



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico #2

Histórico de Competiciones

16 de junio de 2017

Bases de Datos

Grupo 7

Integrante	LU	Correo electrónico
Abdala Leila	950/12	abdalaleila@gmail.com
Bernaus Andres	699/10	andres.bernaus@hotmail.com
Gonzalez Alejandro	32/13	gonzalezalejandro1592@gmail.com
Romero Lucas	440/12	lucasrafael.romero@gmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		



Índice

1. Introducción	2
2. Modelo	2
3. Diagrama de Interrelación de Documentos	4
3.1. Consultas	4
4. Json Schema	5
5. Implementación de Funcionalidades	9
5.1. Sharding	11
6. Conclusión	12

1. Introducción

Para esta segunda parte del trabajo práctico, se desea modelar e implementar el historial de las competiciones mundiales; incluyendo el histórico de árbitros y escuelas que participaron en cada campeonato, los enfrentamientos por categoría y sus respectivos medalleros.

El objetivo principal de esta base de datos será, en primer lugar, responder a las consultas requeridas en el enunciado. Por lo tanto, nos concentraremos fuertemente en modelar los documentos de la misma, para así poder resolverlas eficientemente.

2. Modelo

A continuación veremos en detalle los distintos aspectos del problema a resolver, y como decidimos modelarlos para cumplir con los requerimientos del problema.

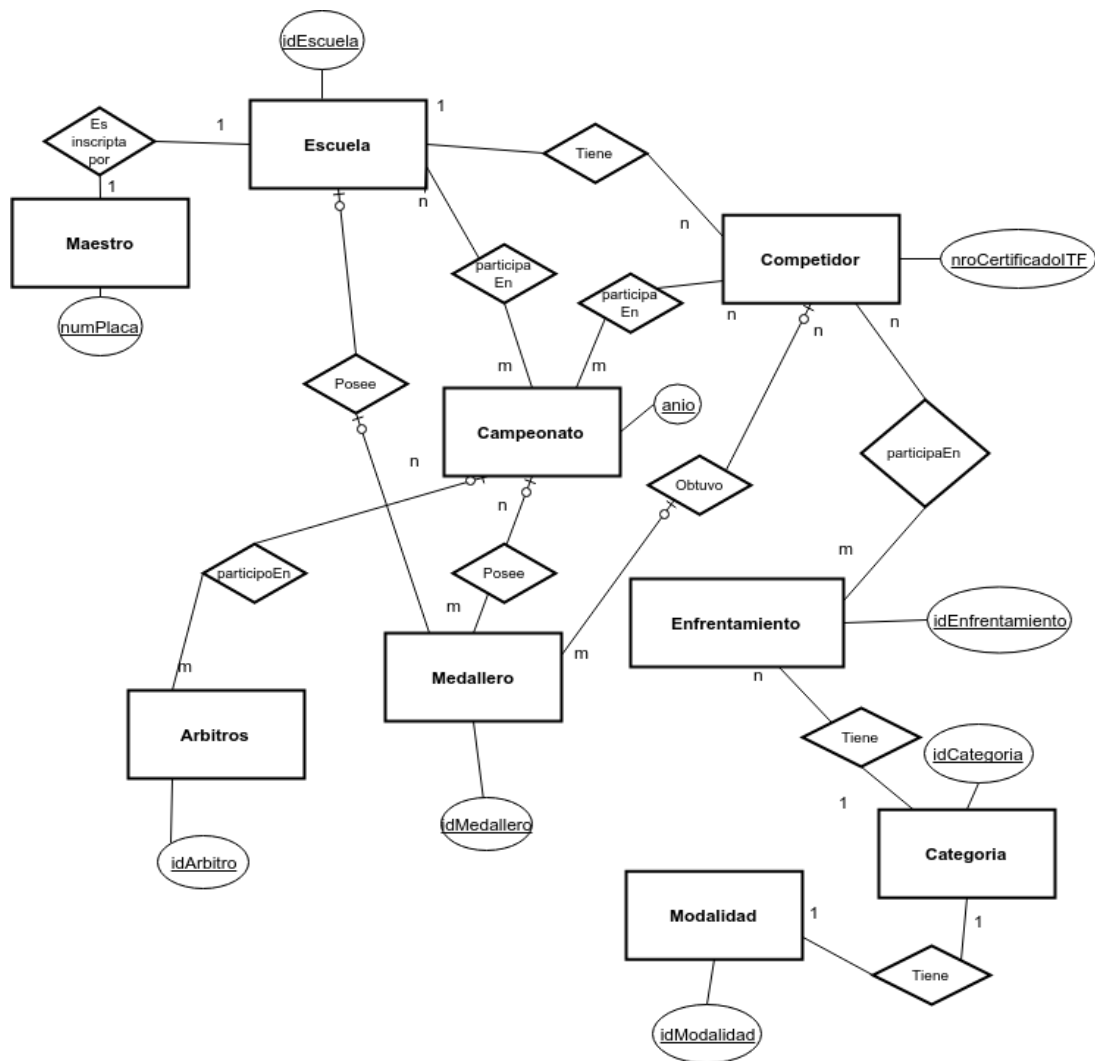
Se nos pide, en primer lugar, guardar para cada campeonato y cada categoría de dicho campeonato, todos los enfrentamientos con sus resultados y medallas. Podemos modelar esto de la siguiente manera:

