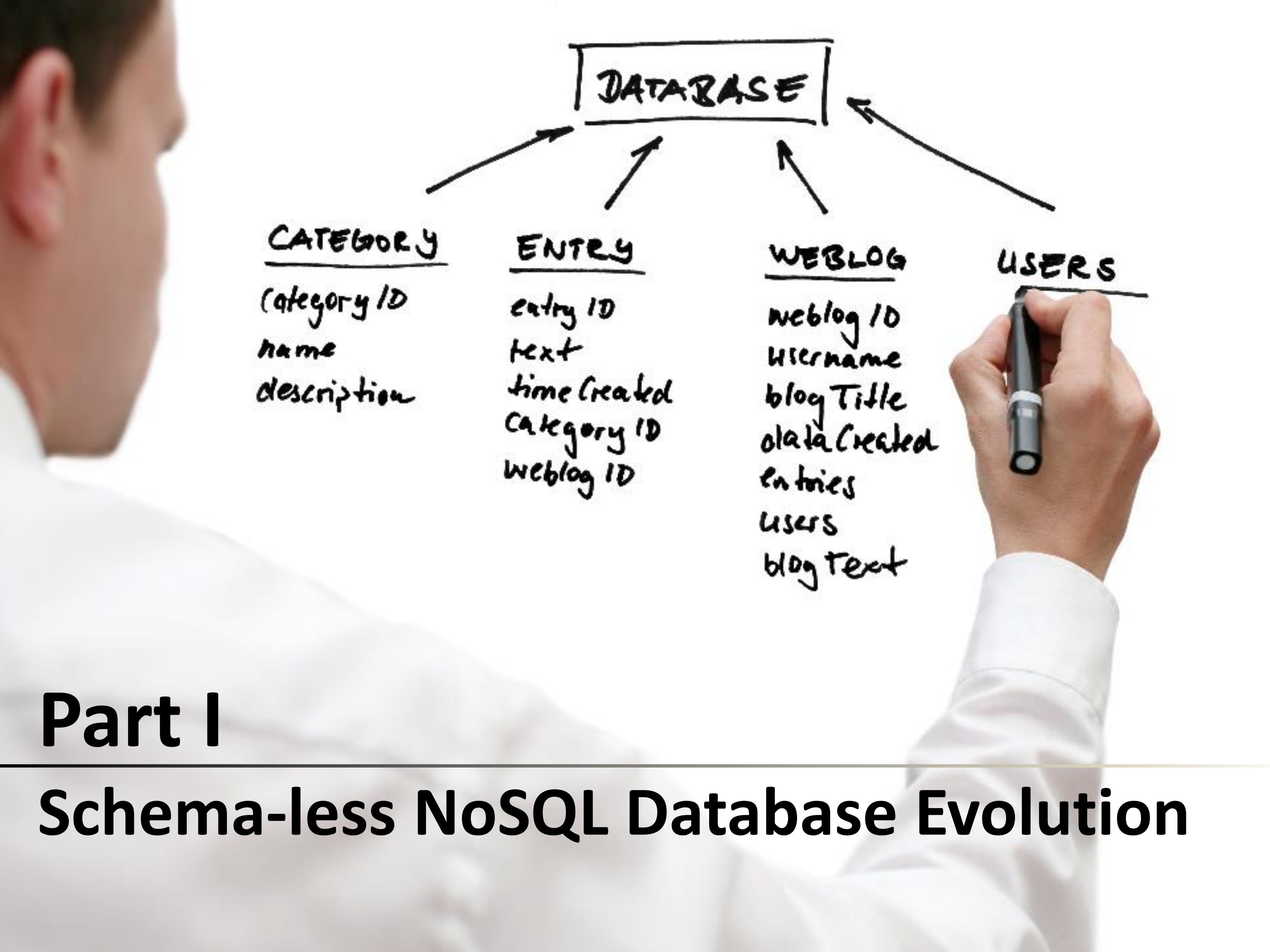
Understanding Schema Evolution in Schema-less NoSQL Data Stores

Loup Meurice and Anthony Cleve

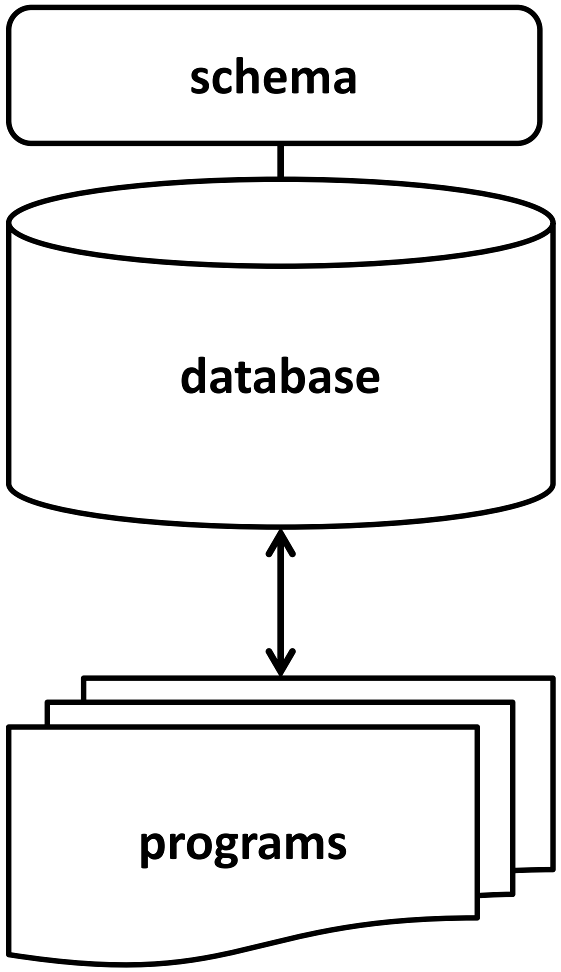
Faculty of Informatics
University of Namur
Belgium



