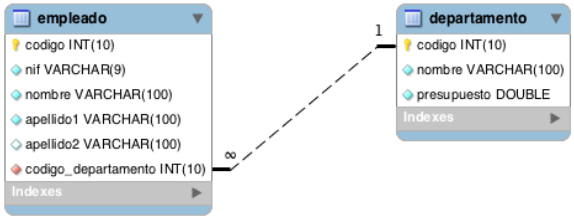
Gestión de empleados

Modelo entidad/relación



Base de datos para MySQL

```sql
DROP DATABASE IF EXISTS empleados;
CREATE DATABASE empleados CHARACTER SET utf8mb4;
USE empleados;

CREATE TABLE departamento (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  presupuesto DOUBLE UNSIGNED NOT NULL,
  gastos DOUBLE UNSIGNED NOT NULL
);

CREATE TABLE empleado (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nif VARCHAR(9) NOT NULL UNIQUE,
  nombre VARCHAR(100) NOT NULL,
  apellido1 VARCHAR(100) NOT NULL,
  apellido2 VARCHAR(100),
  id_departamento INT UNSIGNED,
  FOREIGN KEY (id_departamento) REFERENCES departamento(id)
);

INSERT INTO departamento VALUES(1, 'Desarrollo', 120000, 6000);
INSERT INTO departamento VALUES(2, 'Sistemas', 150000, 21000);
INSERT INTO departamento VALUES(3, 'Recursos Humanos', 280000, 25000);
INSERT INTO departamento VALUES(4, 'Contabilidad', 110000, 3000);
INSERT INTO departamento VALUES(5, 'I+D', 375000, 380000);
INSERT INTO departamento VALUES(6, 'Proyectos', 0, 0);
INSERT INTO departamento VALUES(7, 'Publicidad', 0, 1000);


INSERT INTO empleado VALUES(1, '32481596F', 'Aarón', 'Rivero', 'Gómez', 1);
INSERT INTO empleado VALUES(2, 'Y5575632D', 'Adela', 'Salas', 'Díaz', 2);
INSERT INTO empleado VALUES(3, 'R6970642B', 'Adolfo', 'Rubio', 'Flores', 3);
INSERT INTO empleado VALUES(4, '77705545E', 'Adrián', 'Suárez', NULL, 4);
INSERT INTO empleado VALUES(5, '17087203C', 'Marcos', 'Loyola', 'Méndez', 5);
INSERT INTO empleado VALUES(6, '38382980M', 'María', 'Santana', 'Moreno', 1);
INSERT INTO empleado VALUES(7, '80576669X', 'Pilar', 'Ruiz', NULL, 2);
INSERT INTO empleado VALUES(8, '71651431Z', 'Pepe', 'Ruiz', 'Santana', 3);
INSERT INTO empleado VALUES(9, '56399183D', 'Juan', 'Gómez', 'López', 2);
INSERT INTO empleado VALUES(10, '46384486H', 'Diego','Flores', 'Salas', 5);
INSERT INTO empleado VALUES(11, '67389283A', 'Marta','Herrera', 'Gil', 1);
INSERT INTO empleado VALUES(12, '41234836R', 'Irene','Salas', 'Flores', NULL);
INSERT INTO empleado VALUES(13, '82635162B', 'Juan Antonio','Sáez', 'Guerrero', NULL);
```

