Université de Rennes

cnrs UMR IRISA

Big Data
MongoDB

Laurent d'Orazio

Univ Rennes, CNRS, IRISA

2024-2025

Running example

num_student	lastname	firstname	login	group	grade1	grade2
10001023	TERIEUR	Alain	terieuar	1a	16.0	11.0
10001074	FONFEC	Sophie	fonfecsr	2b	17.0	12.5
10001184	TRAIBIEN	Samira	traibisr	2b	14.0	14.0
10001456	VRANTE	Héléna	vrantehr	1b	2.0	18.0
10002365	ZOUDANLKOU	Debbie	zoudandr	1a	15.0	7.5
10002578	PATAMOB	Adhémar	patamovr	1a	8.0	18.0
10002937	KAECOUTE	Xavier	kaecouxr	3b	13.5	14.0
10004235	ENFAILLITE	Mélusine	enfailmr	1b	12.0	14.0
10004589	KUZBIDON	Alex	kuzbidar	1a	16.0	10.5
10005556	AMARTAKALDIRE	Quentin	amertaqr	2a	8.5	10.0
10005621	BLAIREUR	Terry	blairetr	3a	9.0	6.0
10005644	ODLAVIEILLE	Pacôme	odlavepr	2a	13.5	9.0
10005761	TERIEUR	Alex	terialr	3b	8.0	16.5
10007410	FAIRANT	Teddy	fairantr	3a	12.5	0.0
10007526	HOTDEUGOU	Olaf	hotdeuor	2a	5.5	16.5
10007899	XELAIR	Jacques	xelairjr	1b	11.0	12.0
10008522	ARDELPIC	Helmut	ardelphr	1a	9.5	16.5
10009898	AJERRE	Tex	ajerretr	3b	9.5	15.5

Outline

I. Data model
II. Language

Outline

I. Data model

II. Language

Data model

- BSON
  - Binary JSON

Outline

I. Data model

II. Languague

A. DDL

B. DML

A. Data Definition Language (DDL)

- `db`
- `use`
- `show`
- `dropDatabase`
- `createCollection`
- `createView`
- `drop`
- `update`

db

- Objective
  - Report the name of the current DB

- Syntax
  - `db`

use

- Objective
  - Change the DB
- Syntax
  - `use <db name>`
- Exemple
  - `use university`

`show (dbs)`

- Objective
  - List all databases

- Syntax
  - `show dbs`

dropDatabase

- Objective
  - Delete the current DB
- Syntax
  - `<db name>.dropDatabase`
- Exemple
  - `db.dropDatabase()`

show (collections)

- Objective
  - List all tables

- Syntax
  - `show collections`

createCollection

- Objective
  - Create a collection
- Syntax
  - `db.createCollection(<name>,<options>)`
- Example
  - Create a collection person limited to 10 documents and 100000 bytes
  - `db.createCollection("student",{"capped":true,"size":100000,"max":10})`

Create (DB and collections)

- N.B.
  - Create instruction is optional
  - Databases and collections created after a first reference

drop

- Objective
  - Delete a table
- Syntax
  - `db.<table_name>.drop()`
- Example
  - Removing student table
  - `db.student.drop()`

update

- Objective
  - Update a table
    - Rename
    - Add a column
    - Replace a column

renameCollection

- Objective
  - Rename a collection
- Syntax
  - `db.<oldname>.renameCollection("<newname>")`
- Example
  - Rename table student into person
  - `db.student.renameCollection("person")`

`update` – Add column

- Objective
  - Add an attribute in a collection

- Syntax
  - `db.<table_name>.update(<query>, {$set:{<attribute_name>:value}},{multi:true})`

- Example
  - Add a column year (with value 2023) in table student
  - `db.student.update({},{$set:{year:2023}},{multi:true})`

`update` – Drop column

- Syntax
  - `unset db.<table_name>.update({},{$unset:{<attribute_name>:""}})`

- Example
  - Remove address in table student
  - `db.student.update({},{$unset:{address:""}})`

`update` – Rename

- Objective
  - Change field name
- Syntax
  - `unset db.<table_name>.update({},{$rename:{<old_att_name>:"<new_att_name>"}})`
- Example
  - Replace address in table student by city
  - `db.student.update({},{$rename:{address:"city"}})`

B. Data Manipulation Language (DML)
Data management

- DML
  - Data management
    - Insert
    - Delete
    - Update
  - Querying

insert

- Objective
  - Insert a document

- Syntax
  - `db.<collection_name>.insert({<attribut1>:<value1>, ..., <attributN>:<valueN>})`