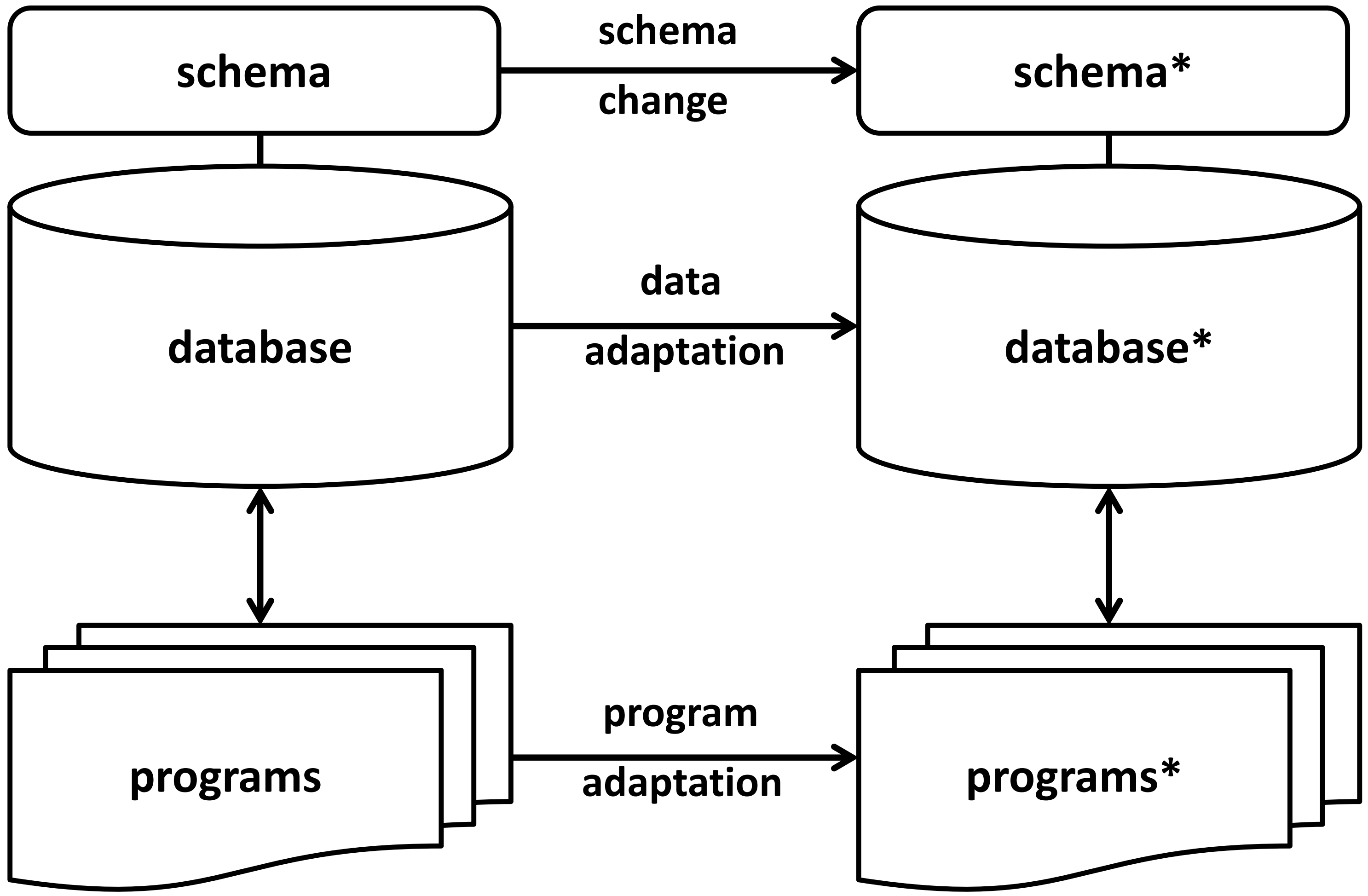
Part I

Schema-less NoSQL Database Evolution

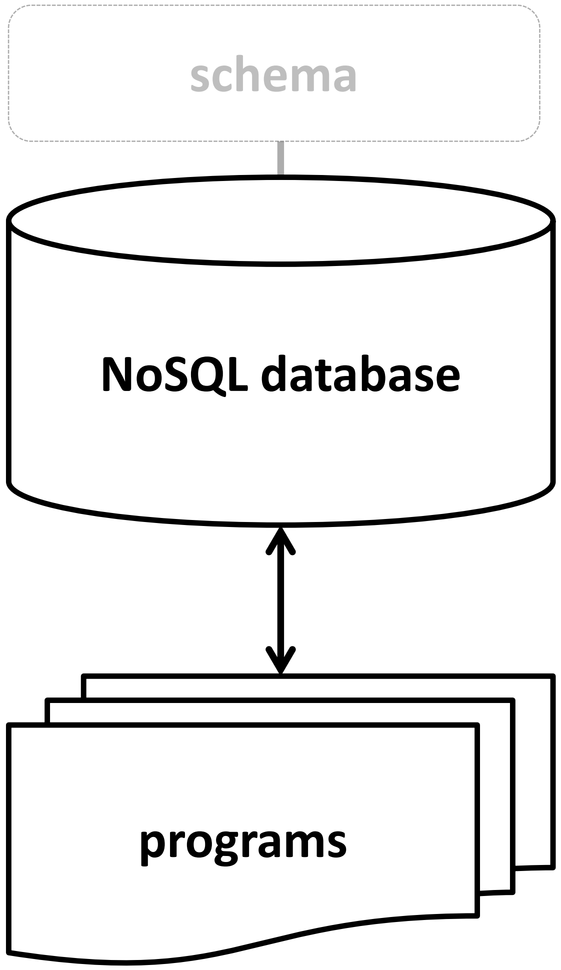
Relational Database



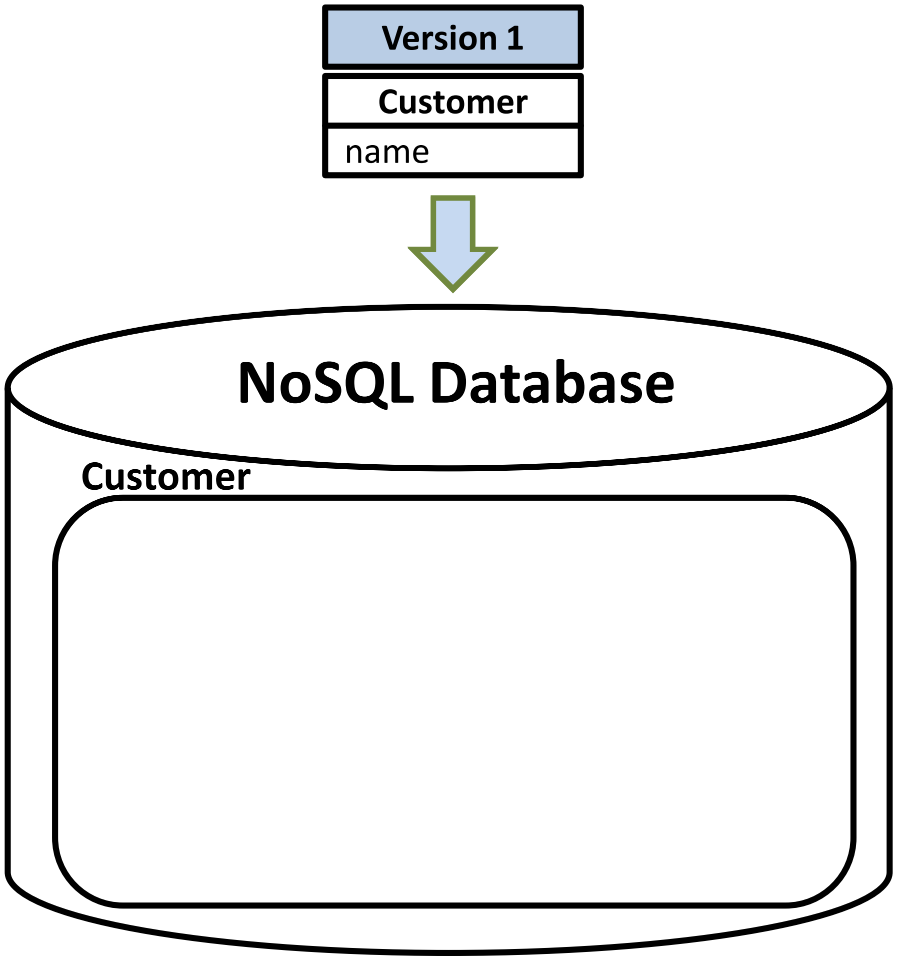
Relational Database



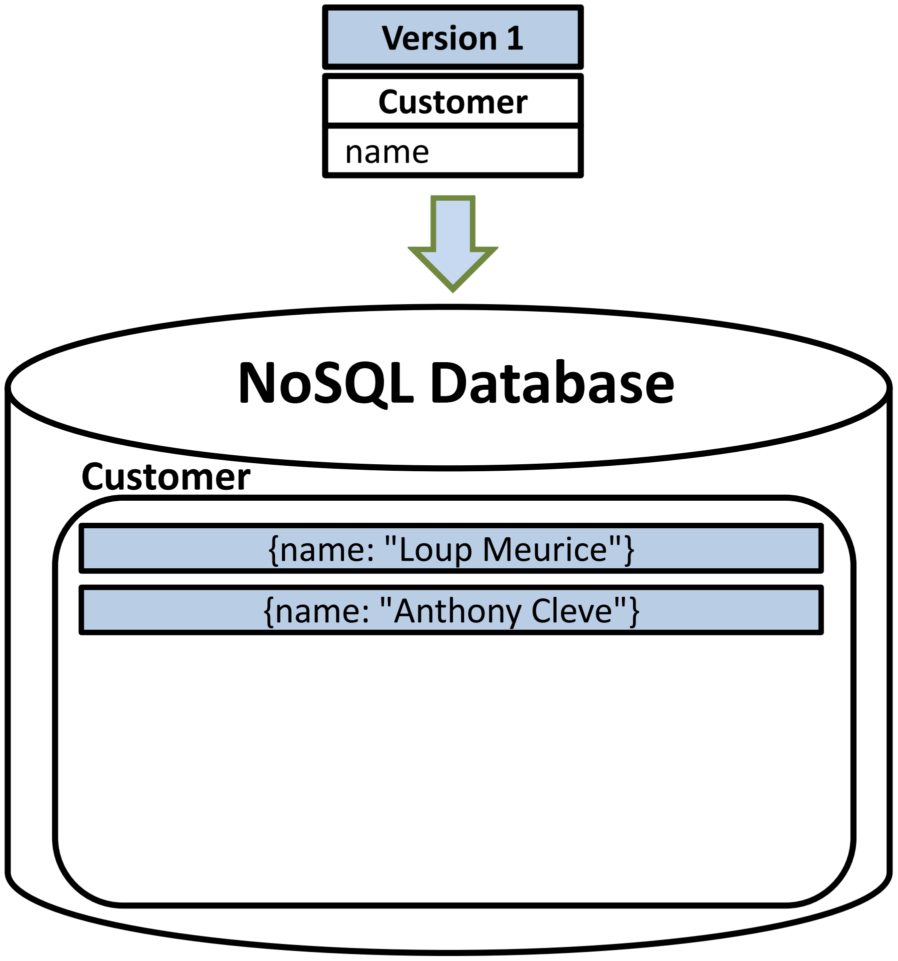
Schema-less NoSQL Database



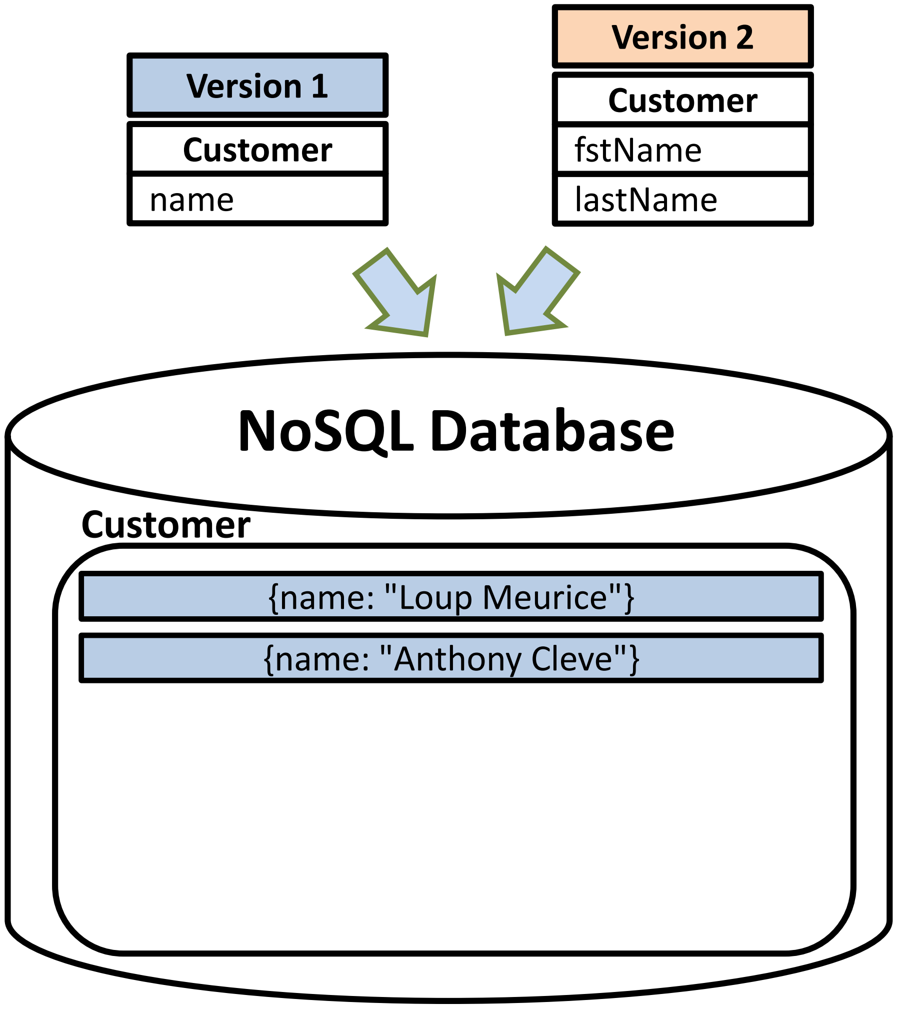
Schema-less NoSQL Database



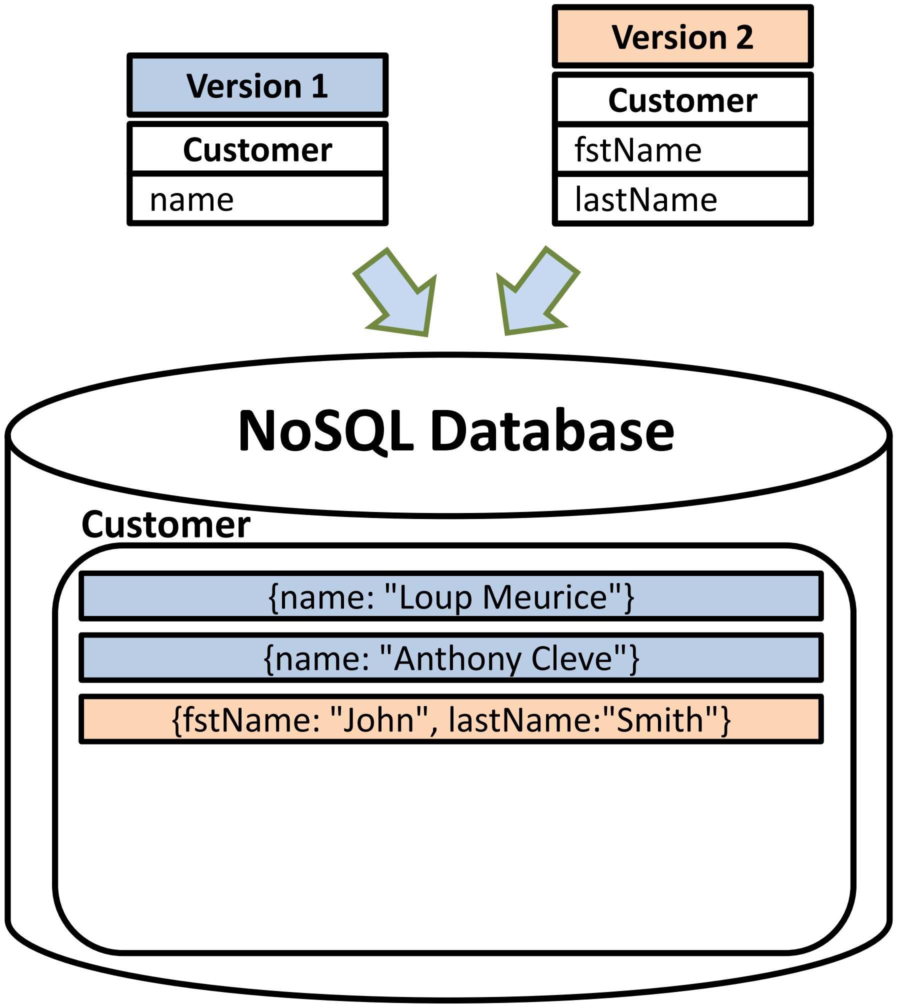
Schema-less NoSQL Database



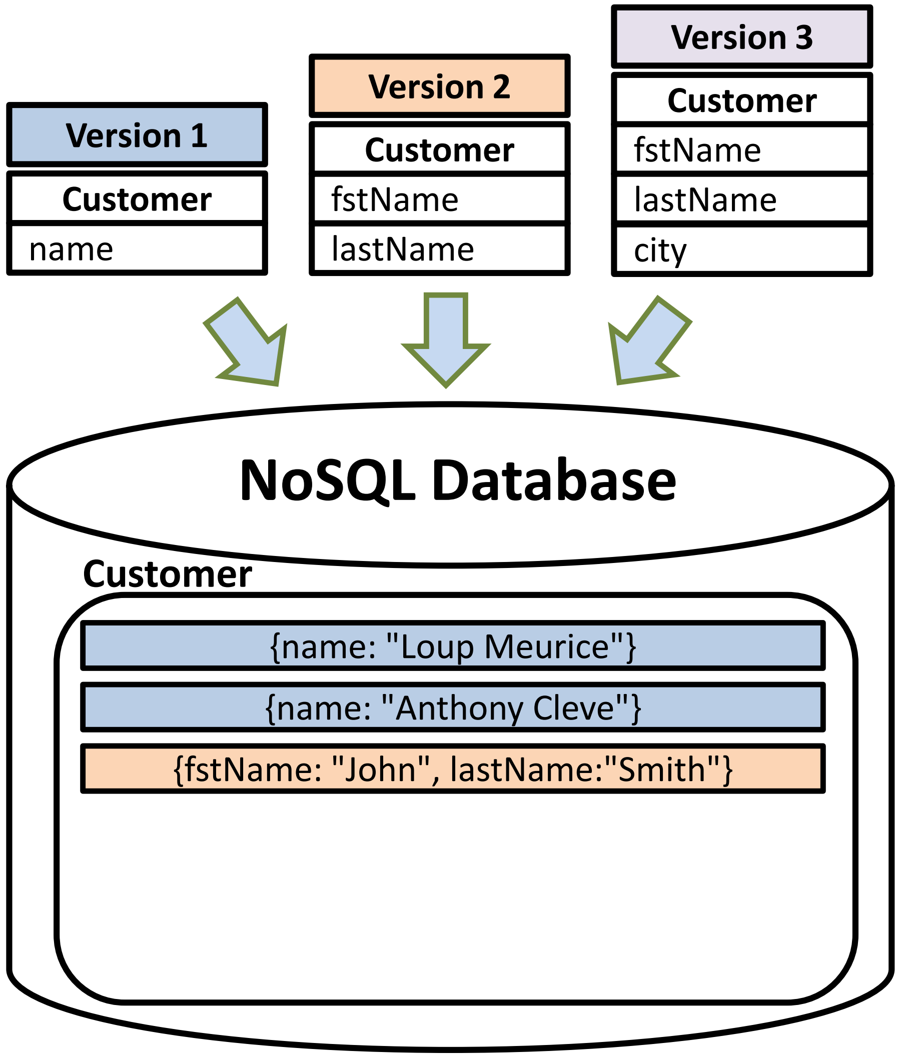
Schema-less NoSQL Database



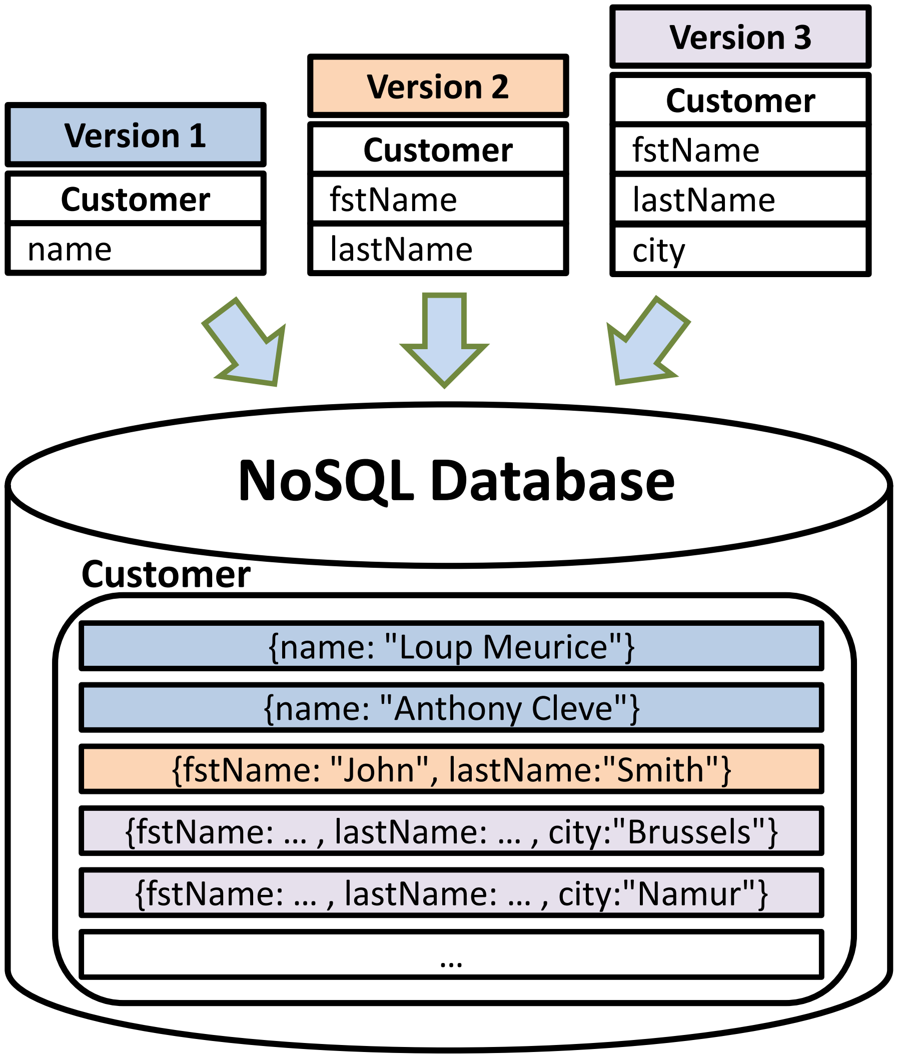
Schema-less NoSQL Database



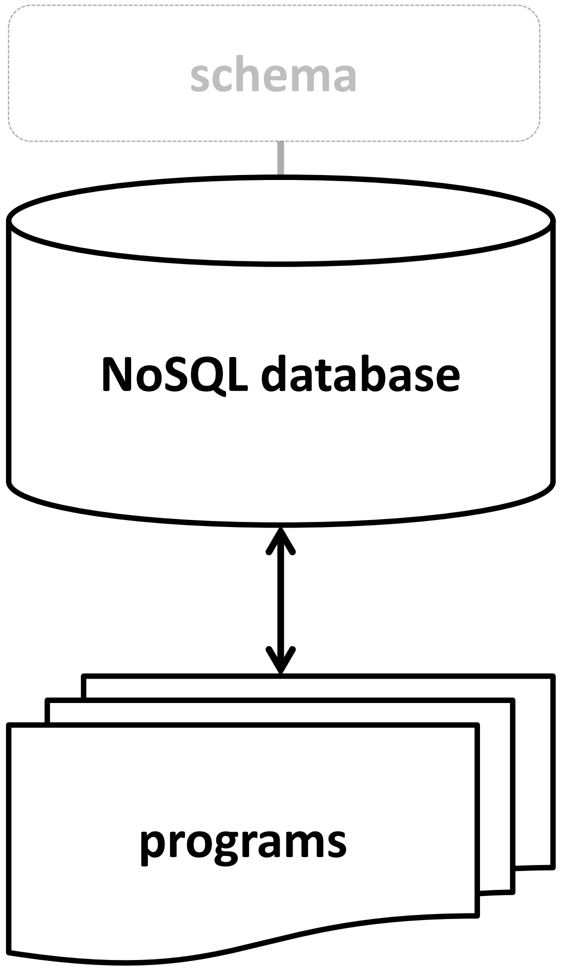
Schema-less NoSQL Database



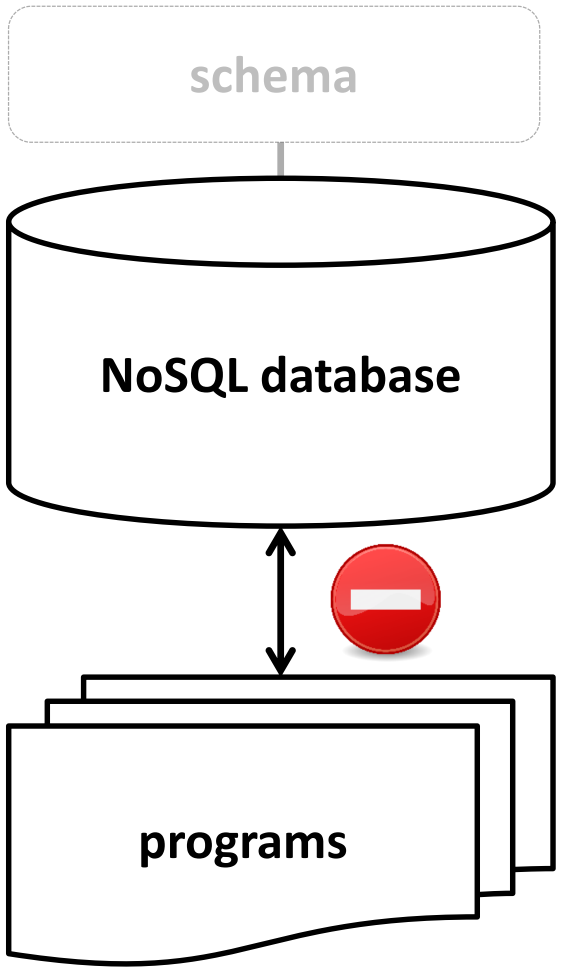
Schema-less NoSQL Database



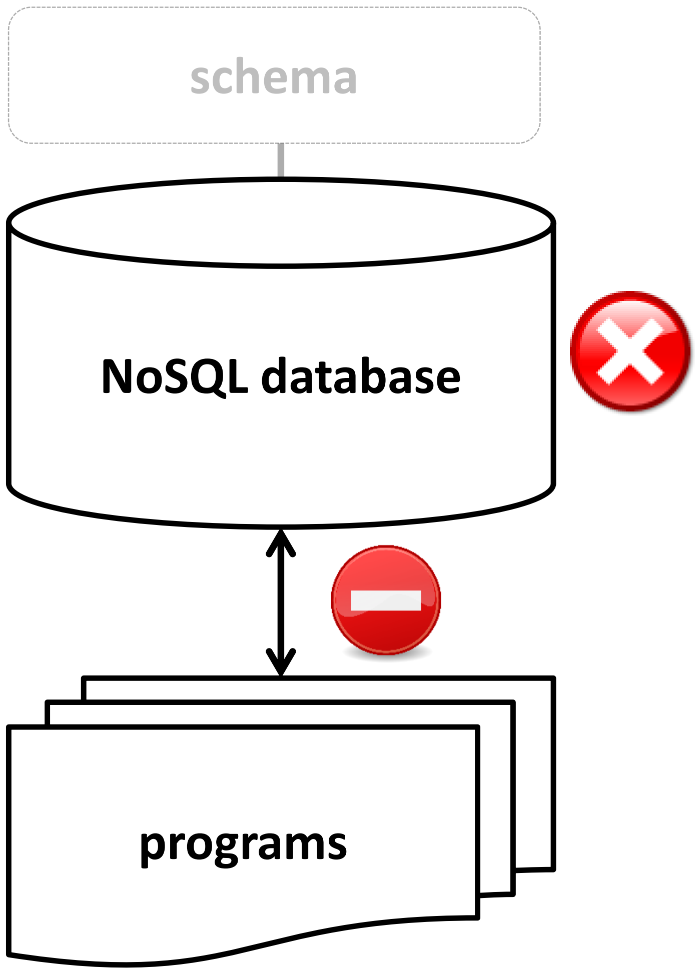
Schema-less NoSQL Database

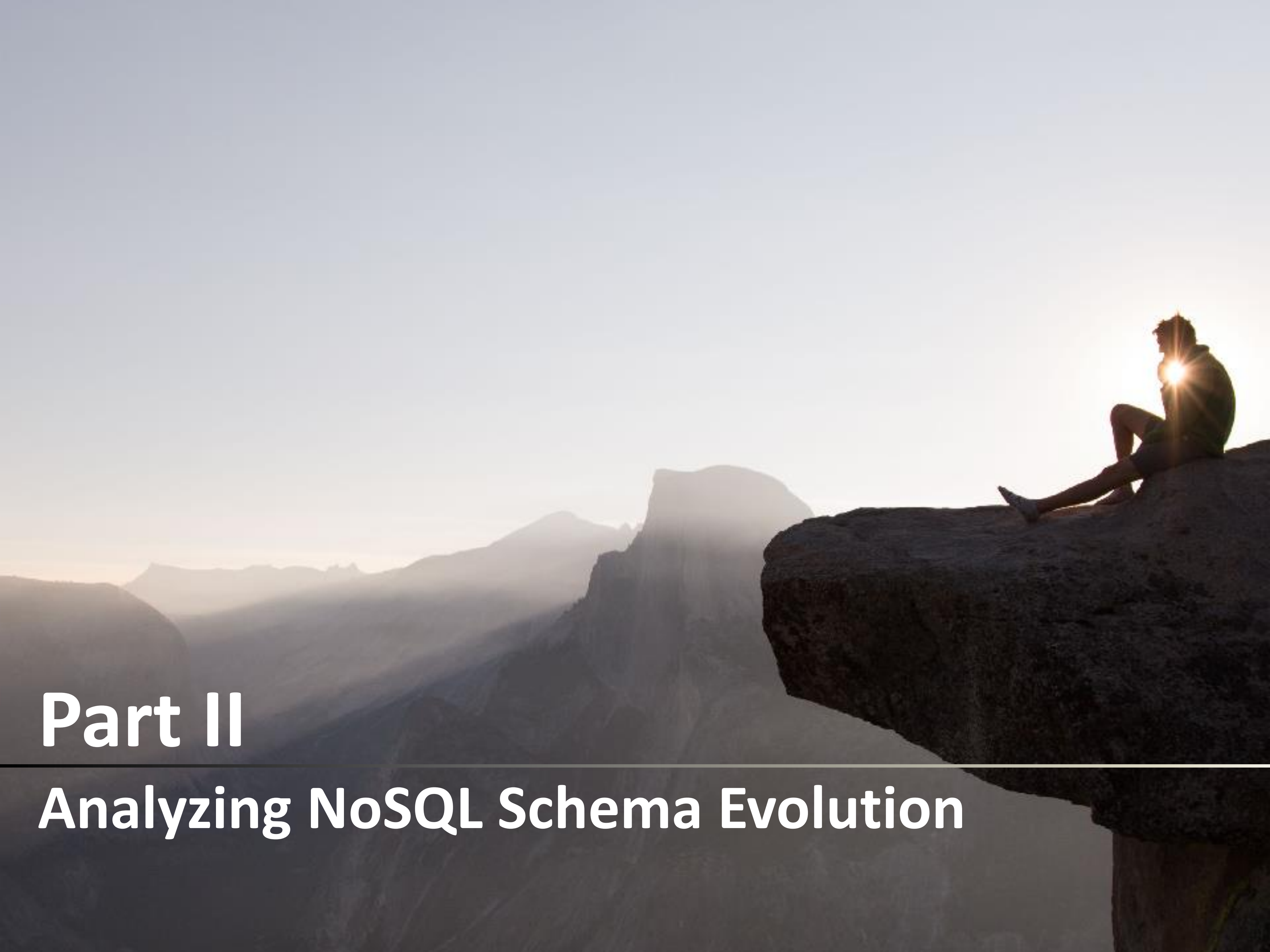


Schema-less NoSQL Database



Schema-less NoSQL Database



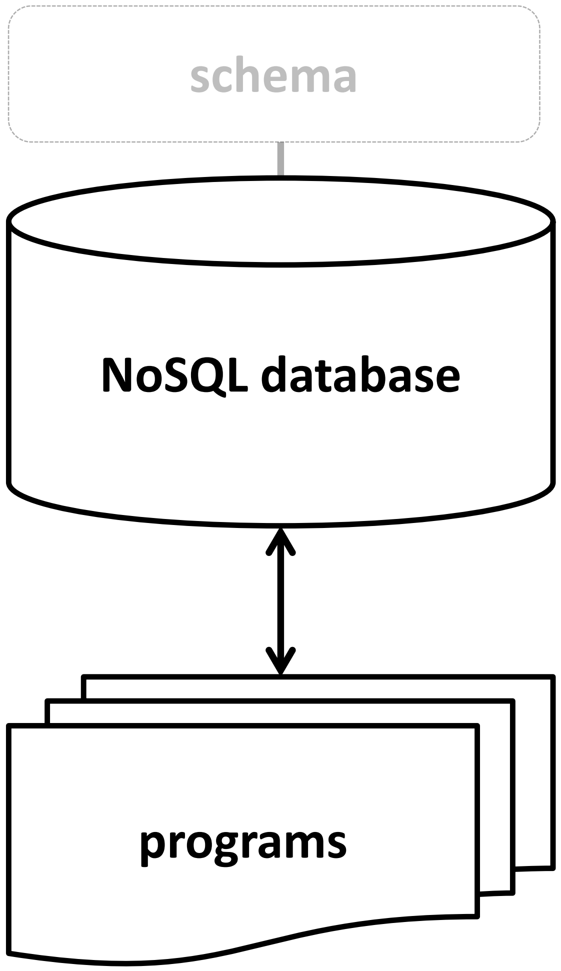


