Database 2

MongoDB Assignment

Name:	Cian Hackett
Student number:	C16723031
Lecturer's Name:	Ciaran Kelly
Programme:	DT228/3
Module:	Database 2
Submission Date:	13/12/18

Declaration of Ownership: I declare that the attached work is entirely my own and that all sources have been acknowledged:

Signed: ________**Cian Hackett**_________________

Date: _____**13/12/18**____________________

Contents

Aim 1
Introduction 1
Schema Structure 1
Queries 2
Conclusion 3
Appendix 4

Aim

To design one or more collections to store data from a dataset into MongoDB, using a 1:few, 1:many or 1:squillions design, create a new collection or set of collections to contain new documents. Outline the pros and cons of the redesigned schema. Write several queries on the new design structure.

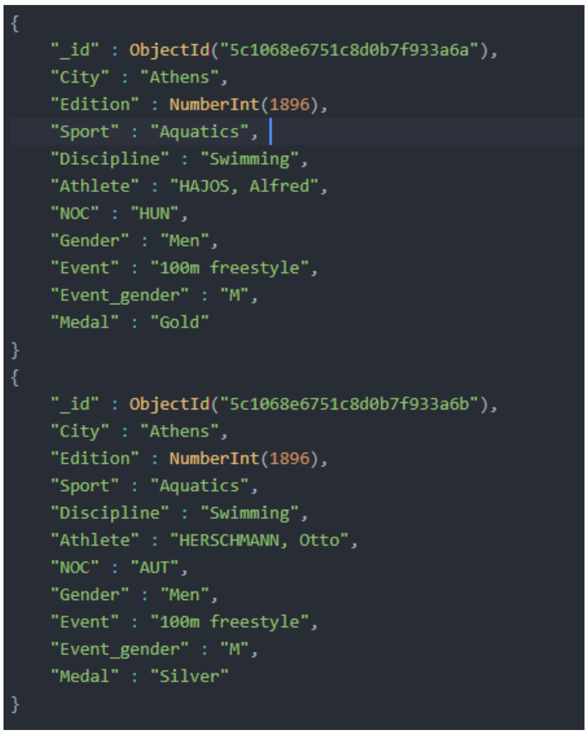
Introduction

This report will explain why the schema was redesigned with 1: few relationships. It will compare the original schema design with the new design to express reason why the new method was chosen. It will outline the positives for using the new design such as search speed and the negatives such as difficulties in insertion and deletion. By the end of this report the reader should expect to have a better understanding of when to use 1: few relationships in MongoDB.

Schema Structure

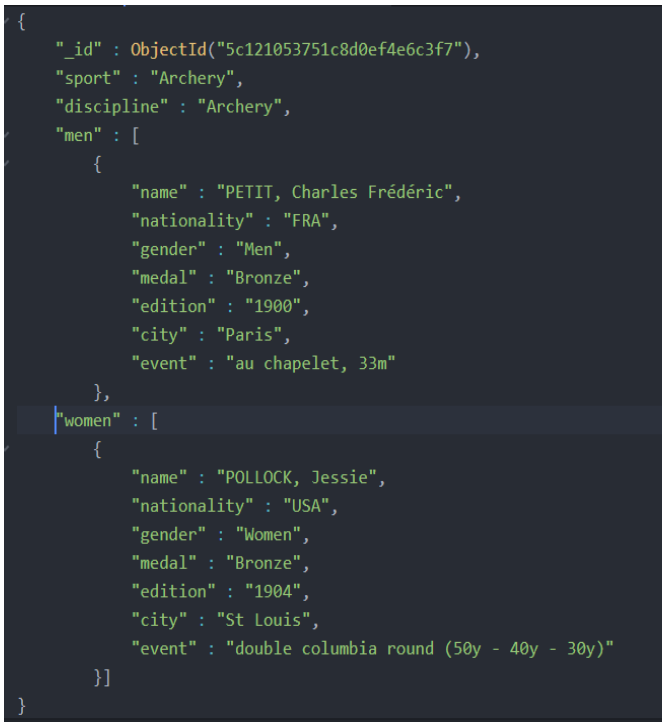
In the original schema design each object has a sports and discipline fields which repeat a for every object within those categories. If searches are made to find people who won medals within those categories it will be very costly as the query will need to iterate and check each object. This will not be the case for the new implementation because all the medallists are put into gender arrays under their categories. This in return speeds up the queries on a chosen discipline or sport. However, this leads to complication when inserting a new medallist. Therefore, it can be said that in this case a one to few relationship (as seen in new design) improves query speed but hampers insertion. Few, in 'one to few' is related to the number of gender arrays in the new schema. If the number of arrays was greater than twelve perhaps it might be referred to as one to many.

