Taller MongoDB

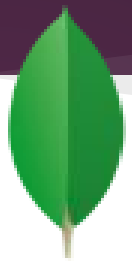
2do Cuatrimestre 2015
Base de datos

Planificación

- Definir esquema a partir de un DER.
  - Aplicar desnormalización.
- Crear los distintos documentos.
- Insertar, eliminar, modificar.
- Realizar consultas.
  - Consultas de agregación
- Aplicar Map Reduce
- Presentamos el TP2 !!
- Devolvemos los parciales

MongoDB

MongoDB



mongoDB

{ name: mongo, type: DB }

- Orientado a Documentos
- Gran soporte de consultas
- Replication
- Balanceo de carga - Sharding
- File Storage
- Map Reduce
- Server-side JavaScript execution

MongoDB

- As of July 2015, MongoDB is the ***fourth most popular*** type of database management system, and ***the most popular for document stores.***
- También tiene criticas!
  - Concurrencia
  - Memoria
  - Esquema no definido

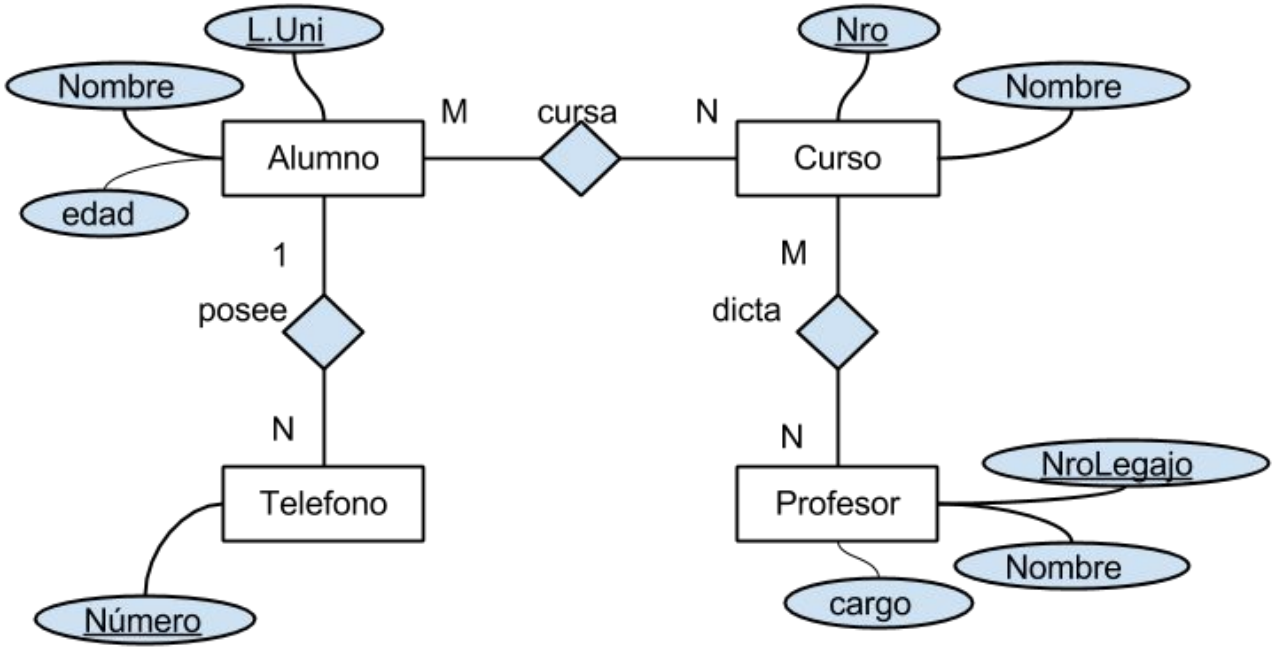
Consola

```
fernando@insp15:~$ mongo
MongoDB shell version: 2.4.9
connecting to: test
>
```

Consola

- `show databases`
- `use <db>`
- `show collections`

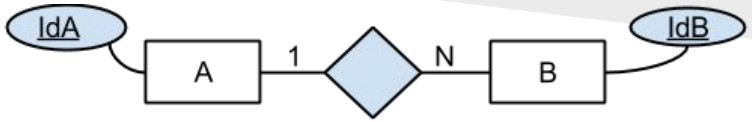
Diagrama


L.Uni
Nombre
Alumno
edad
M
cursa
N
Nro
Nombre
Curso
1
posee
N
Telefono
Número
M
dicta
N
Profesor
NroLegajo
Nombre
cargo

Desnormalización

- Qué debemos desnormalizar?
- En qué lugares desnormalizamos?
- Hay una sola forma de desnormalizar?