Creando las colecciones en mongoDB

1: colección empleado

db.empleado.insertMany([

{_id:1,

nif: “32481596F” ,

nombre: “Aarón” ,

apellido1: “Rivero” ,

apellido2: “Gómez” ,

id_departamento: 1 },

{_id:2,

nif: “Y5575632D” ,

nombre: “Adela” ,

apellido1: “Salas” ,

apellido2: “Díaz” ,

id_departamento: 2 },

{_id:3,

nif: “R6970642B” ,

nombre: “Adolfo” ,

apellido1: “Rubio” ,

apellido2: “Flores” ,

id_departamento: 3 },

{_id:4 ,

nif: “77705545E” ,

nombre: “Adrián” ,

apellido1: “Suárez” ,

apellido2: null ,

id_departamento: 4 },

{_id:5 ,

nif: “17087203C” ,

nombre: “Marcos” ,

apellido1: “Loyola” ,

apellido2: “Méndez” ,

id_departamento: 5 },

{_id: 6 ,

nif: “38382980M” ,

nombre: “María” ,

apellido1: “Santana” ,

apellido2: “Moreno” ,

id_departamento: 1 },

{_id: 7 ,

nif: “80576669X” ,

nombre: “Pilar” ,

apellido1: “Ruiz” ,

apellido2: null ,

id_departamento: 2 },

{_id: 8 ,

nif: “71651431Z” ,

nombre: “Pepe” ,

apellido1: “Ruiz” ,

apellido2: “Santana” ,

id_departamento: 3 },

{_id: 9 ,

nif: “56399183D” ,

nombre: “Juan” ,

apellido1: “Gómez” ,

apellido2: “López” ,

id_departamento: 2 },

{_id: 10 ,

nif: “46384486H” ,

nombre: “Diego” ,

apellido1: “Flores” ,

apellido2: “Salas” ,

id_departamento: 5 },

{_id: 11 ,

nif: “67389283A” ,

nombre: “Marta" ,

apellido1: “Herrera” ,

apellido2: “Gil” ,

id_departamento: 1 },

{_id: 12 ,

nif: “41234836R” ,

nombre: “Irene” ,

apellido1: “Salas” ,

apellido2: “Flores” ,

id_departamento: null },

{_id: 13 ,

nif: “82635162B” ,

nombre: “Juan Antonio” ,

apellido1: “Sáez” ,

apellido2: “Guerrero” ,

id_departamento: null }

])

2: Colección departamento

db.departamento.insertMany([

{_id:1,

nombre: “Desarrollo” ,

presupuesto: 120000 ,

gastos: 6000 },

{_id:2,

nombre: “Sistemas” ,

presupuesto: 150000 ,

gastos: 21000 },

{_id:3,

nombre: “Recursos Humanos” ,

presupuesto: 280000,

gastos: 25000 },

{_id:4,

nombre: "Contabilidad" ,

presupuesto: 110000 ,

gastos: 3000 },

{_id:5,

nombre: “I+D” ,

presupuesto: 375000 ,

gastos: 380000 },

{_id:6,

nombre: “Proyectos” ,

presupuesto: 0,

gastos: 0 },

{_id:7,

nombre: “Publicidad” ,

presupuesto: 0 ,

gastos: 1000},

])

Consultas sobre una tabla

1. Lista el primer apellido de todos los empleados.

   db.empleado.find({}, { _id: 0, apellido1: 1})

2. Lista el primer apellido de los empleados eliminando los apellidos que estén repetidos.

   db.empleado.distinct(“apellido1”)

3. Lista todas las columnas de la tabla `empleado`.
   db.empleado.find()

4. Lista el nombre y los apellidos de todos los empleados.

   db.empleado.find({} , { _id: 0, nombre: 1 , apellido1 :1 , apellido2: 1 })

5. Lista el identificador de los departamentos de los empleados que aparecen en la tabla `empleado`.

   db.empleado.find({} , { _id: 0, nif:1 })

6. Lista el identificador de los departamentos de los empleados que aparecen en la tabla `empleado`, eliminando los identificadores que aparecen repetidos.
   db.empleado.distinct(“id_departamento”)

7. Lista el nombre y apellidos de los empleados en una única columna.

```
db.empleado.aggregate([
 {
  $project: {
   _id :0 ,
   Nombre_Completo: {
    $concat: ["$nombre", " ", "$apellido1", " ", "$apellido2"]
   }
  }
 }
])
```

8. Lista el nombre y apellidos de los empleados en una única columna, convirtiendo todos los caracteres en mayúscula.

```
db.empleado.aggregate([
 {
  $project: {
   _id: 0,
   nombreCompletoEnMayuscula: {
    $toUpper: {
     $concat: ["$nombre", " ", "$apellido1", " ", "$apellido2"]
    }
   }
  }
 }
])
```

9. Lista el nombre y apellidos de los empleados en una única columna, convirtiendo todos los caracteres en minúscula.

```
db.empleado.aggregate([
 {
  $project: {
   _id: 0,
   nombreCompletoEnMinuscula: {
    $toLower: {
     $concat: ["$nombre", " ", "$apellido1", " ", "$apellido2"]
    }
   }
  }
 }
])
```