Original schema design



```
{
    "_id" : ObjectId("5c1068e6751c8d0b7f933a6a"),
    "City" : "Athens",
    "Edition" : NumberInt(1896),
    "Sport" : "Aquatics",
    "Discipline" : "Swimming",
    "Athlete" : "HAJOS, Alfred",
    "NOC" : "HUN",
    "Gender" : "Men",
    "Event" : "100m freestyle",
    "Event_gender" : "M",
    "Medal" : "Gold"
}
{
    "_id" : ObjectId("5c1068e6751c8d0b7f933a6b"),
    "City" : "Athens",
    "Edition" : NumberInt(1896),
    "Sport" : "Aquatics",
    "Discipline" : "Swimming",
    "Athlete" : "HERSCHMANN, Otto",
    "NOC" : "AUT",
    "Gender" : "Men",
    "Event" : "100m freestyle",
    "Event_gender" : "M",
    "Medal" : "Silver"
}
```

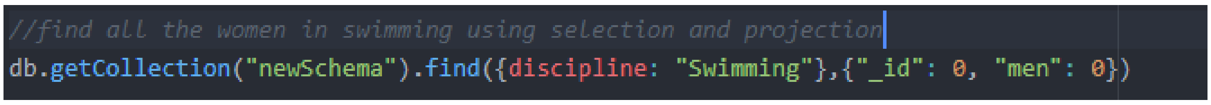
New Schema Design



```
{
    "_id" : ObjectId("5c121053751c8d0ef4e6c3f7"),
    "sport" : "Archery",
    "discipline" : "Archery",
    "men" : [
        {
            "name" : "PETIT, Charles Frédéric",
            "nationality" : "FRA",
            "gender" : "Men",
            "medal" : "Bronze",
            "edition" : "1900",
            "city" : "Paris",
            "event" : "au chapelet, 33m"
        },
    "women" : [
        {
            "name" : "POLLOCK, Jessie",
            "nationality" : "USA",
            "gender" : "Women",
            "medal" : "Bronze",
            "edition" : "1904",
            "city" : "St Louis",
            "event" : "double columbia round (50y - 40y - 30y)"
        }]
}
```

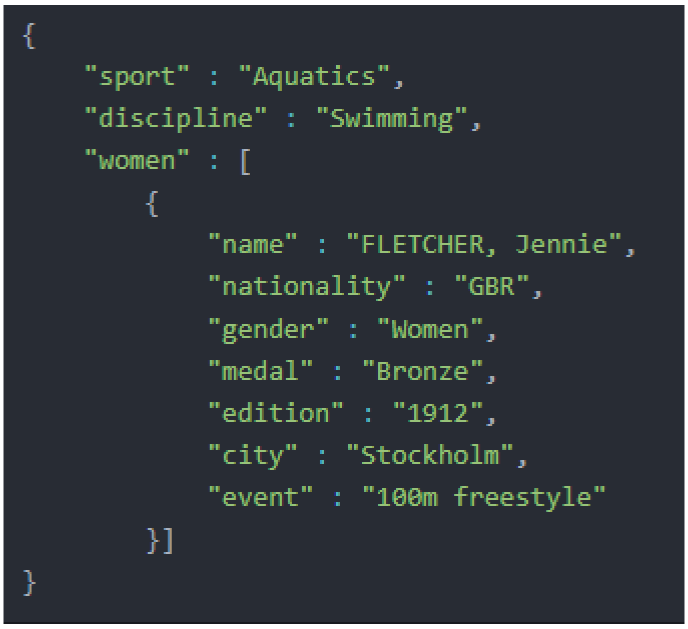
Queries

Selection and Projection Query – selecting object with swimming as a discipline. Projection: removing _id and men from result.