Desnormalización




```
CREATE TABLE A (
 IdA INT NOT NULL;
 ...
);
```

```
CREATE TABLE B (
 IdA INT NOT NULL;
 ...
);
```

```
"A": {
  "IdA" : 1,
  "B": [
        {"IdB": 1},
        {"IdB": 2},
        {"IdB": 3}
      ]
}
```

Modelo Relacional

Modelo Orientado a Documentos

Insertar datos

- Qué utilizamos como primary key? como foreign key?
- Quién es el responsable del ingreso de datos?
- Quien es el responsable de respetar el esquema?

Operaciones sobre Collections

- `db.collection.find()`
- `db.collection.insert()`
- `db.collection.update()`

Operaciones sobre Collections

```
db.users.insert (
   {
     name: "sue",
     age: 26,
     status: "A"
   }
)
```

collection

field: value

field: value

field: value

document

```
db.users.remove(
   { status: "D" }
)
```

collection

remove criteria

```
db.users.update(
   { age: { $gt: 18 } },
   { $set: { status: "A" } },
   { multi: true }
)
```

collection

update criteria

update action

update option

Consultas

- Tomar todos los alumnos entre 18 y 22 años
- Devolver los alumnos del curso número "55"
- Contabilizar la cantidad de profesores por cargo
- La carrera que tenga más de 2 alumnos entre 23 y 30 años

Consultas

- Tomar todos los alumnos entre 18 y 22 años

Consultas

- Tomar todos los alumnos entre 18 y 22 años
  - ```
    db.alumnos.find(
        {age: {"$gt": 18, "$lt": 22}})
    ```
    ```

Consultas

- Devolver los alumnos del curso número "55"

Consultas

- Devolver los alumnos del curso número "55"
  - ```
    db.alumnos.find(
    {"cursos": {$elemMatch: { nro: "55"}}}  )
    ```
    ```

Consultas

- Devolver los alumnos del curso número "55"
  - `db.alumnos.find(`

    `{"cursos": {$elemMatch: { nro: "55"}}}`

- La cantidad de alumnos del curso...
  - `db.alumnos.count(`

    `{"cursos": {$elemMatch: { nro: "55"} } })`

Consultas

- Contabilizar la cantidad de profesores por cargo

Consultas

- Contabilizar la cantidad de profesores por cargo
  - ```
    db.profesores.aggregate([{
    $group: { _id: "$cargo",
              total: {$sum: 1}} }])
    ```
    ```

Consultas

- La carrera que tenga más de 2 alumnos entre 23 y 30 años

Consultas

- La carrera que tenga más de 2 alumnos entre 23 y 30 años
  - ```
    db.alumnos.aggregate([
        {$match: {"edad": { $lt: 30, $gt: 23}}},
        {$group: { _id: "$carrera",
                   total: {$sum: 1}} },
        {$match: {"total": {$gt: 2}}}    ])
    ```
    ```