10. Lista el identificador de los empleados junto al nif, pero el nif deberá aparecer en dos columnas, una mostrará únicamente los dígitos del nif y la otra la letra.

```
db.empleado.aggregate([
 {
  $project: {
   _id: 1,
   digitosNIF: { $substr: ["$nif", 0, 8] }, // Los primeros 8 caracteres son los dígitos
   letraNIF: { $substr: ["$nif", 8, 1] } // El último carácter es la letra
  }
 }
])
```

11. Lista el nombre de cada departamento y el valor del presupuesto actual del que dispone. Para calcular este dato tendrá que restar al valor del presupuesto inicial (columna `presupuesto`) los gastos que se han generado (columna `gastos`). Tenga en cuenta que en algunos casos pueden existir valores negativos. Utilice un alias apropiado para la nueva columna columna que está calculando.

```
db.departamento.aggregate([
 {
  $project: {
   _id: 0,
   nombre: 1,
   presupuestoActual: { $subtract: ["$presupuesto", "$gastos"] }
  }
 }
])
```

12. Lista el nombre de los departamentos y el valor del presupuesto actual ordenado de forma ascendente.

```
db.departamento.find({},{_id:0,nombre:1 , presupuesto:1}).sort({presupuesto:1})
```

13. Lista el nombre de todos los departamentos ordenados de forma ascendente.

    db.departamento.find({},{_id:0,nombre:1}).sort({nombre:1})

14. Lista el nombre de todos los departamentos ordenados de forma descendente.

    db.departamento.find({},{_id:0,nombre:1}).sort({nombre:-1})

15. Lista los apellidos y el nombre de todos los empleados, ordenados de forma alfabética tendiendo en cuenta en primer lugar sus apellidos y luego su nombre.

    db.empleado.find({},{_id:0,nombre:1, apellido1:1 , apellido2:1}).sort({apellido1:1 , apellido2:1, nombre:1})

16. Devuelve una lista con el nombre y el presupuesto, de los 3 departamentos que tienen mayor presupuesto.

    db.departamento.find({},{_id:0,nombre:1 , presupuesto:1}).sort({presupuesto:-1}).limit(3)

17. Devuelve una lista con el nombre y el presupuesto, de los 3 departamentos que tienen menor presupuesto.

    db.departamento.find({},{_id:0,nombre:1 , presupuesto:1}).sort({presupuesto:1}).limit(3)

18. Devuelve una lista con el nombre y el gasto, de los 2 departamentos que tienen mayor gasto.

    db.departamento.find({},{_id:0,nombre:1 , gastos:1}).sort({gastos:-1}).limit(2)

19. Devuelve una lista con el nombre y el gasto, de los 2 departamentos que tienen menor gasto.

    db.departamento.find({},{_id:0,nombre:1 , gastos:1}).sort({gastos:1}).limit(2)

20. Devuelve una lista con 5 filas a partir de la tercera fila de la tabla `empleado`. La tercera fila se debe incluir en la respuesta. La respuesta debe incluir todas las columnas de la tabla `empleado`.

    db.departamento.find().skip(2).limit(5)

21. Devuelve una lista con el nombre de los departamentos y el presupuesto, de aquellos que tienen un presupuesto mayor o igual a 150000 euros.

    db.departamento.find({presupuesto: {$gte: 150000}},{_id:0,nombre:1 , presupuesto:1})

22. Devuelve una lista con el nombre de los departamentos y el gasto, de aquellos que tienen menos de 5000 euros de gastos.

    db.departamento.find({gastos: {$lte: 5000}},{_id:0,nombre:1 , gastos:1})

23. Devuelve una lista con el nombre de los departamentos y el presupuesto, de aquellos que tienen un presupuesto entre 100000 y 200000 euros. Sin utilizar el operador BETWEEN.

    db.departamento.find({presupuesto: {$gte: 100000, $lte:200000}},{_id:0,nombre:1 , presupuesto:1})

24. Devuelve una lista con el nombre de los departamentos que **no** tienen un presupuesto entre 100000 y 200000 euros. Sin utilizar el operador BETWEEN.

    db.departamento.find({
    $nor: [{ presupuesto: { $gte: 100000, $lte: 200000 } }
    ]} , { nombre: 1, _id: 0 })