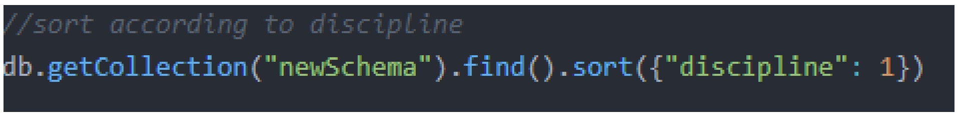
```
//find all the women in swimming using selection and projection
db.getCollection("newSchema").find({discipline: "Swimming"},{"_id": 0, "men": 0})
```

Result – (for example this only copied one-woman object)



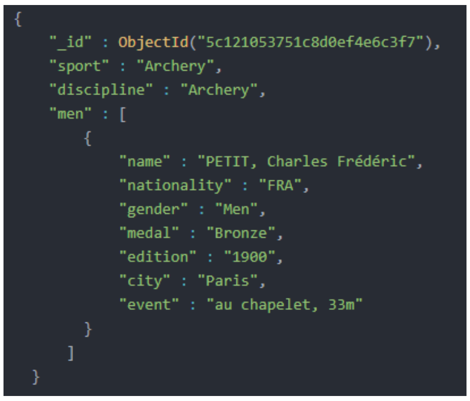
```
{
    "sport" : "Aquatics",
    "discipline" : "Swimming",
    "women" : [
        {
            "name" : "FLETCHER, Jennie",
            "nationality" : "GBR",
            "gender" : "Women",
            "medal" : "Bronze",
            "edition" : "1912",
            "city" : "Stockholm",
            "event" : "100m freestyle"
        }]
}
```

Sorting Query – Sort using the discipline column. Going from A to Z. (otherwise it would be discipline: -1 for Z to A)



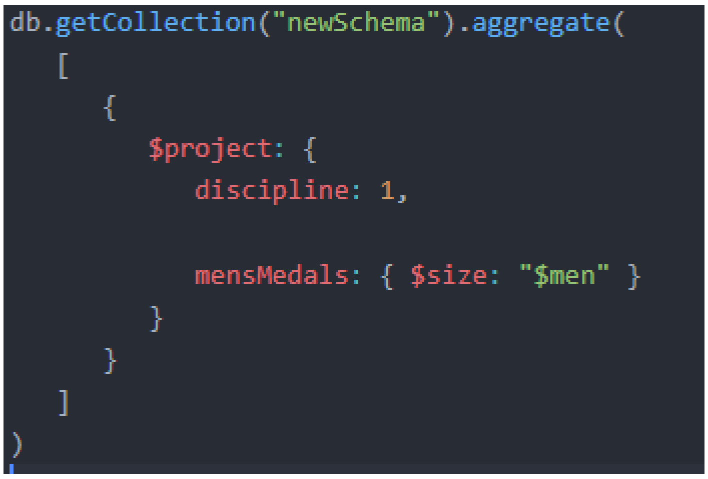
```
//sort according to discipline
db.getCollection("newSchema").find().sort({"discipline": 1})
```

Result example



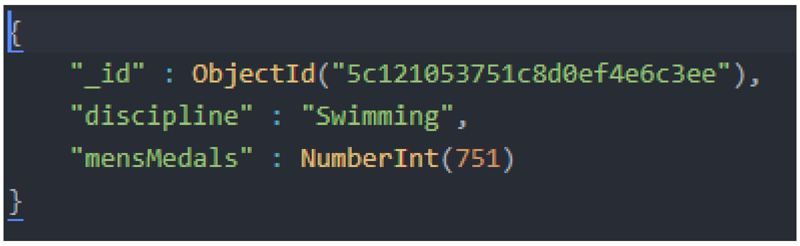
```
{
    "_id" : ObjectId("5c121053751c8d0ef4e6c3f7"),
    "sport" : "Archery",
    "discipline" : "Archery",
    "men" : [
        {
            "name" : "PETIT, Charles Frédéric",
            "nationality" : "FRA",
            "gender" : "Men",
            "medal" : "Bronze",
            "edition" : "1900",
            "city" : "Paris",
            "event" : "au chapelet, 33m"
        }
      ]
  }
```

Aggregation with Array Size – counting the number of medals gave out per discipline in the history of the Olympics.


```
db.getCollection("newSchema").aggregate(
   [
      {
         $project: {
            discipline: 1,

            mensMedals: { $size: "$men" }
         }
      }
   ]
)
```

Result example


```
{
    "_id" : ObjectId("5c121053751c8d0ef4e6c3ee"),
    "discipline" : "Swimming",
    "mensMedals" : NumberInt(751)
}
```

Conclusion

To conclude this report has outlined the reason what is a one to few relationships and why was it used. It went on to discuss the advantages and disadvantages of using one to few relationships. Finally, it goes on to display the required queries with their outputs.

