. The background is a clear blue sky.

Part IV Conclusion

A black and white photograph of a man in a dark suit, white shirt, and tie. He is standing on what appears to be a ship's deck, leaning against a railing. He is looking through a pair of binoculars towards the horizon. The background shows a bright, hazy sky and a distant horizon.

Summary

An automatic approach to infer the schema of
schema-less NoSQL database

A black and white photograph of a man in a dark suit, white shirt, and tie. He is standing on what appears to be a ship's deck, looking through a pair of binoculars towards the horizon. The background shows a bright sky and some distant structures.

Summary

An automatic approach to infer the schema of
schema-less NoSQL database

... designed to be applied to the whole system history

A black and white photograph of a man in a dark suit, white shirt, and tie. He is standing on what appears to be a ship's deck, looking through a pair of binoculars towards the horizon. The background shows a bright sky and some distant structures.

Summary

An automatic approach to infer the schema of schema-less NoSQL database

... designed to be applied to the whole system history

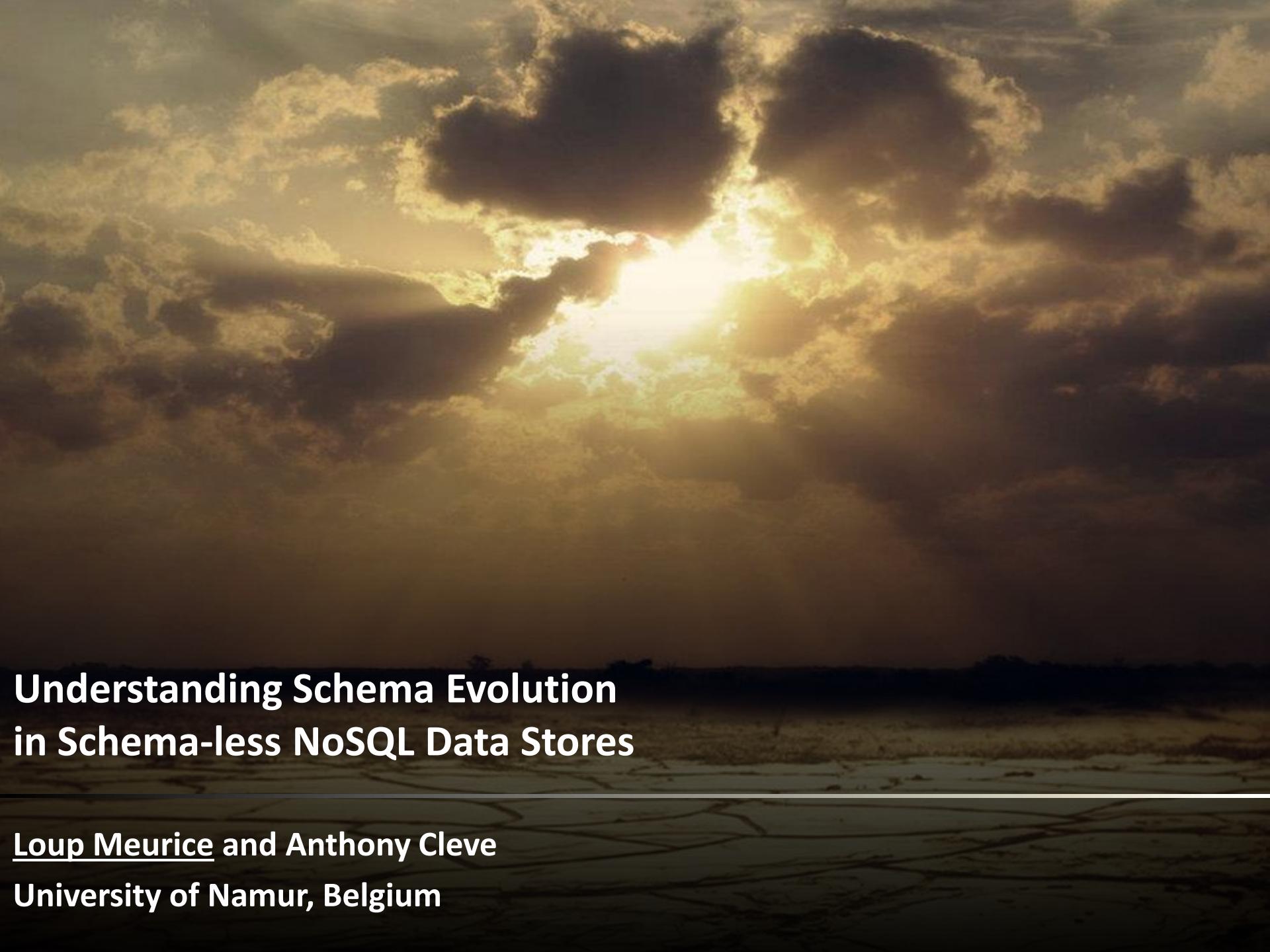
... for preventing program crashes and data losses



Summary

An automatic approach to infer the schema of schema-less NoSQL database

- ... designed to be applied to the whole system history
- ... for preventing program crashes and data losses
- ... currently designed for Java systems using MongoDB



Understanding Schema Evolution in Schema-less NoSQL Data Stores

Loup Meurice and Anthony Cleve
University of Namur, Belgium