25. Devuelve una lista con el nombre de los departamentos que tienen un presupuesto entre 100000 y 200000 euros. Utilizando el operador BETWEEN.

    db.departamento.find({presupuesto: {$gte: 100000, $lte:200000}},{_id:0,nombre:1})

26. Devuelve una lista con el nombre de los departamentos que **no** tienen un presupuesto entre 100000 y 200000 euros. Utilizando el operador BETWEEN.

    db.departamento.find({
    $nor: [{ presupuesto: { $gte: 100000, $lte: 200000 } }
    ]} , { nombre: 1, _id: 0 })

27. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos donde los gastos sean mayores que el presupuesto del que disponen.

    db.departamento.aggregate([

    {$match: { $expr: { $gt: ["$gastos", "$presupuesto"] }}},

    {

    $project: {

    nombre: 1,

    gastos: 1,

    presupuesto: 1,

    _id: 0}

    }])

28. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos donde los gastos sean menores que el presupuesto del que disponen.

```
db.departamento.aggregate([
 {$match: { $expr: { $lt: ["$gastos", "$presupuesto"] }}},
 {
  $project: {
   nombre: 1,
   gastos: 1,
   presupuesto: 1,
   _id: 0}
 }])
```

29. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos donde los gastos sean iguales al presupuesto del que disponen.

```
db.departamento.aggregate([
 {$match: { $expr: { $eq: ["$gastos", "$presupuesto"] }}},
 {
  $project: {
   nombre: 1,
   gastos: 1,
   presupuesto: 1,
   _id: 0}
 }])
```

30. Lista todos los datos de los empleados cuyo segundo apellido sea `NULL`.

```
db.empleado.find({apellido2: null})
```

31. Lista todos los datos de los empleados cuyo segundo apellido **no sea** `NULL`.

    db.empleado.find({apellido2: {$ne : null}})

32. Lista todos los datos de los empleados cuyo segundo apellido sea `López`.

    db.empleado.find({apellido2: "López"})

33. Lista todos los datos de los empleados cuyo segundo apellido sea `Díaz` o `Moreno`. Sin utilizar el operador `IN`.

    db.empleado.find({
    $or: [{ apellido2: "Díaz" },{apellido2: "Moreno"}
    ]})

34. Lista todos los datos de los empleados cuyo segundo apellido sea `Díaz` o `Moreno`. Utilizando el operador `IN`.

    db.empleado.find({ apellido2: { $in:["Díaz", "Moreno"]} })

35. Lista los nombres, apellidos y nif de los empleados que trabajan en el departamento `3`.

    db.empleado.find({id_departamento :3} , {_id:0 , nombre:1 , apellido1:1, apellido2:1 , nif :1})

36. Lista los nombres, apellidos y nif de los empleados que trabajan en los departamentos `2`, `4` o `5`.

    db.empleado.find({id_departamento : { $in : [2, 4, 5]}} , {_id:0 , nombre:1 , apellido1:1, apellido2:1 , nif :1})

Consultas multitabla

Resuelva todas las consultas utilizando la sintaxis de SQL1 y SQL2.

1. Devuelve un listado con los empleados y los datos de los departamentos donde trabaja cada uno.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_empleados'
      }
    },
    {
      $project: {
        _id: 0,
        'Nombre del empleado': '$nombre',
        'Datos de departamentos': {
          $arrayElemAt: ['$info_empleados', 0]
        }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

2. Devuelve un listado con los empleados y los datos de los departamentos donde trabaja cada uno. Ordena el resultado, en primer lugar por el nombre del departamento (en orden alfabético) y en segundo lugar por los apellidos y el nombre de los empleados.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_empleados'
      }
    },
    {
      $project: {
        _id: 0,
        'Nombre del empleado': '$nombre',
        'Apellido Paterno': '$apellido1',
        'Apellido Materno': '$apellido2',
        'Datos de departamentos': {
          $arrayElemAt: [
            '$info_empleados.nombre',
            0
          ]
        }
      }
    },
    {
      $sort: {
        'Datos de departamentos': 1,
        'Apellido Paterno': 1,
        'Apellido Materno': 1,
        'Nombre del empleado': 1
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

3. Devuelve un listado con el identificador y el nombre del departamento, solamente de aquellos departamentos que tienen empleados.