Part II

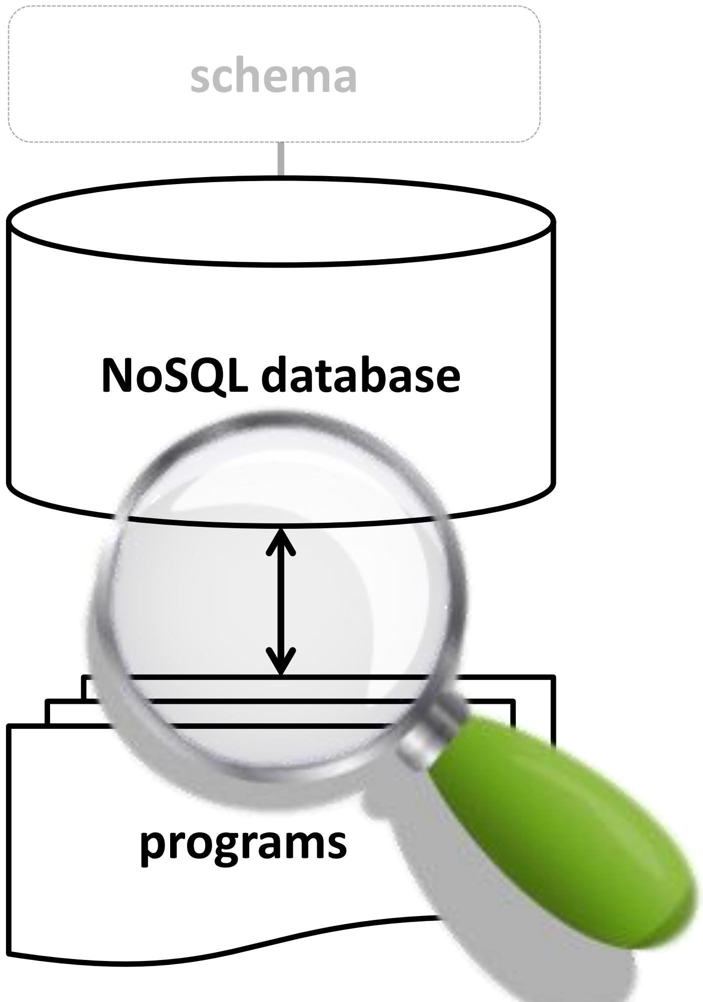
Analyzing NoSQL Schema Evolution

The Present is Reflection of the Past

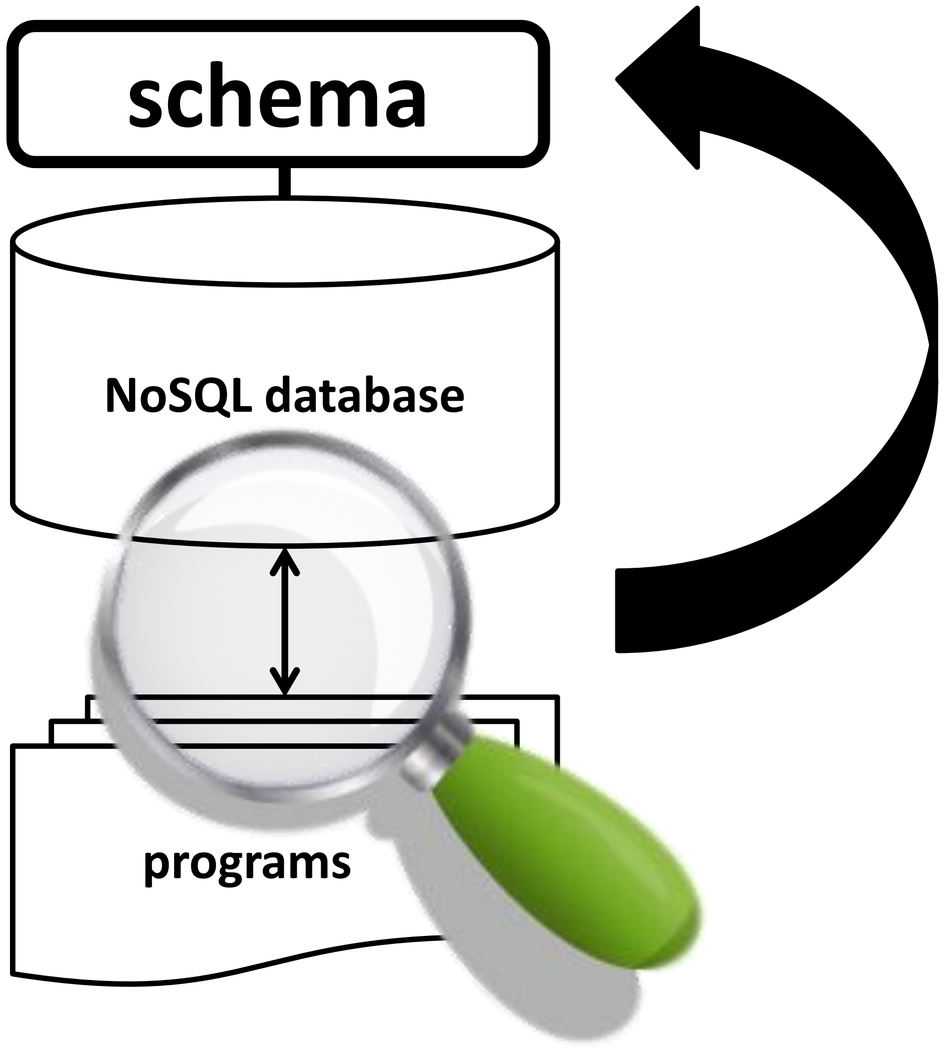
Extracting NoSQL Database Schema



Extracting NoSQL Database Schema



Extracting NoSQL Database Schema

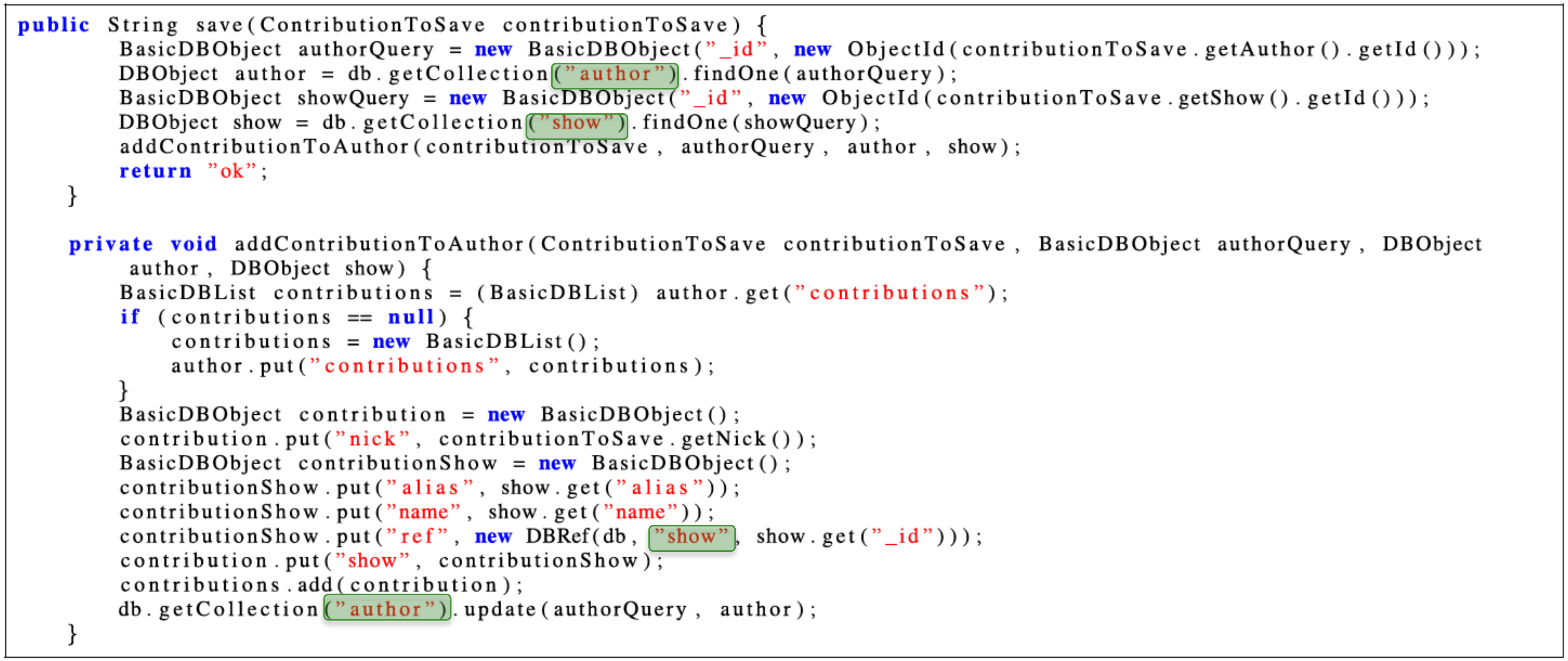


Extracting NoSQL Database Schema

```
public String save(ContributionToSave contributionToSave) {
        BasicDBObject authorQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getAuthor().getId()));
        DBObject author = db.getCollection("author").findOne(authorQuery);
        BasicDBObject showQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getShow().getId()));
        DBObject show = db.getCollection("show").findOne(showQuery);
        addContributionToAuthor(contributionToSave, authorQuery, author, show);
        return "ok";
    }

    private void addContributionToAuthor(ContributionToSave contributionToSave, BasicDBObject authorQuery, DBObject
         author, DBObject show) {
        BasicDBList contributions = (BasicDBList) author.get("contributions");
        if (contributions == null) {
            contributions = new BasicDBList();
            author.put("contributions", contributions);
        }
        BasicDBObject contribution = new BasicDBObject();
        contribution.put("nick", contributionToSave.getNick());
        BasicDBObject contributionShow = new BasicDBObject();
        contributionShow.put("alias", show.get("alias"));
        contributionShow.put("name", show.get("name"));
        contributionShow.put("ref", new DBRef(db, "show", show.get("_id")));
        contribution.put("show", contributionShow);
        contributions.add(contribution);
        db.getCollection("author").update(authorQuery, author);
    }
```