Appendix

This appendix includes the code used to covert the old schema structure to the new structure.

main.java

```
package mongodb;


import java.io.BufferedWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.Writer;

import java.lang.reflect.Type;

import java.net.UnknownHostException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashMap;

import java.util.List;

import java.util.Map;


import org.bson.Document;


import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import com.mongodb.BasicDBList;

import com.mongodb.BasicDBObject;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.DBCursor;
import com.mongodb.DBObject;

import com.mongodb.MongoClient;

import com.mongodb.MongoClientURI;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import com.mongodb.util.JSON;

import com.mongodb.util.JSON;




public class Main {


	public static void main(String[] args) throws IOException {

		MongoClient mongoClient = new MongoClient(new
MongoClientURI("mongodb://localhost:27017"));

		DB database = mongoClient.getDB("MongoDB");

		DBCollection collection = database.getCollection("Olympics");

		DBCursor cursor = collection.find();


		List<DBObject> jo = (List<DBObject>) cursor.toArray();

		List<OlympicNew> olympicList = new ArrayList<>();

		String json  = jo.toString();

		Gson gson = new Gson();

		Type type = new TypeToken<List<Olympic>>() {}.getType();

		List<Olympic> olympic_games = gson.fromJson(json, type);

		List<Man> men = null;

		List<Woman> women = null;

		String fileName = "newSchema.json";

		List<Sport> sports = new ArrayList<Sport>();

		Map<String, Integer> checkInserts = new HashMap<String, Integer>();

		Gson gson1 = new Gson();
```

```
                    File file = new File(fileName);
                    FileWriter fr = new FileWriter(file, true);
                    BufferedWriter br = new BufferedWriter(fr);
                    String sport = null;
                    String discipline = null;
                    Map<String,Integer> person = new HashMap<String,Integer>();
                    Map<String,Integer> previous_sport = new HashMap<String,Integer>();
                    Map<String,Integer> previous_discipline = new HashMap<String,Integer>();
                    Sport sportie = null;

                     for(int i = 0; i < olympic_games.size(); i++) {

                         sport = olympic_games.get(i).getSport();
                         discipline = olympic_games.get(i).getDiscipline();
                         //empty list each time
                         men = new ArrayList<Man>();
                         women = new ArrayList<Woman>();

                         if(previous_sport.get(sport) == null && previous_discipline.get(discipline)
== null) {

                              previous_sport.put(sport, 1);
                              previous_discipline.put(discipline, 1);

                              for(int j = 0; j < olympic_games.size(); j++) {

                                   if(olympic_games.get(j).getSport().equals(sport) &&
olympic_games.get(j).getDiscipline().equals(discipline) ) {

     if(person.get(olympic_games.get(j).getAthlete()) == null) {
            person.put(olympic_games.get(j).getAthlete(),1);


            if(olympic_games.get(j).getGender().equals("Men")) {


             men.add(new Man(olympic_games.get(j).getAthlete(), olympic_games.get(j).getNOC(),

                                                                        olympic_games.get(j).getGender(),
olympic_games.get(j).getMedal(),

                                                                        olympic_games.get(j).getEdition(),
olympic_games.get(j).getCity(),

                                                                        olympic_games.get(j).getEvent()));




                                                                                }else if
(olympic_games.get(j).getGender().equals("Women")) {

                                                                //woman

             women.add(new Woman(olympic_games.get(j).getAthlete(),
olympic_games.get(j).getNOC(),

                                                                        olympic_games.get(j).getGender(),
olympic_games.get(j).getMedal(),

                                                                       olympic_games.get(j).getEdition(),
olympic_games.get(j).getCity(),

                                                                       olympic_games.get(j).getEvent() ));
```

```
                                        }


                                                    } // if inner arent same person
                                         }//if inner has sport and diciplines


                                   }// inner loop
                                   sportie = new Sport(sport,discipline,men,women);
                                        String json2 = gson1.toJson(sportie);
//                                      br.write(json2);
                                        System.out.println(json2 + "\n");
                                        br.write(json2);


                             }//if sports previous
                       }// outthere


                             br.close();
                             fr.close();
                       }
}


olymipic.java
package mongodb;


public class Olympic {
       private String Athlete;


       private String Medal;


       private Object _id;
       private String Edition;

       private String Event;

       private String NOC;

       private String Gender;

       private String Sport;

       private String Event_gender;

       private String Discipline;

       private String City;

       public String getAthlete ()
       {
          return Athlete;
       }

       public void setAthlete (String Athlete)
       {
          this.Athlete = Athlete;
       }

       public String getMedal ()
       {
          return Medal;
       }
```

```
public void setMedal (String Medal)
{
   this.Medal = Medal;
}

public Object get_id ()
{
   return _id;
}

public void set_id (Object _id)
{
   this._id = _id;
}

public String getEdition ()
{
   return Edition;
}

public void setEdition (String Edition)
{
   this.Edition = Edition;
}

public String getEvent ()
{
   return Event;
}

public void setEvent (String Event)
```

```
{
   this.Event = Event;
}

public String getNOC ()
{
   return NOC;
}

public void setNOC (String NOC)
{
   this.NOC = NOC;
}

public String getGender ()
{
   return Gender;
}

public void setGender (String Gender)
{
   this.Gender = Gender;
}

public String getSport ()
{
   return Sport;
}

public void setSport (String Sport)
{
    this.Sport = Sport;
  }

  public String getEvent_gender ()
  {
    return Event_gender;
  }

  public void setEvent_gender (String Event_gender)
  {
    this.Event_gender = Event_gender;
  }

  public String getDiscipline ()
  {
    return Discipline;
  }

  public void setDiscipline (String Discipline)
  {
    this.Discipline = Discipline;
  }

  public String getCity ()
  {
    return City;
  }

  public void setCity (String City)
  {
    this.City = City;
		}

		@Override
		public String toString()
		{
		    return "Olympic [Athlete = "+Athlete+", Medal = "+Medal+", _id = "+_id.toString()+",
Edition = "+Edition+", Event = "+Event+", NOC = "+NOC+", Gender = "+Gender+", Sport =
"+Sport+", Event_gender = "+Event_gender+", Discipline = "+Discipline+", City = "+City+"]";
		}
}
```