- En primer lugar tenemos una relación N, M , entre los campeonatos y los árbitros que participaron en dicho campeonato. De la misma manera, existe una relación N, M entre los competidores y un campeonato determinado, y las escuelas y un campeonato.
- El campeonato debe estar relacionado con la entidad enfrentamientos. Como cada enfrentamiento está directamente asociado a una categoría y a un campeonato, entonces podemos modelar una relación 1 a 1 entre enfrentamiento y su modalidad; y una relación 1 a N entre el campeonato y sus enfrentamientos.
- Por último, los campeonatos deben conocer los medalleros correspondientes a cada categoría. Luego necesariamente debemos modelarlo como una relación N, M entre el medallero y el campeonato.

Además de cumplir con los requerimientos básicos del enunciado, debemos asegurarnos de modelar el problema de tal manera que las consultas puedan resolverse eficientemente.

Algunas relaciones triviales que se desprenden de las consultas son: los enfrentamientos están relacionados con competidores; las escuelas están relacionadas con competidores.

Luego el Diagrama de entidad relación para este problema es:



Un aspecto a tener en cuenta es que a diferencia del TP1, como tenemos datos historicos. los datos de peso graduacion y edad no se van a corresponder con las restricciones de cada competencia. Es decir, podria llegar a tener a un competido de 40 años, pero en su historial figura que compitio en una categoria de hasta 30 años. Lo Cual tiene sentido si dicha competencia ocurrior con bastante más anterioridad.

3. Diagrama de Interrelación de Documentos

Una vez obtenido el diseño de la base de datos que deseamos modelar, el siguiente paso es comenzar a transformar dicho modelo en un esquema de documentos. Para ello utilizamos el Diagrama de interrelacion de Documentos, con el objetivo de poder ilustrar como nuestro modelo resuelve las consultas del enunciado de forma eficiente. Para ello iremos viendo cada consulta y trataremos de modelar los documentos en base a ellos.

3.1. Consultas

La **primera consulta** requiere: devolver la cantidad de enfrentamientos ganados por competidor para un campeonato dado. Es decir, que si busco un registro particular de Campeonato, necesitamos poder devolver tuplas de (*competidor, cantidadDeEnfrentamientosGanados*). En consecuencia, además de contar con la lista de competidores que participaron en cada Campeonato, almacenaremos tambien sus cantidades de enfrentamientos ganados.

Para la **segunda consulta** y **tercera consulta**, se necesita poder obtener las cantidades de medallas pr campeonato y el total de medallas en la historia (consultas 2.3 y 2.2 respectivamente). Usando el mismo razonamiento que en la consulta anterior; almacenamos un arreglo de campeonatos junto a la cantidad de medallas obtenidas en el mismo. Luego resolver la consulta se reduce a sumar todas las medallas, en el caso de la 2.2, y encontrar el maximo valor para la consulta 2.3.

El caso del punto **2.4** quedo resuelto en la etapa de diseño. Como los arbitros poseen una lista con sus participaciones en los campoeonatos, podemos resolver esta consulta fácilmente tomando el tamaño de la misma.

Para la **consulta 2.5** volvemos a interactuar con campeonatos y competidores de forma similar a la primer consulta. La única diferencia es que ahora necesitamos conocer las escuelas a las cuales pertenecen los competidores. La solución que encontramos fue la siguiente: reorganizar la lista de competidores de cada y agruparlos por Escuela. De esta manera, podemos contabilizar rápidamente la cantidad de competidores por Escuela, y ademas cumplir con las consultas 2.2 y 2.3.