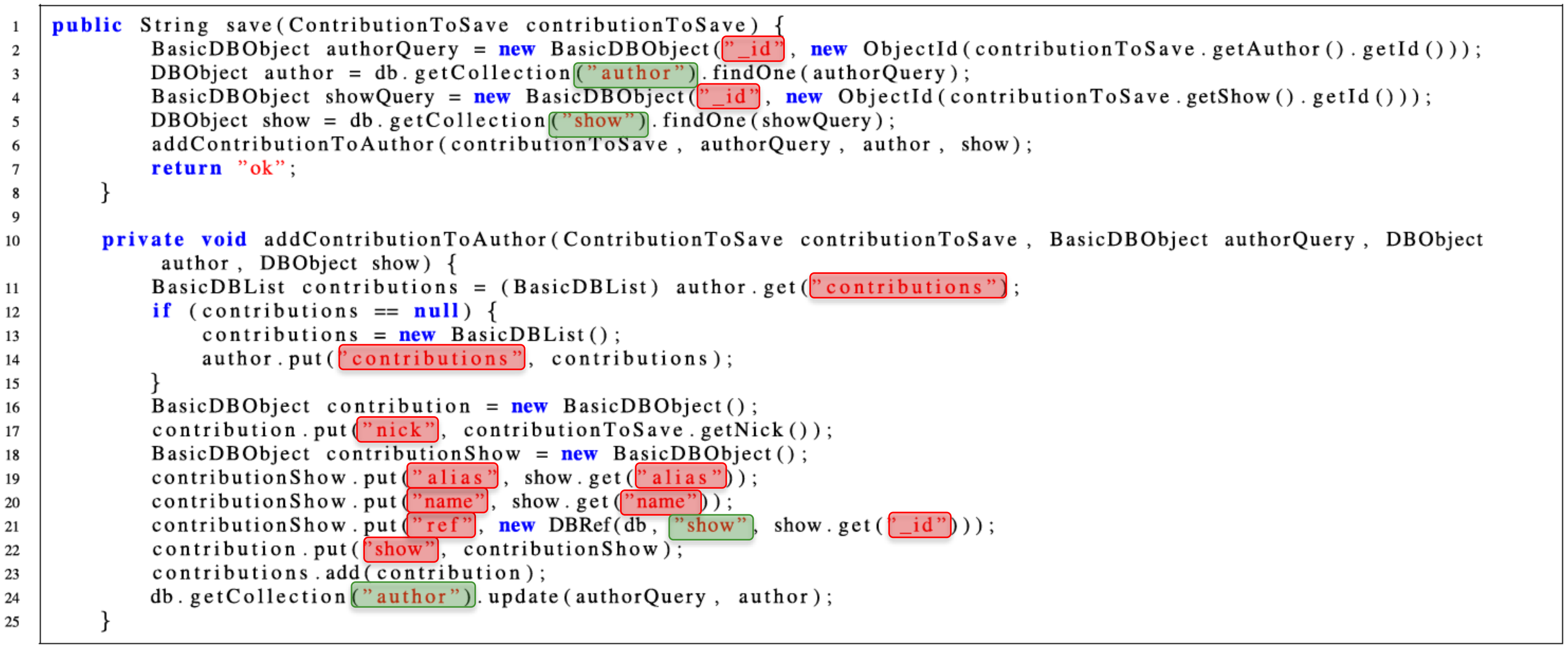
Extracting NoSQL Database Schema



```java
public String save(ContributionToSave contributionToSave) {
        BasicDBObject authorQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getAuthor().getId()));
        DBObject author = db.getCollection("author").findOne(authorQuery);
        BasicDBObject showQuery = new BasicDBObject("_id", new ObjectId(contributionToSave.getShow().getId()));
        DBObject show = db.getCollection("show").findOne(showQuery);
        addContributionToAuthor(contributionToSave, authorQuery, author, show);
        return "ok";
    }

    private void addContributionToAuthor(ContributionToSave contributionToSave, BasicDBObject authorQuery, DBObject
         author, DBObject show) {
        BasicDBList contributions = (BasicDBList) author.get("contributions");
        if (contributions == null) {
            contributions = new BasicDBList();
            author.put("contributions", contributions);
        }
        BasicDBObject contribution = new BasicDBObject();
        contribution.put("nick", contributionToSave.getNick());
        BasicDBObject contributionShow = new BasicDBObject();
        contributionShow.put("alias", show.get("alias"));
        contributionShow.put("name", show.get("name"));
        contributionShow.put("ref", new DBRef(db, "show", show.get("_id")));
        contribution.put("show", contributionShow);
        contributions.add(contribution);
        db.getCollection("author").update(authorQuery, author);
    }
```

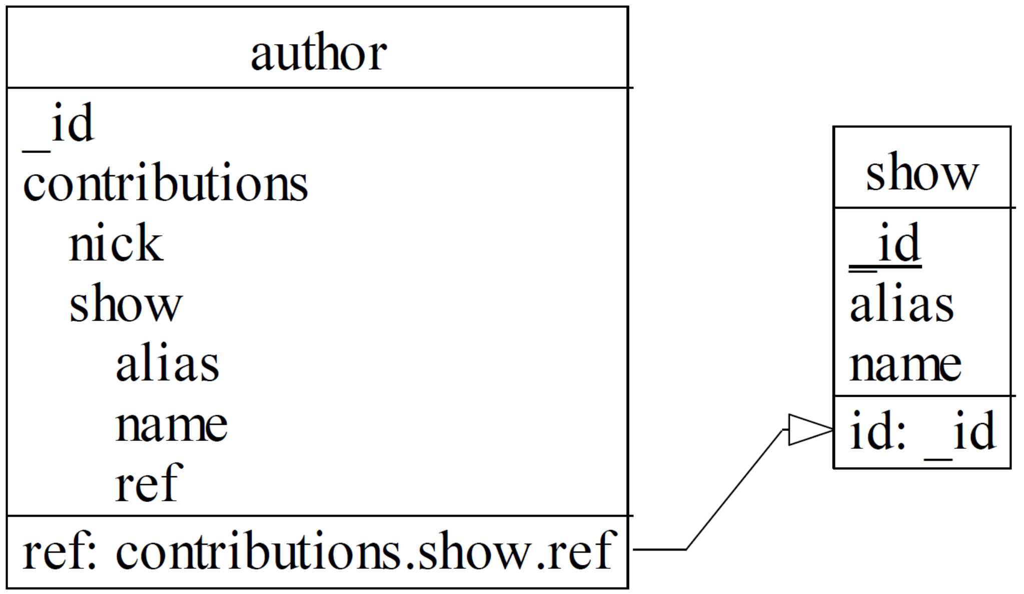
Extracting NoSQL Database Schema



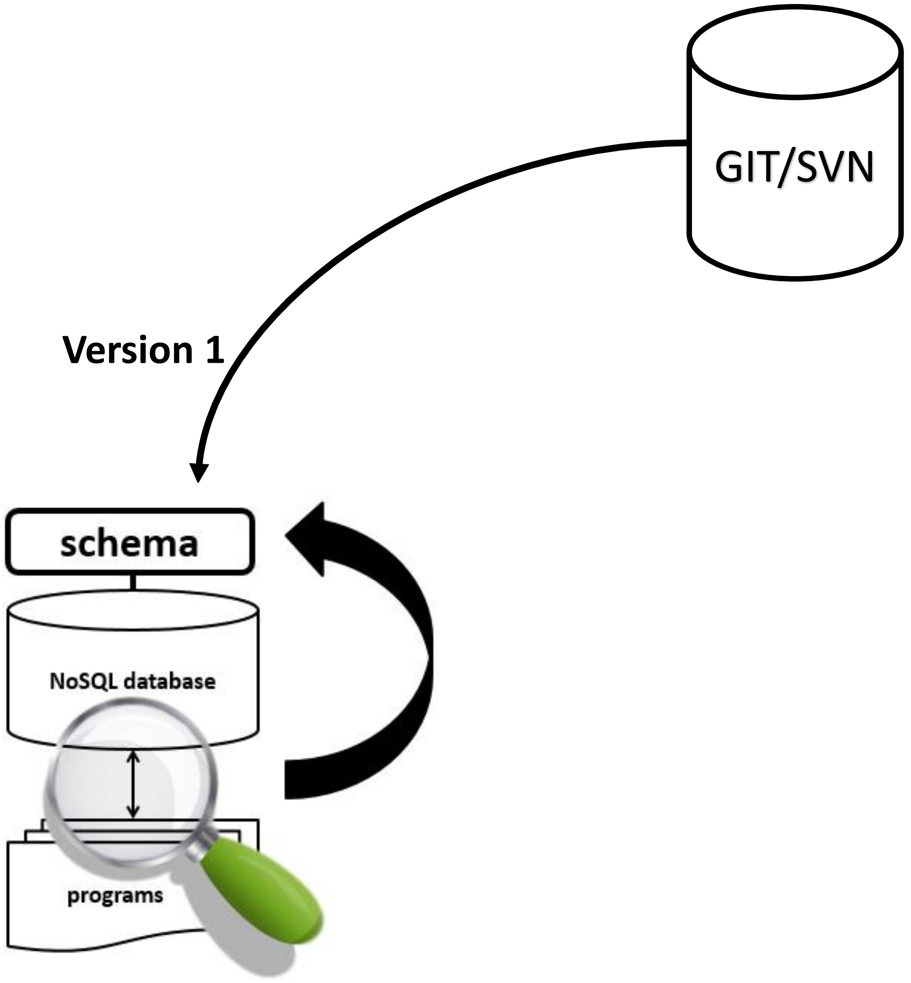
```
public String save(ContributionToSave contributionToSave) {
        BasicDBObject authorQuery = new BasicDBObject(”_id”, new ObjectId(contributionToSave.getAuthor().getId()));
        DBObject author = db.getCollection(”author”).findOne(authorQuery);
        BasicDBObject showQuery = new BasicDBObject(”_id”, new ObjectId(contributionToSave.getShow().getId()));
        DBObject show = db.getCollection(”show”).findOne(showQuery);
        addContributionToAuthor(contributionToSave, authorQuery, author, show);
        return ”ok”;
    }

    private void addContributionToAuthor(ContributionToSave contributionToSave, BasicDBObject authorQuery, DBObject
         author, DBObject show) {
        BasicDBList contributions = (BasicDBList) author.get(”contributions”);
        if (contributions == null) {
            contributions = new BasicDBList();
            author.put(”contributions”, contributions);
        }
        BasicDBObject contribution = new BasicDBObject();
        contribution.put(”nick”, contributionToSave.getNick());
        BasicDBObject contributionShow = new BasicDBObject();
        contributionShow.put(”alias”, show.get(”alias”));
        contributionShow.put(”name”, show.get(”name”));
        contributionShow.put(”ref”, new DBRef(db, ”show”, show.get(”_id”)));
        contribution.put(”show”, contributionShow);
        contributions.add(contribution);
        db.getCollection(”author”).update(authorQuery, author);
    }
```