4.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_departamentos'
      }
    },
    { $match: { nombre: { $ne: null } } },
    {
      $project: {
        _id: 1,
        'Nombre Departamento': {
          $arrayElemAt: [
            '$info_departamentos.nombre',
            0
          ]
        }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

5. Devuelve un listado con el identificador, el nombre del departamento y el valor del presupuesto actual del que dispone, solamente de aquellos departamentos que tienen empleados. El valor del presupuesto actual lo puede calcular restando al valor del presupuesto inicial (columna `presupuesto`) el valor de los gastos que ha generado (columna `gastos`).

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_departamentos'
      }
    },
    {
      $match: {
        nombre: { $ne: null },
        info_departamentos: {
          $exists: true,
          $not: { $size: 0 }
        }
      }
    },
    {
      $project: {
        _id: 1,
        'Nombre Departamento': {
          $arrayElemAt: [
            '$info_departamentos.nombre',
            0
          ]
        },
        'Presupuesto Actual': {
          $subtract: [
            {
              $arrayElemAt: [
                '$info_departamentos.presupuesto',
                0
              ]
            },
            {
              $arrayElemAt: [
                '$info_departamentos.gastos',
                0
              ]
            }
          ]
        }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

6. Devuelve el nombre del departamento donde trabaja el empleado que tiene el nif `38382980M`.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_empleados'
      }
    },
    { $match: { nif: { $eq: '38382980M' } } },
    {
      $project: {
        _id: 0,
        'Nombre del Departamento': {
          $arrayElemAt: [
            '$info_empleados.nombre',
            0
          ]
        }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

7. Devuelve el nombre del departamento donde trabaja el empleado `Pepe Ruiz Santana`.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_empleados'
      }
    },
    {
      $match: {
        nombre: { $eq: 'Pepe' },
        apellido1: { $eq: 'Ruiz' },
        apellido2: { $eq: 'Santana' }
      }
    },
    {
      $project: {
        _id: 0,
        'Nombre del Departamento': {
          $arrayElemAt: [
            '$info_empleados.nombre',
            0
          ]
        }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

8. Devuelve un listado con los datos de los empleados que trabajan en el departamento de I+D. Ordena el resultado alfabéticamente.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_empleados'
      }
    },
    {
      $match: { 'info_empleados.nombre': 'I+D' }
    },
    {
      $project: {
        _id: 0,
        nif: 1,
        nombre: 1,
        apellido1: 1,
        apellido2: 1,
        id_departamento: 1
      }
    },
    { $sort: { nombre: 1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

9. Devuelve un listado con los datos de los empleados que trabajan en el departamento de `Sistemas`, `Contabilidad` o `I+D`. Ordena el resultado alfabéticamente.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_departamentos'
      }
    },
    {
      $match: {
        'info_departamentos.nombre': {
          $in: ['Sistemas', 'Contabilidad', 'I+D']
        }
      }
    },
    {
      $project: {
        _id: 0,
        nif: 1,
        nombre: 1,
        apellido1: 1,
        apellido2: 1,
        id_departamento: 1
      }
    },
    { $sort: { nombre: 1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

10. Devuelve una lista con el nombre de los empleados que tienen los departamentos que **no** tienen un presupuesto entre 100000 y 200000 euros.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_departamentos'
      }
    },
    {
      $match: {
        $nor: [
          {
            'info_departamentos.presupuesto': {
              $gt: 100000,
              $lt: 200000
            }
          }
        ]
      }
    },
    { $project: { _id: 0, nombre: 1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

11. Devuelve un listado con el nombre de los departamentos donde existe algún empleado cuyo segundo apellido sea NULL. Tenga en cuenta que no debe mostrar nombres de departamentos que estén repetidos.

```
db.getCollection('empleado').aggregate(
  [
    {
      $lookup: {
        from: 'departamento',
        localField: 'id_departamento',
        foreignField: '_id',
        as: 'info_departamentos'
      }
    },
    { $match: { apellido2: null } },
    {
      $project: {
        _id: 0,
        'Nombre Departamento': {
          $arrayElemAt: [
            '$info_departamentos.nombre',
            0
          ]
        }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```