Para la **ultima consulta (2.6)** se nos pide obtener los competidores que más medallas obtuvieron por categoria. Para ello decidimos incluir en el *Competidor*, un campo por cada una de las categorias en el cual se lleve la cuenta de las medallas obtenidas por dicho competidor. Luego, para resolver la consulta, podemos recorrer la tabla de competidores y quedarnos con aquellos que tengan mayor cantidad de medallas en alguna de las categorias.

Teniendo en cuenta estas consideraciones, el diagrama de interrelación de Documentos resultante será:

4. Json Schema

A partir de las consideraciones del punto anterior tomamos cada Documento y obtuvimos los siguientes Json schemas:

```
"Campeonato":{
  "properties": {
    "Medallero": {
      "items": {
        "properties": {
          "idCategoria": {
            "type": "integer"
          },
          "idPrimerCompetidor": {
            "type": "integer"
          },
          "idSegundoCompetidor": {
            "type": "integer"
          },
          "idTercerCompetidor": {
            "type": "integer"
          }
        }
      },
      "type": "object"
    },
    "type": "array"
  },
  "arbitros": {
    "items":{
      "properties": {
        "idArbitro":{
          "type": "integer"
        }
      }
    },
    "type": "array"
  },
  "año": {
    "type": "integer"
  },
  "escuelas": {
    "items": {
      "properties": {
        "competidores": {
          "items": {
            "properties": {
              "cantEnfrentamientos": {
                "type": "integer"
              },
              "idCompetidor": {
                "type": "integer"
              }
            }
          },
          "type": "object"
        },
        "type": "array"
      },
      "nombreEscuela": {
        "type": "string"
      }
    }
  }
}
```

```

        },
        "type": "object"
    },
    "type": "array"
},
"required":["año","escuelas","arbitros"]
},
"type": "object"
}

```

```

"Categoria":{
  "properties": {
    "Edad": {
      "properties": {
        "max": {
          "type": "integer"
        },
        "min": {
          "type": "integer"
        }
      },
      "type": "object"
    },
    "Graduacion": {
      "type": "integer"
    },
    "Modalidad": {
      "type": "string"
    },
    "Peso": {
      "properties": {
        "max": {
          "type": "integer"
        },
        "min": {
          "type": "integer"
        }
      },
      "type": "object"
    },
    "idCategoria": {
      "type": "integer"
    },
    "required":["Edad","Graduacion","Modalidad","Peso","idCategoria"]
  },
  "type": "object"
}

```

```

"Enfrentamiento": {
  "properties": {
    "Competidores": {
      "items": {
        "type": "integer"
      },

```

```

        "type": "array"
    },
    "Ganador": {
        "type": "integer"
    },
    "idCategoria": {
        "type": "integer"
    },
    "idEnfrentamiento": {
        "type": "integer"
    },
    "required":["Competidores","Ganador","idCategoria","idEnfrentamiento"]
},
"type": "object"
}

"escuela":{
"type":"object",
"properties":{
"nombreMaestro":{"type":"string"},
"apellidoMaestro":{"type":"string"},
"idEscuela":{"type":"integer"},
"nombreEscuela":{"type":"string"},
"graduacionMaestro":{"type":"number", "multipleOf": 1.0,"minimum":1,"maximum":6},
"nroPlacaInstructor":{"type":"string"},
"pais":{"type":"string"},
"medallasPorCampeonato":{
"type":"array",
"items":{
"type":"object"
"properties":{
"campeonato":{"type":"integer"},
"cantidadMedallas":{"type":"number", "multipleOf": 1.0,"minimum":0}
}
}
},
"required":["nombreMaestro","apellidoMaestro","idEscuela","nombreEscuela",
"graduacion","nroPlacaInstructor","pais"]
}
}

"arbitro":{
"type":"object",
"properties":{
"nombre":{"type":"string"},
"apellido":{"type":"string"},
"graduacion":{"type":"number", "multipleOf": 1.0,"minimum":1,"maximum":6},
"nroPlaca":{"type":"string"},
"pais":{"type":"string"},
"participoEn":{
"type":"array",
"items": {"type":"integer","uniqueItems":true}
},
"required":["nombre","apellido","graduacion","nroPlaca","pais"]
}
}