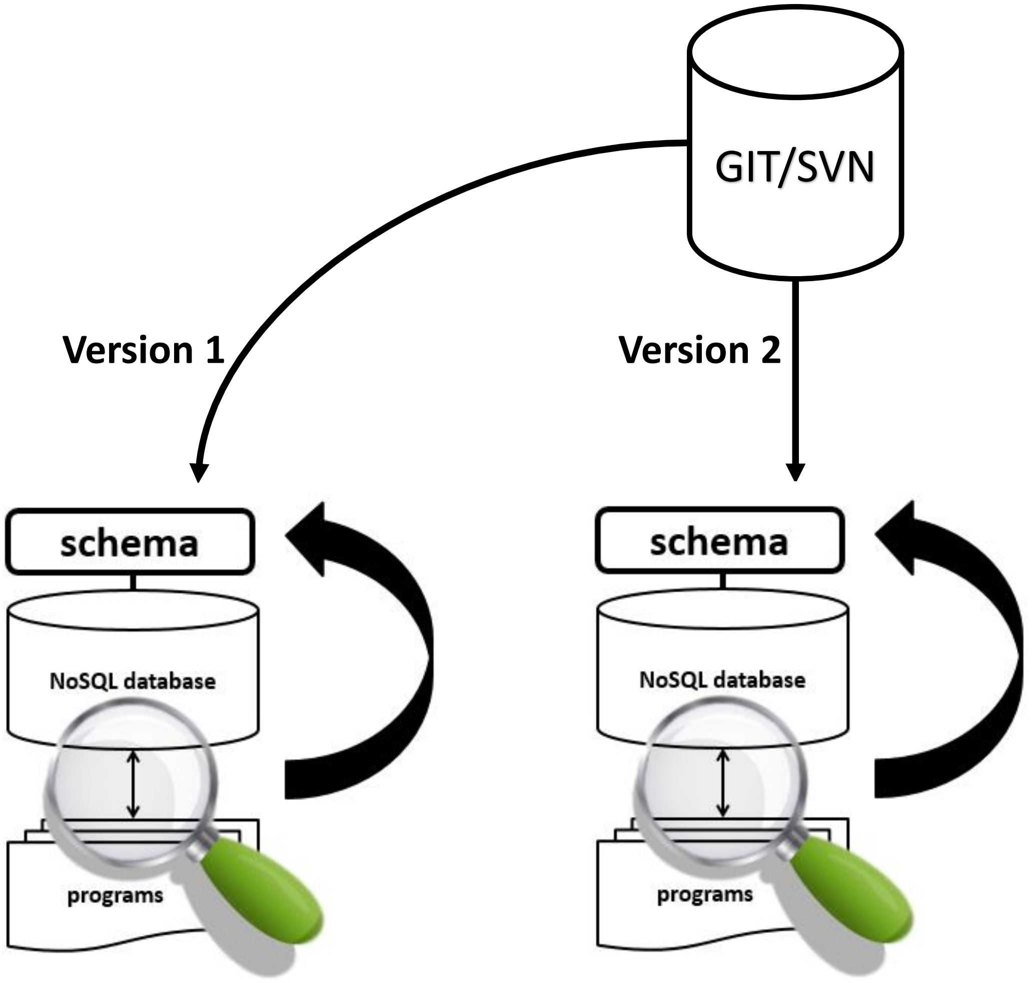
Extracting NoSQL Database Schema



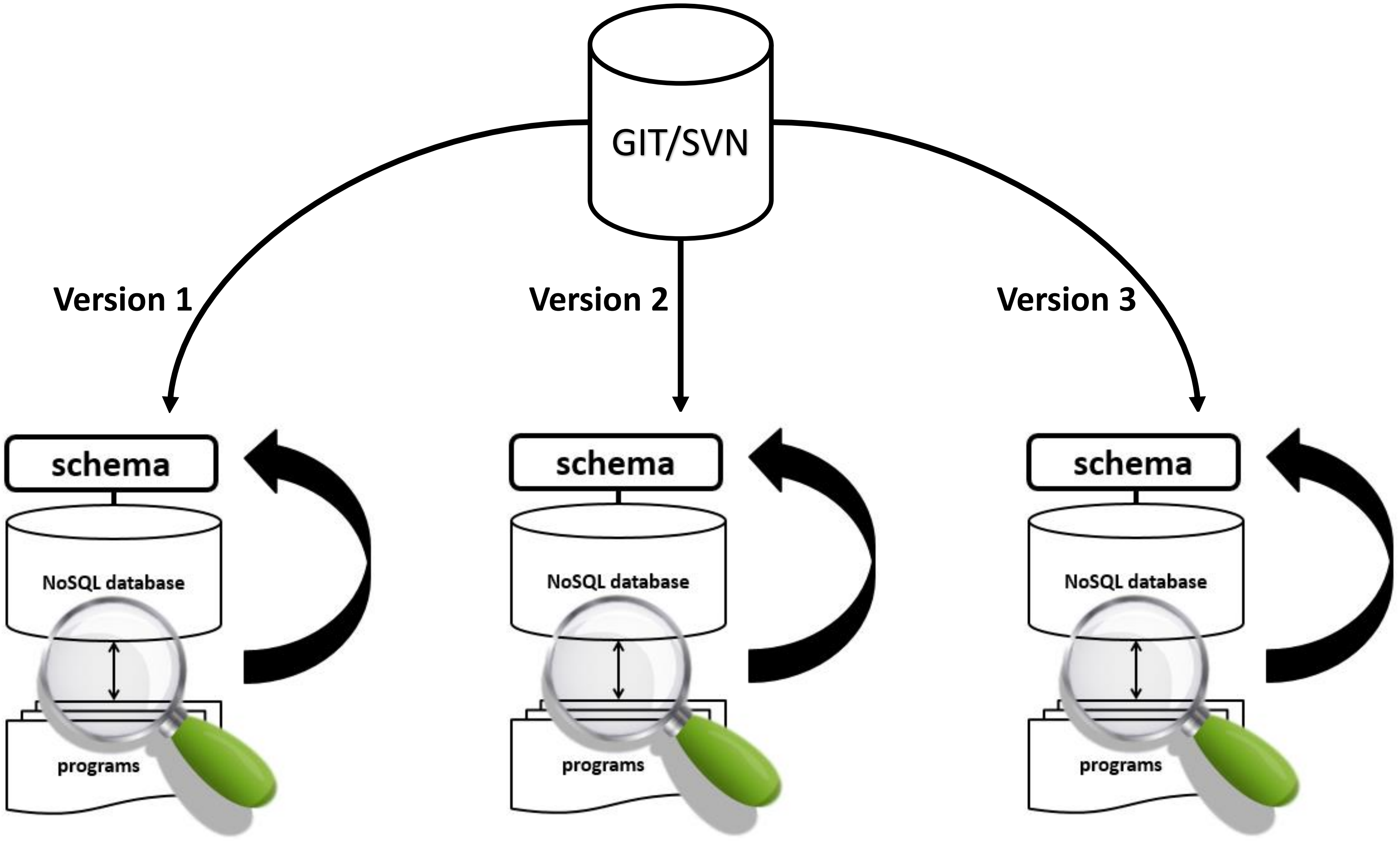
Extracting NoSQL Database SchemaS



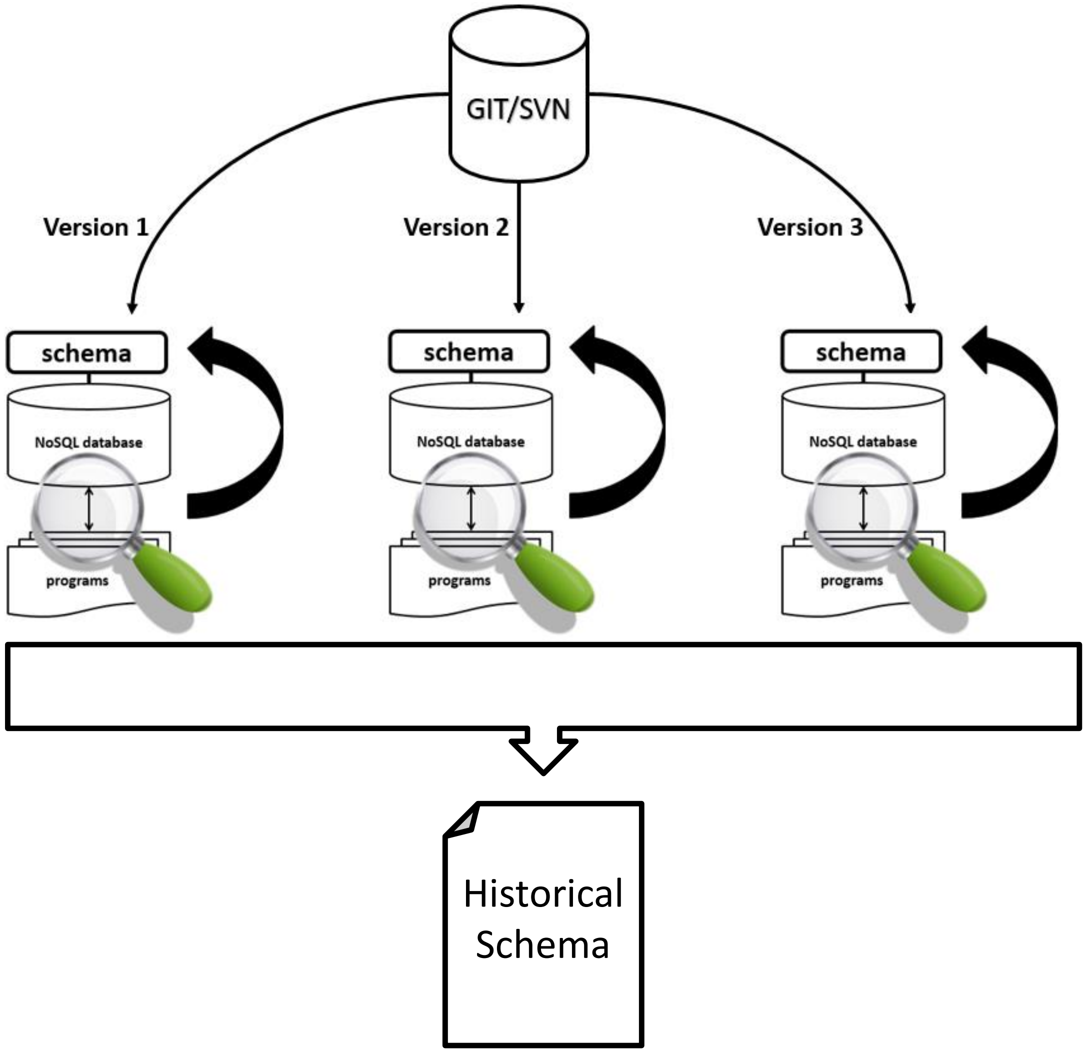
Extracting NoSQL Database SchemaS



Extracting NoSQL Database Schema<u>S</u>



Extracting NoSQL Database SchemaS



A Subject System

Tilos 23

A Subject System

Non-Profit Hungarian Radio

A Subject System

Non-Profit Hungarian Radio
Java + MongoDB

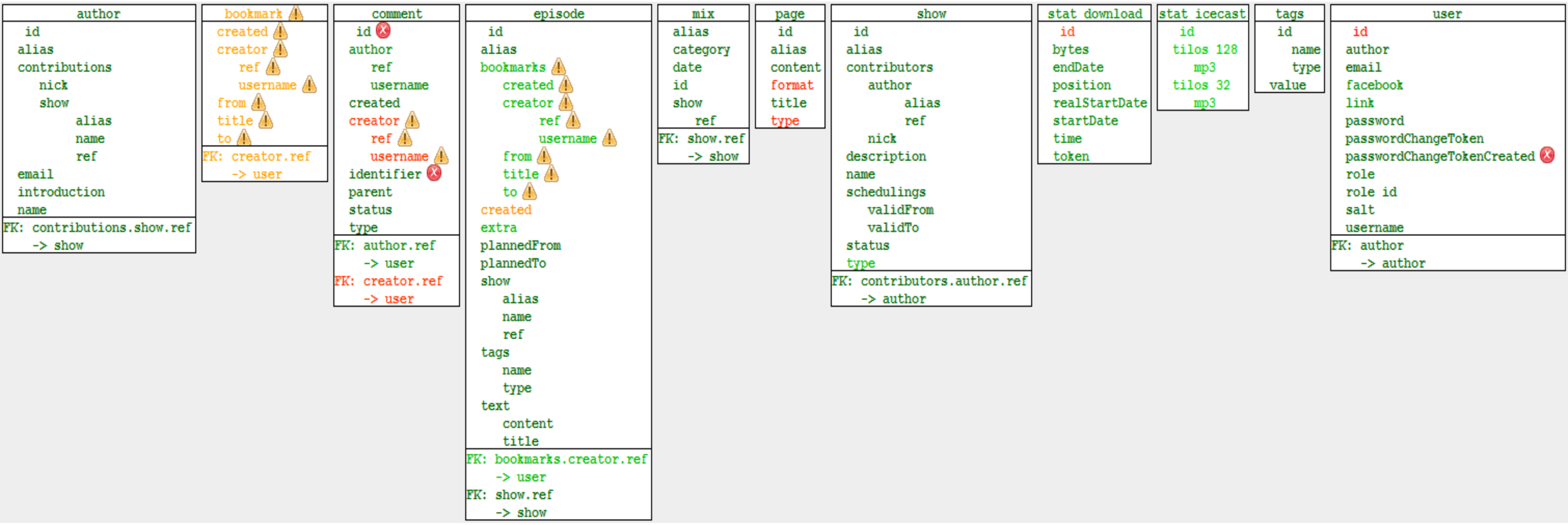
A Subject System

Non-Profit Hungarian Radio

Java + MongoDB

Two-Year History (303 versions)

Extracting Historical Schema



```
author
 id
 alias
 contributions
   nick
   show
     alias
     name
     ref
 email
 introduction
 name
FK: contributions.show.ref
  -> show
```

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

```
comment
 id ✖
 author
   ref
   username
 created
 creator ⚠
   ref ⚠
   username ⚠
 identifier ✖
 parent
 status
 type
FK: author.ref
  -> user
FK: creator.ref
  -> user
```

```
episode
 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
```

```
mix
 alias
 category
 date
 id
 show
   ref
FK: show.ref
  -> show
```

```
page
 id
 alias
 content
 format
 title
 type
```

```
show
 id
 alias
 contributors
   author
     alias
     ref
   nick
 description
 name
 schedulings
   validFrom
   validTo
 status
 type
FK: contributors.author.ref
  -> author
```

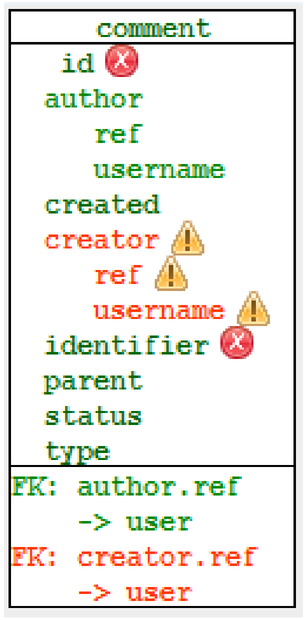
```
stat download
 id
 bytes
 endDate
 position
 realStartDate
 startDate
 time
 token
```

```
stat icecast
 id
 tilos 128
   mp3
 tilos 32
   mp3
```

```
tags
 id
   name
   type
 value
```

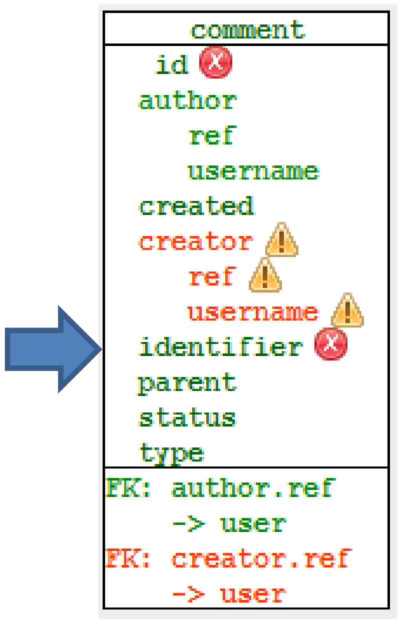
```
user
 id
 author
 email
 facebook
 link
 password
 passwordChangeToken
 passwordChangeTokenCreated ✖
 role
 role id
 salt
 username
FK: author
  -> author
```

Type Mismatch Detection



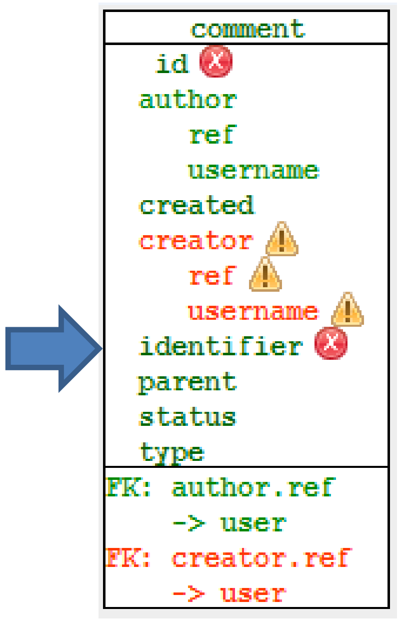
```
        comment
   id ✗
  author
     ref
     username
  created
  creator ⚠
     ref ⚠
     username ⚠
  identifier ✗
  parent
  status
  type
FK: author.ref
    -> user
FK: creator.ref
    -> user
```

