MongoDB

SQL and MongoDB

SQL	MongoDB
Database	Database
Table	Collection
Index	Index
Row	Document
Column	Field
JOIN	Linking & Embedding

DATA MODEL

- Stores data in form of **BSON (Binary JavaScript Object Notation)** documents:

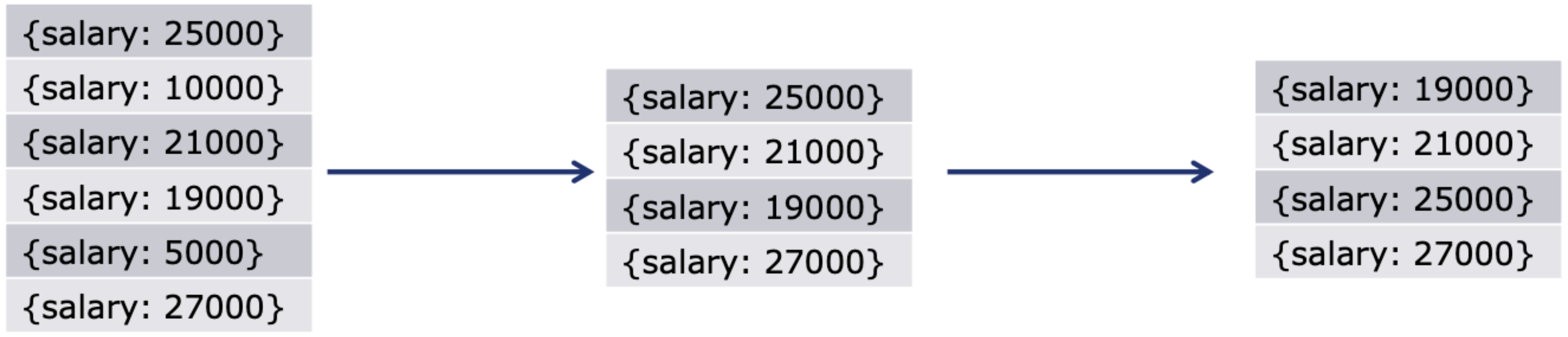
```
{
   name: "travis",
   salary: 24000,
   designation: "Computer Scientist",
   teams: ["front-end", "database"]
}
```

- A Group of related documents with a shared common index is a **collection**

Typical Query

- Query all employee names with salary greater than 18000 sorted in ascending order
- `db.employee.find({salary:{$gt:18000},{name:1}}).sort({salary:1})`

  Collection *Condition* *Projection* *Modifier*



INSERT

- Insert a row entry for new employee "Sally"

```
db.employee.insert({
    name: "Sally",
    salary: 24000,
    designation: "Graphical Designer",
    teams: ["front-end"]
})
```

UPDATE

- All employees with salary greater than 18000 get a designation of "Manager"

```
db.employee.update(
    {salary: {$gt: 18000}},
    {$set: {designation: "Manager"}},
    {multi: true}
)
```

Update Criteria

Update Action

Update Option

DELETE

- Remove all employees who earn less than 10000

```
db.employee.remove(
    {salary: {$lt: 10000}}    Remove Criteria
)
```

Considerations for Schema Design

- Design your schema according to user requirements
- Combine objects into one document if you will use them together, otherwise separate them
    - Make sure there should not be the need of joins)
- Duplicate the data (but limited), because disk space is rather cheap compared to compute time
- Optimize your schema for most frequent use-cases (reads)
    - Joins on writes, not reads

Try yourself

- Start MongoDB in a docker container:
  - `docker run -d -p 27017:27017 --name some-mongo mongo:latest`
- Connect to it and start the mongo CLI:
  - `docker exec -it some-mongo mongosh`
- Try the following commands:
  - `show dbs`
  - `use local`
  - `show collections`
  - `exit`