

### Schéma relationnel de la base

**FONCTION** (Numfonct, libfonct, salaire)

**SERVICE** (Numserv, libserv)

**EMPLOYE** (Numemp, nom, prenom, #idFonction, #idService)

### Travail à réaliser (NoSQL)

1. Afficher par ordre alphabétique du nom et prenom, les employés de l'entreprise.

```
db.employe.find().sort({nom: 1, prenom:
```

2. Afficher par ordre alphabétique les fonctions de l'entreprise.

```
db.employe.find[
  {},
  {"fonction.libfonct": 1}
].sort({"fonction.libfonct": 1})
```

```
//sans doublon
db.employe.distinct("fonction.libfonct")
```

```
//sans doublon trié
db.employe.aggregate([
  { $group: { _id: "$fonction.libfonct" } },
  { $sort: { _id: 1 } }
]);
```

3. Afficher pour chaque employé, sa fonction et son service.

```
db.employe.find[
  {},
  {nom: 1,
  prenom: 1,
  "fonction.libfonct": 1,
  "service.libserv" : 1
}
]
```

```
//sans sous collection et sans id
db.employe.find[
  {},
  {_id : 0, nom: 1, prenom: 1, fonction : "$fonction.libfonct", service : "$service.libserv"}
]
```

4. Afficher le nom et prénom des personnes qui sont « Chef de Projet ».

```
db.employe.find({"fonction.libfonct": "Manager"},{nom:1, prenom:1})
```

5. Afficher les employés qui ont la même fonction que Durand, Paul.

```
var f_durand = db.employe.findOne({
  $and: [
    { nom: "Durand" },
    { prenom: "Paul" }
  ]
});

db.employe.find({"fonction.libfonct":f_durand.fonction.libfonct})
```

6. Afficher le salaire max pour chaque service

```
db.employe.aggregate([
  {
    $group: {
      _id: "$service.libserv",      // on groupe par service
      salaireMax: { $max: "$fonction.salaire" } // salaire maximum
    }
  }
]);
```

7. Afficher les employés qui sont Manager et Développeur

```
db.employe.find({
  $or: [
    { "fonction.libfonct": "Manager" },
    { "fonction.libfonct": "Développeur" }
  ]
});
```

8. Modifier le prenom de Luc Bernard par Lucio

```
db.employe.updateOne(
  { nom: "Bernard", prenom: "Luc" },
  { $set: { prenom: "Lucio" } }
);
```