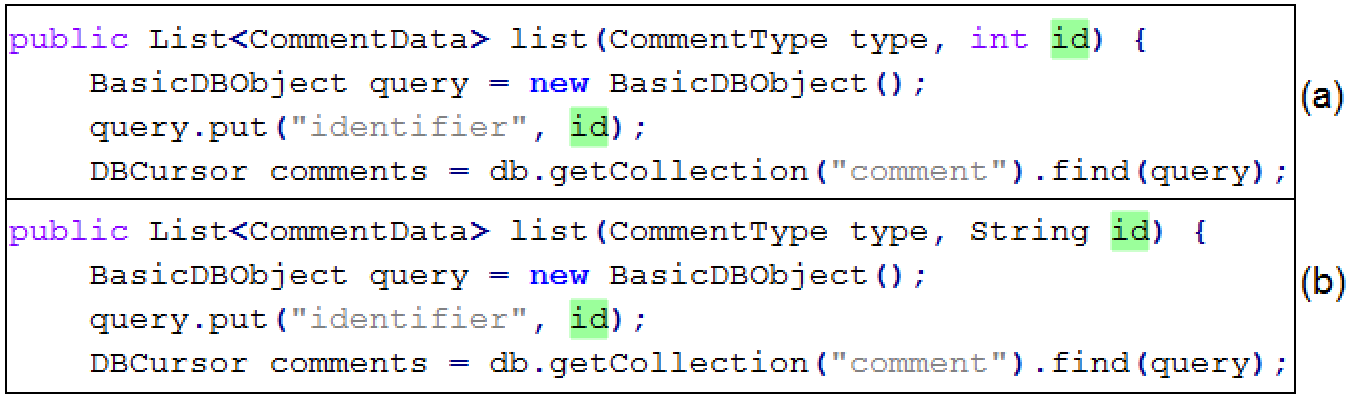
Type Mismatch Detection



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

Type Mismatch Detection

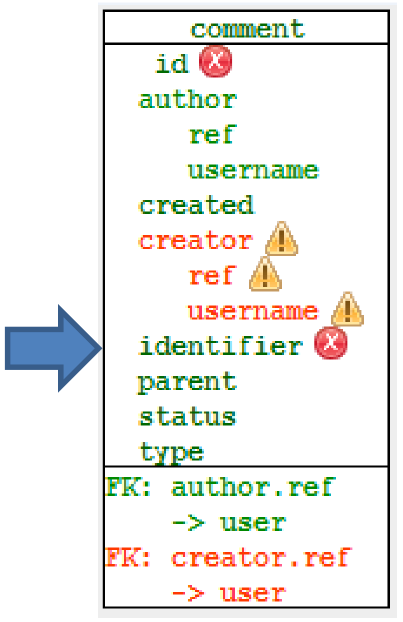




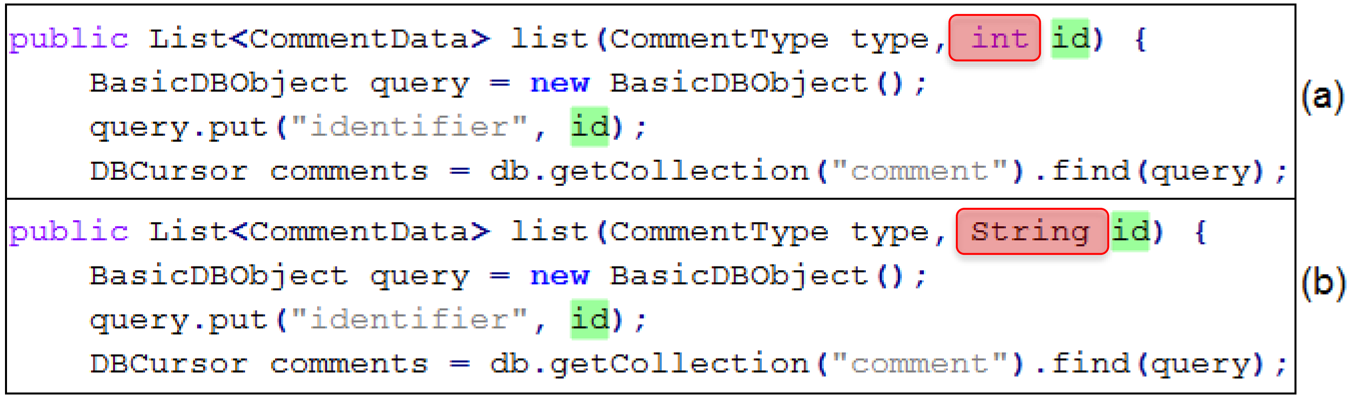
```
public List<CommentData> list(CommentType type, int id) {
    BasicDBObject query = new BasicDBObject();
    query.put("identifier", id);
    DBCursor comments = db.getCollection("comment").find(query);
```
(a)

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

Type Mismatch Detection



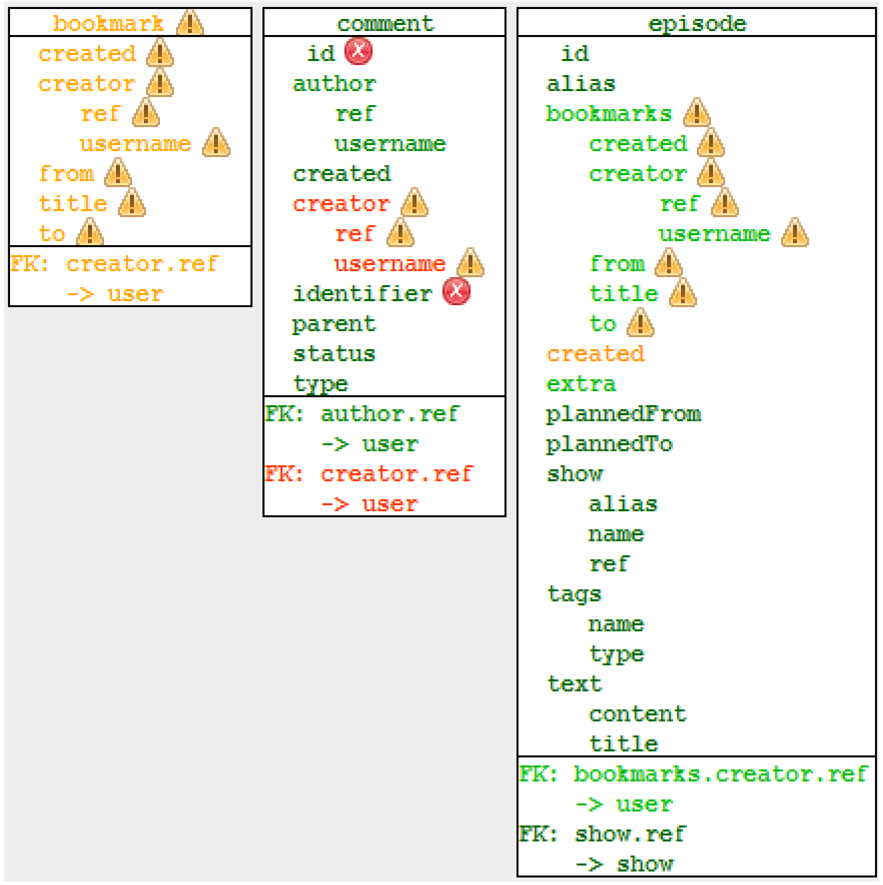
```
comment
  id
 author
    ref
    username
 created
 creator
    ref
    username
 identifier
 parent
 status
 type
FK: author.ref
    -> user
FK: creator.ref
    -> user
```



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

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

Renaming Detection

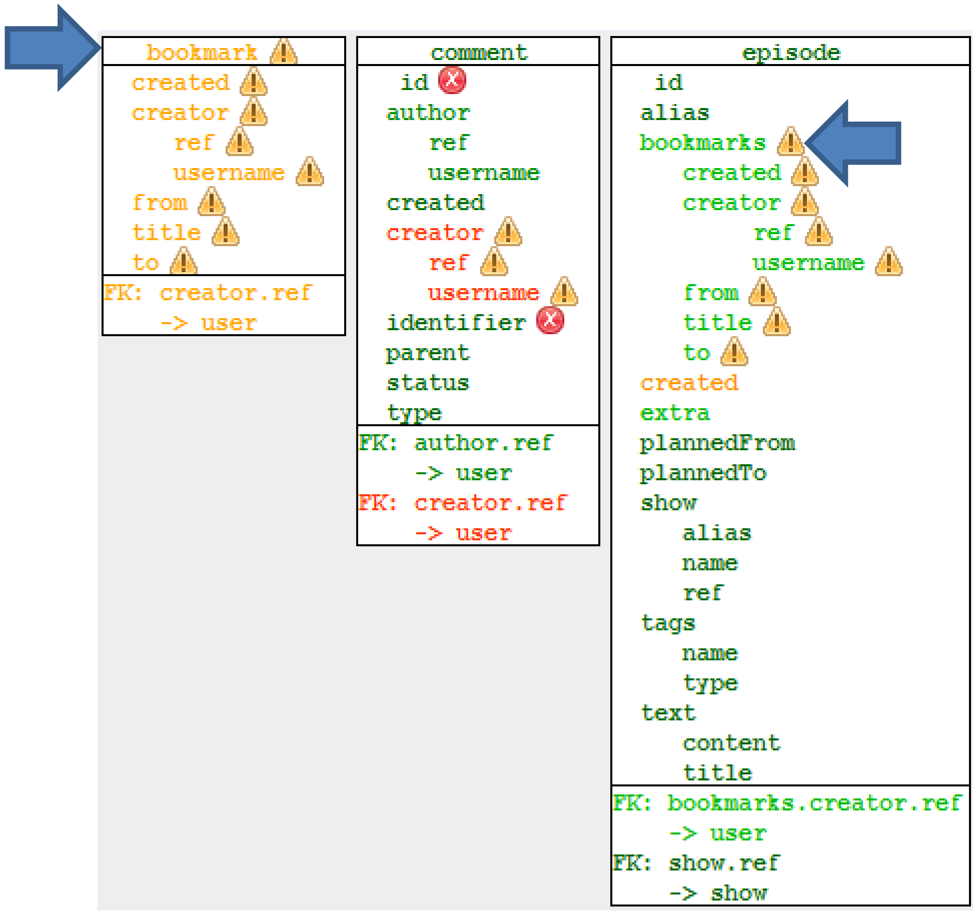


```
bookmark ⚠
  created ⚠
  creator ⚠
     ref ⚠
     username ⚠
  from ⚠
  title ⚠
  to ⚠
FK: creator.ref
    -> user
```

```
comment
  id ✖
  author
     ref
     username
  created
  creator ⚠
     ref ⚠
     username ⚠
  identifier ✖
  parent
  status
  type
FK: author.ref
    -> user
FK: creator.ref
    -> user
```

```
episode
  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

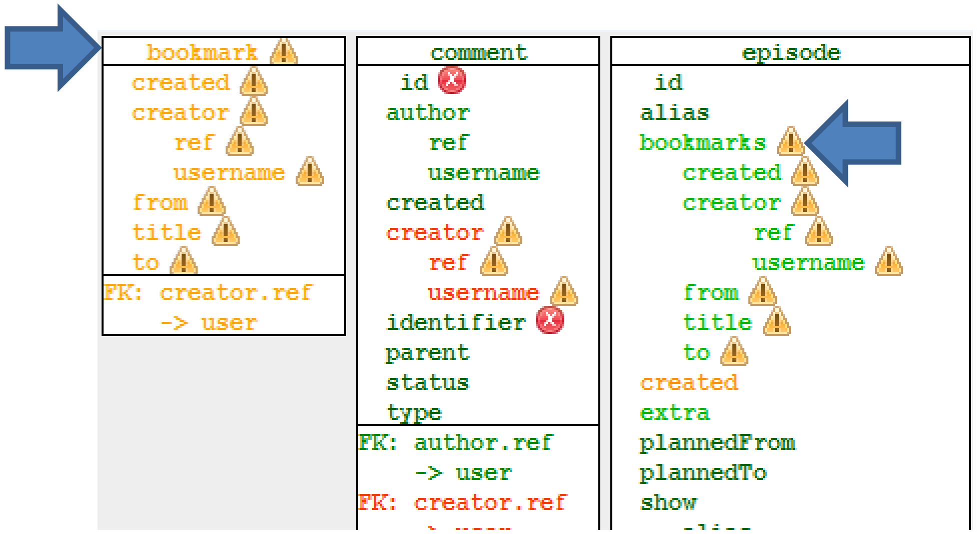


```
bookmark ⚠
  created ⚠
  creator ⚠
     ref ⚠
     username ⚠
  from ⚠
  title ⚠
  to ⚠
FK: creator.ref
    -> user
```

```
comment
  id ⊗
  author
     ref
     username
  created
  creator ⚠
     ref ⚠
     username ⚠
  identifier ⊗
  parent
  status
  type
FK: author.ref
    -> user
FK: creator.ref
    -> user
```