```



```

"competidor":{
  "type": "object",
  "properties": {
    "nombre": {
      "type": "string"
    },
    "apellido": {
      "type": "string"
    },
    "dni": {
      "type": "string"
    },
    "fechaNacimiento": {
      "type": "string",
      "format": "date-time"
    },
    "genero": {
      "type": "string",
      "enum": ["masculino", "femenino"]
    },
    "graduacion": {
      "type": "number",
      "multipleOf": 1.0,
      "minimum": 1,
      "maximum": 6
    },
    "nroCertificadoITF": {
      "type": "string"
    },
    "peso": {
      "type": "number",
      "multipleOf": 1.0,
      "minimum": 1
    },
    "nombreEscuela": {
      "type": "string"
    },
    "pais": {
      "type": "string"
    },
    "numMedallasCombate": {
      "type": "integer"
    },
    "numMedallasSalto": {
      "type": "integer"
    },
    "numMedallasRotura": {
      "type": "integer"
    },
    "numMedallasFormas": {
      "type": "integer"
    },
    "numMedallasEquipos": {
      "type": "integer"
    },
    "required": ["nombre", "apellido", "graduacion",
      "dni", "nombreEscuela", "nroCertificadoITF"]
  }
}

```

5. Implementación de Funcionalidades

Una vez que terminado el modelo del problema, el siguiente paso fue comenzar a implementar la base de datos. Es decir generar en RethinkDB las distintas tablas, documentos y consultas necesarias para resolver el problema.

Para ello creamos un pequeño programa que genere de manera aleatoria todos los datos necesarios, asegurándose de ser consistente al momento de crear documentos para los enfrentamientos, campeonatos, escuelas y demás. Ya que, por ejemplo, si un competidor gana un enfrentamiento, el mismo debe ser contabilizado en el Documento *Campeonato*.

Una vez llenadas las tablas con los suficientes datos, implementamos cada una de las consultas:

■ Cantidad de enfrentamientos ganados por competidor para un campeonato dado.

Como esta consulta debe realizarse para un Campeonato determinado, el primer paso es filtrar la tabla *Campeonatos* hasta obtener el documento indicado. Para ello utilizamos la función *filter* sobre el campo *año* que, por lo visto anteriormente, identifica inequívocamente a cada Campeonato.

Una vez seleccionado el campeonato, hicimos un *concatMap* y un *reduce* sobre todos los pares (*competidor, enfrentamientosGanados*) de todas las escuelas que participan del campeonato, obteniendo así los datos necesarios para esta consulta.

```
r.db('TP2').table('Campeonato').filter({año: 2017})("escuelas")
.concatMap(function(elem){
  return elem("competidores");
})
.reduce(function(left, right){
  return left.add(right);
})
```

■ Cantidad de medallas por nombre de escuela en toda la historia

Usando la función *map*, podemos tomar cada documento *Escuela* y devolver únicamente los campos que nos interesan para resolver la consulta, es decir, su *nombreEscuela* y la sumatoria de todos los campos *cantidadMedallas* que se encuentren en el arreglo *medallasPorCampeonato*.

```
r.db("TP2").table("Escuela").map(
function(elem){
  return {id: elem("idEscuela"), nombre: elem("nombreEscuela"),
    cantidadTotal: elem("medallasPorCampeonato").sum("cantidadMedallas")};
}
)
```

■ Para cada escuela, el campeonato donde ganó más medallas

De forma similar a la consulta anterior usamos una función *map*. La cual devolverá, *nombreEscuela* junto con el año del campeonato cuya tupla

(*anio, cantidadMedallas*) asociada sea máxima. Es decir, corresponda al elemento de *medallasPorCampeonato* con mayor cantidad de medallas.

```
r.db("TP2").table("Escuela").map(
  function(elem){
    var res = elem("medallasPorCampeonato").max("cantidadMedallas");
    return {escuela: elem("nombreEscuela"), campeonato: res("anio")};
  })
```

■ Los arbitros que participaron en al menos 4 campeonatos

Como cada documento *Arbitro* contiene una lista con los años de los campeonatos en los que participo, responder esta consulta significa filtrar aquellos árbitros que cumplan con la condición. Esto lo podemos hacer usando un *filter* sobre el largo del arreglo *participoEn*, y devolviendo solo aquellos que tengan longitud mayor o igual a 4.

```
r.db("TP2").table("Arbitro").filter(
  function(elem){
    return elem("participoEn").count().ge(4);
  })
```

■ Las escuelas que han presentado el mayor numero de competidores en cada campeonato

Usando un map podemos tomar cada campeonato, y como cada uno tiene asociado un arreglo con todas las escuelas, que a su vez contiene una lista con todos los competidores, podemos entonces responder a la consulta tomando el largo del arreglo de competidores, y comparándolo con las otras escuelas.

```
r.db('TP2').table('Campeonato').map(
  function(elem){

    var max = elem("escuelas").max(
      function(esc){
        return esc("competidores").count();
      })("competidores").count();

    return {anio: elem("anio"), escuelas: elem("escuelas").filter(
      function(val){
        return val("competidores").count().eq(max)
      }).pluck("idEscuela", "nombreEscuela")};
  })
```

■ Obtener los competidores que más medallas obtuvieron por modalidad

En esta consulta usaremos un map reduce. La idea es: en el map tomar a cada competidor y devolver una tupla con los campos

(*idCompetidorSalto, maxMedallasSalto, idCompetidorCombate, maxMedallasCombate, ...*) inicializándolos con su respectivo id, y con las cantidades de medallas en cada categoría (las cuales están almacenadas en cada competidor). Luego, en el reduce, comparamos dichos valores y modificamos el *idDelCompetidor* según sea necesario.