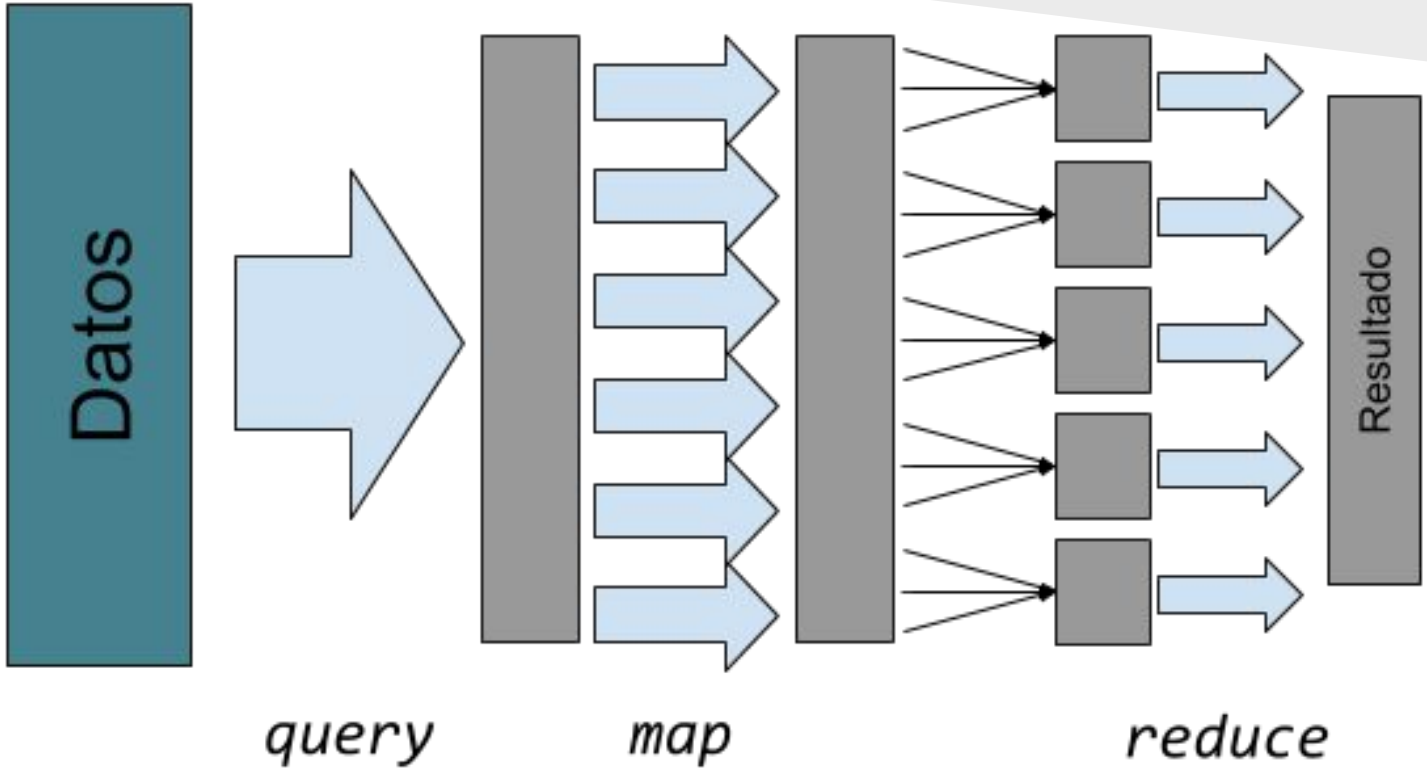
Map Reduce

Se define varios pasos utilizando funciones:

- (opcional) query: filtra los documentos por un atributo.
- ***map:*** función que emite una tupla <key, value> por cada documento. Agrupa todos los key y se devuelve la lista de values.

- ***reduce:*** función que toma cada <key, [value]> y devuelve el resultado.

Map Reduce



Consulta 1

- Devolver el nombre de los Alumnos con más de 5 teléfonos

Consulta 1

- Devolver el nombre de los Alumnos con más de 5 teléfonos
  - Map: filtro cada alumno por su cantidad de telefonos. Si es mayor a 5, emito <Lu, Nombre>
  - Reduce: Devuelvo cada alumno

Consulta 1

- Devolver el nombre de los Alumnos con más de 5 teléfonos

```
var m = function(){
    if(this["telefonos"].length>5)
        {emit(this["lu"],this["nombre"])
    }}
var r = function(key, values){ return values[0]}
db.alumnos.mapReduce(m, r, {out: "map_res"})
```

Consulta 2

- La cantidad de Profesores que hay por cargo

Consulta 2

- La cantidad de Profesores que hay por cargo
  - Map: emitimos <cargo, 1>
  - Reduce: sumamos todos los elementos de la lista

Consulta 2

- La cantidad de Profesores que hay por cargo

```
var m = function(){emit(this["cargo"],1)}
var r = function(key, values){ return Array.sum(values)}
db.profesores.mapReduce(m, r, {out: "map_res"})
```

Consulta 3

- Cantidad de alumnos totales en cursos obligatorios.

Consulta 3

- Cantidad de alumnos totales en cursos obligatorios.
  - Map: si es curso obligatorio emitimos

    <tipo de curso, #alumnos>
  - Reduce: sumamos cantidad de alumnos.

Consulta 3

- Cantidad de alumnos totales en cursos obligatorios.

```
var m = function(){
    if(this["tipo"] =="obligatorio"){
    emit(this["tipo"],this["alumnos"].length)}
    }
var r = function(key, values){
        return Array.sum(values)  }
```

Consulta de tarea

- Devolver los Alumnos más viejos por cada curso.