```
episode
  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
```

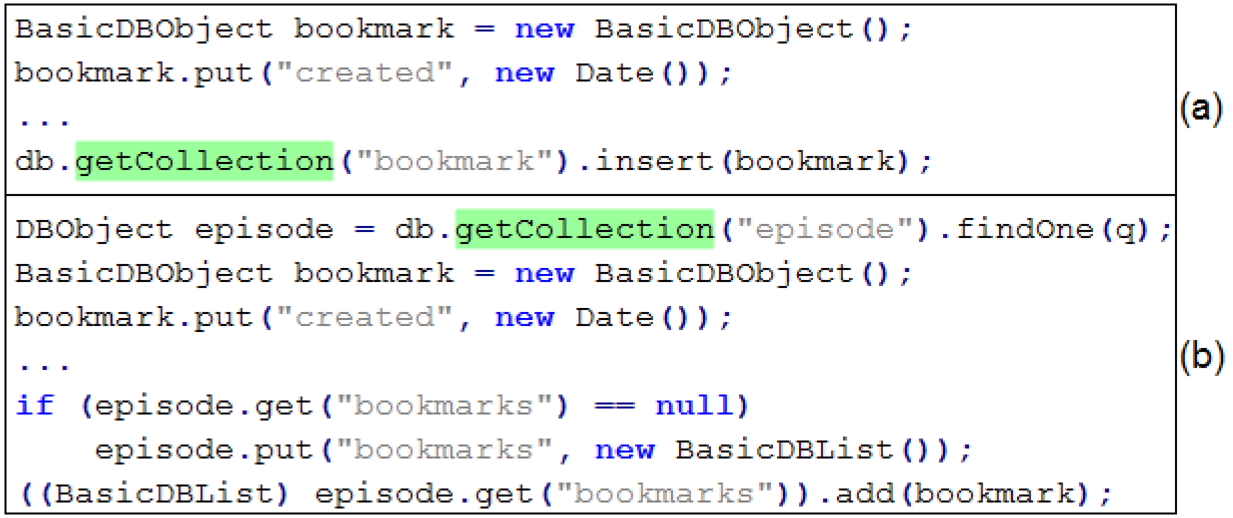
Renaming Detection



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

```
comment
  id
  author
     ref
     username
  created
  creator
     ref
     username
  identifier
  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
```



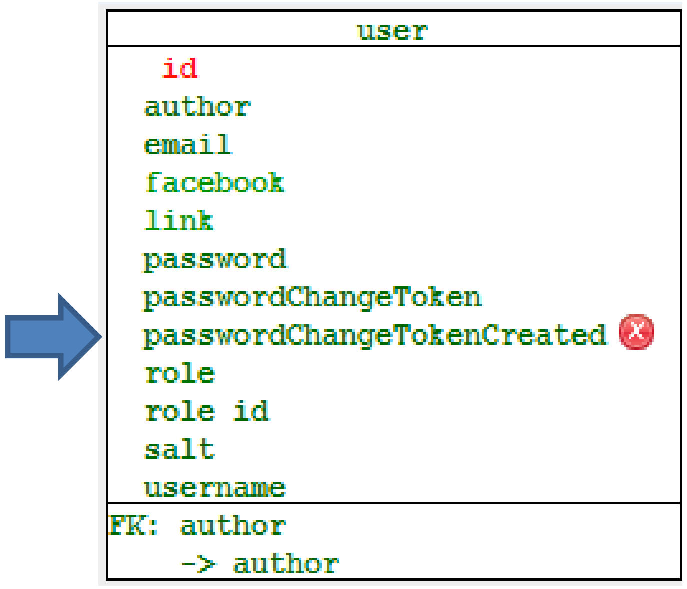
```
BasicDBObject bookmark = new BasicDBObject();
bookmark.put("created", new Date());
...
db.getCollection("bookmark").insert(bookmark);
```

(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);
```

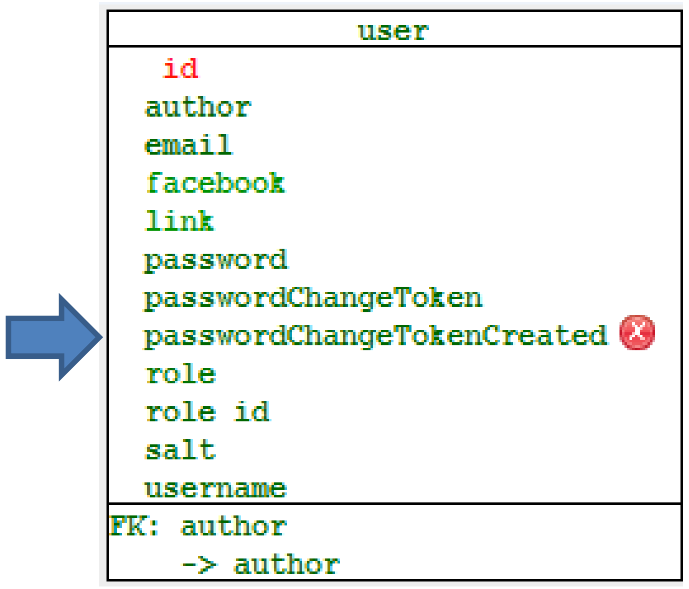
(b)

Data Loss Detection

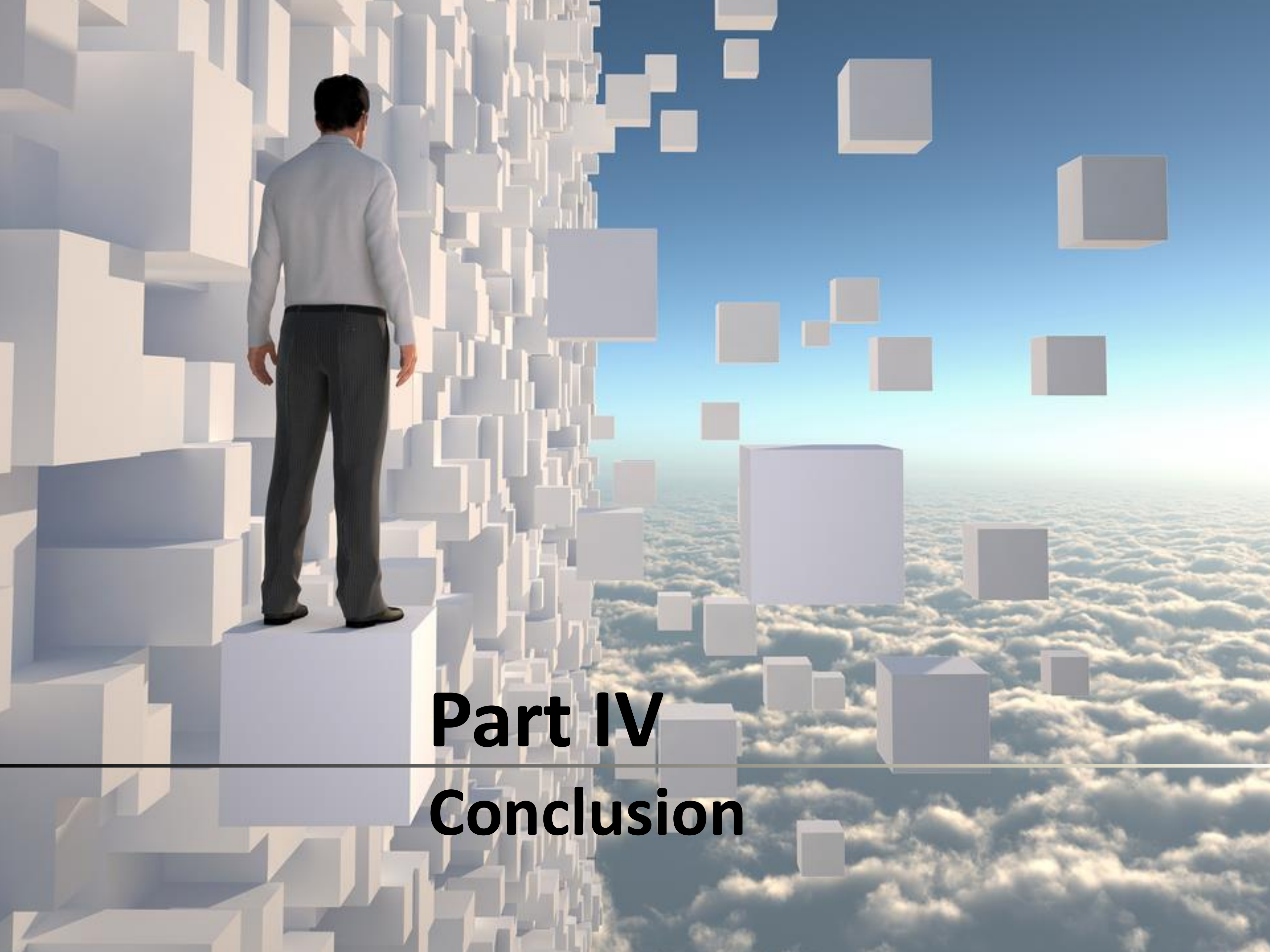


```
user
  id
 author
 email
 facebook
 link
 password
 passwordChangeToken
 passwordChangeTokenCreated
 role
 role id
 salt
 username
FK: author
    -> author
```

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);
```



Part IV

Conclusion

Summary

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

Summary

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

… designed to be applied to the whole system history

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

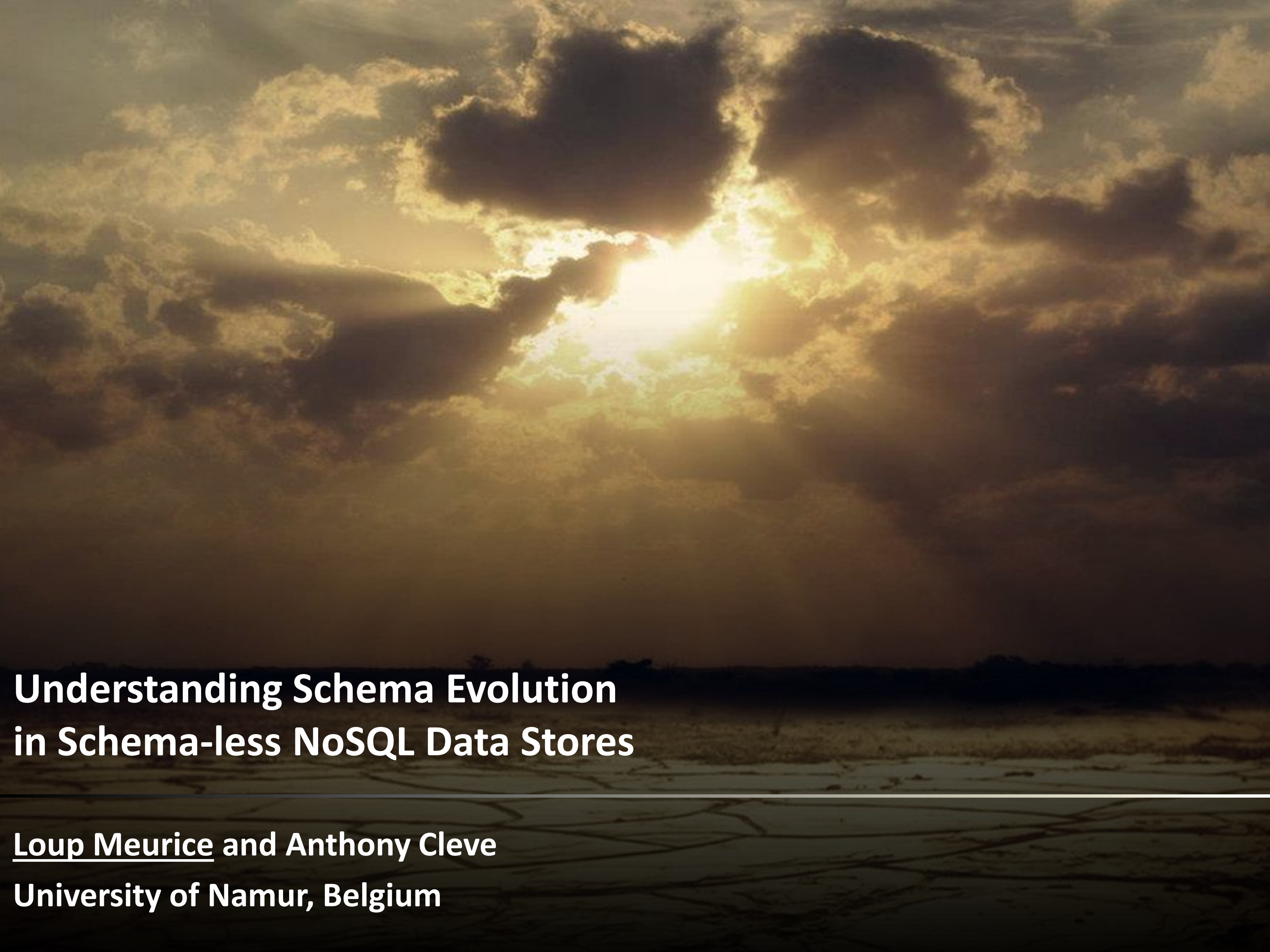
Summary

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

… currenlty designed for Java systems using MongoDB



Understanding Schema Evolution in Schema-less NoSQL Data Stores

Loup Meurice and Anthony Cleve

University of Namur, Belgium