La consulta entonces quedaría:

```

r.db('tp2').table("Competidor").filter(
  function(elem){
    var maximos = {
      formas: r.db('tp2').table("Competidor").max("formas")("formas"),
      salto: r.db('tp2').table("Competidor").max("salto")("salto"),
      rotura: r.db('tp2').table("Competidor").max("rotura")("rotura"),
      combate: r.db('tp2').table("Competidor").max("combate")("combate")
    };

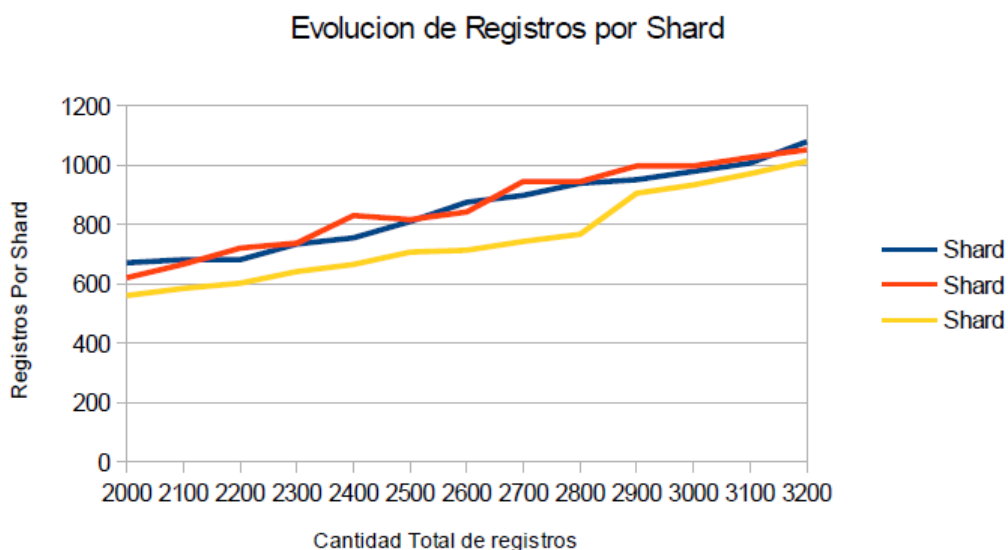
    return elem("salto").eq(maximos["salto"]).or(
      elem("formas").eq(maximos["formas"]),
      elem("combate").eq(maximos["combate"]),
      elem("rotura").eq(maximos["rotura"]))
  }).group(
    function(elem){
      var maximos = {
        formas: r.db('tp2').table("Competidor").max("formas")("formas"),
        salto: r.db('tp2').table("Competidor").max("salto")("salto"),
        rotura: r.db('tp2').table("Competidor").max("rotura")("rotura"),
        combate: r.db('tp2').table("Competidor").max("combate")("combate")
      };

      return elem("salto").eq(maximos["salto"]).branch(["salto"], []).add(
        elem("formas").eq(maximos["formas"]).branch(["formas"], []).add(
          elem("combate").eq(maximos["combate"]).branch(["combate"], []).add(
            elem("rotura").eq(maximos["rotura"]).branch(["rotura"], [])
          )
        )
      )
    }, {multi: true})

```

5.1. Sharding

Tomamos la tabla de Competidores, pues es la que tiene mayor cantidad de registros (alrededor de 2000), y creamos 3 shards. Inicialmente la base de datos particionó el total de los registros en un shard con 672 registros, el segundo con 621 y el tercero con 561. Luego realizamos inserciones de a 100 competidores y obtuvimos los siguientes resultados:



Como podemos ver en el grafico, rethinkDB distribuye equitativamente las inserciones a todos los shards, de manera de aliviar la carga de la base de datos al momento de responder consultas o realizar operaciones como inserciones.

6. Conclusión

Sobre este trabajo, queremos destacar algunas conclusiones a las que llegamos durante el transcurso del mismo.