- Example
  - Insert student Jean Dupont with number 1 into collection students
  - `db.student.insert({lastname:"Dupont",firstname:"Jean",num_student:1})`

remove

- Objective
  - Remove a document
- Syntax
  - `db.<collection_name>.remove({<condition>})`
- Example
  - Remove student Dupont into collection students
  - `db.student.remove({lastname:"Dupont"})`

update

- Objective
  - Update a document
- Syntax (example `$set`)
  - `db.<collection_name>.update({<condition>},$set:{<field1>:<value1>,...})`
- Example
  - Change name of student Dupont by Durand into collection student
  - `db.student.update({lastname: "Dupont" }, { $set: { lastname: "Durant" }})`

B. Data Manipulation Language (DML) Querying

- DML
  - Data management
  - Querying
    - Select
    - Project
    - Left outer Join
    - Group by

find

- Objective
  - Select documents and project fields

- Syntax
  - `db.collection.find(<query>, <projection>)`

- Examples
  - Student living in Lannion
    - `db.student.find({lastname:"Durant"})`
  - Name of students
    - `db.student.find({},{lastname:true})`
  - Name of students living in Lannion
    - `db.student.find({lastname:"Durant"},{lastname:true})`

Comparison query operators

- Operators
  - `$eq`: Matches values that are equal to a specified value
  - `$gt`: Matches values that are greater than a specified value
  - `$gte`: Matches values that are greater than or equal to a specified value
  - `$lt`: Matches values that are less than a specified value
  - `$ne`: Matches all values that are not equal to a specified value.
  - Etc.
- Syntax
  - `{ <field>: { $eq: <value> } }`
- Example
  - Student not living in Lannion
    - `db.student.find({lastname:{ $ne: "Durant" }})`

find or

- Syntax
  - db.collection.find({$or:[<query$_1$>, <query$_2$>, ..., <query$_n$>]}, <projection>)

- Example
  - Students with grade1 greater than 10 or grade2 greater than 10
    - db.student.find({$or:[{grade1:{$gt:10}},{grade2:{$gt:10}}]})

find and (1)

- Syntax
  - db.collection.find({$and:[<query$_1$>, <query$_2$>, ..., <query$_n$>]}, <projection>)

- Example
  - Students with grade1 greater than 10 or grade2 greater than 10
    - db.student.find({$and:[{grade1:{$gt:10}},{grade2:{$gt:10}}]})

find and (2)

- Note
  - `and` can be implicit

- Syntaxes
  - `db.collection.find({<query1>, <query2>, ..., <queryn>}, <projection>)`

- Example
  - Students with grade1 greater than 14 and grade1 less than 16
    - `db.student.find({ grade1: { $gt: 14, $lt: 16 } })`
    - `db.student.find({ grade1: { $gt: 14 }, grade1: { $lt: 16 }})`
  - Students with grade1 greater than 14 and grade12 greater than 12
    - `db.student.find({ grade1: { $gt: 12 }, grade2: { $gt: 12 }})`

limit

- Objective
  - Limit the number of documents on a cursor
- Syntax
  - `db.collection.find(<condition>).limit(<nb>)`
- Example
  - 2 first students
    - `db.student.find().limit(2)`

distinct

- Objective
  - Returns the distinct values for a given fields in a cursor
- Syntax
  - `db.collection.distinct("<field>")`
- Example
  - Distinct firstnames of students
    - `db.student.distinct("firstname")`

aggregate

- Objective
  - Process multiple documents
    - Group values from multiple documents together
    - Perform operations on the grouped data to return a single result
    - Analyze data changes over time

- Syntax
  - `db.collection.aggregate([<stage1>, …, <stagen>])`
  - With `stagei` being `$sort`, `$group`, `$match`, etc.

- Example

```
db.student.aggregate(
    [
      { $sort : { lastname : -1 } }
    ]
)
```

aggregate
sort

- Objective
  - Sort documents

- Syntax
  - `{ $sort: { <field1>: <sort order>*, <field2>: <sort order>* ... } }`

*1 for asc, -1 for desc

- Example

```
db.student.aggregate(
    [
      { $sort : { name : -1 } }
    ]
)
```

aggregate group

- Objective
  - Separate documents into groups according to a group key

- Syntax

```
{$group:{
    id: <expression>, // Group key
            <field1>: { <accumulator1> : <expression1> },
            ...
  }
 }
```

- Example

```
{$group: { _id: "$group", avggrade1: { $avg: "$grade1" } } }
```

aggregate
match

- Objective
  - Filter documents according the specified condition(s)

- Syntax
  - `{ $match: { <query> } }`

- Example

```
db.student.aggregate([{$match:{group:"1a"}}])
```

aggregate project

- Objective
  - Pass documents with the requested fields
- Syntax
  - `{ $project: { <specification(s)> } }`
- Example

```
db.student.aggregate([{$project:{lastname:1}}])
```

Bibliography

- www.mongodb.com