sport.java

```
package mongodb;

import java.util.List;

public class Sport {

	private String sport;
	private String discipline;
	private String event_gender;
	private Man[] men = {};
	private Woman[] women = {};

	public Sport(String sport, String discipline, List<Man> men, List<Woman> women) {
		this.setSport(sport);
		this.setDiscipline(discipline);
		this.setMen(men);
		this.setWomen(women);
	}
```

```
String getSport() {
        return sport;
}

void setSport(String sport) {
        this.sport = sport;
}

String getDiscipline() {
        return discipline;
}

void setDiscipline(String discipline) {
        this.discipline = discipline;
}

String getEvent_gender() {
        return event_gender;
}

void setEvent_gender(String event_gender) {
        this.event_gender = event_gender;
}

Man[] getMen() {
        return men;
}

void setMen(List<Man> men) {
        this.men = men.toArray(this.men);
    }

    Woman[] getWomen() {
        return women;
    }

    void setWomen(List<Woman> women) {
        this.women = women.toArray(this.women);
    }
}
```

man.java

```
package mongodb;

public class Man {


    private String name;
    private String nationality;
    private String gender;
    private String medal;
    private String edition;
    private String city;
    private String event;

    public Man(String name, String nationality,
            String gender, String medal, String edition, String city, String event) {
        this.setName(name);
        this.setNationality(nationality);
        this.setGender(gender);
        this.setMedal(medal);
        this.setEdition(edition);

        this.setCity(city);

        this.setEvent(event);

}




String getName() {

        return name;

}

void setName(String name) {

        this.name = name;

}

String getNationality() {

        return nationality;

}

void setNationality(String nationality) {

        this.nationality = nationality;

}

String getGender() {

        return gender;

}

void setGender(String gender) {

        this.gender = gender;

}

String getMedal() {

        return medal;

}

void setMedal(String medal) {

        this.medal = medal;

}
```

```
	String getEdition() {
		return edition;
	}
	void setEdition(String edition) {
		this.edition = edition;
	}
	String getCity() {
		return city;
	}
	void setCity(String city) {
		this.city = city;
	}




	public String getEvent() {
		return event;
	}




	public void setEvent(String event) {
		this.event = event;
	}
}
```

woman.java

```
package mongodb;

public class Woman {

	private String name;
private String nationality;

private String gender;

private String medal;

private String edition;

private String city;

private String event;



public Woman(String name, String nationality,

                    String gender, String medal, String edition, String city, String event) {

          this.setName(name);

          this.setNationality(nationality);

          this.setGender(gender);

          this.setMedal(medal);

          this.setEdition(edition);

          this.setCity(city);

          this.setEvent(event);


}




String getName() {

          return name;

}

void setName(String name) {

          this.name = name;

}

String getNationality() {

          return nationality;

}
```

```
void setNationality(String nationality) {
        this.nationality = nationality;
}
String getGender() {
        return gender;
}
void setGender(String gender) {
        this.gender = gender;
}
String getMedal() {
        return medal;
}
void setMedal(String medal) {
        this.medal = medal;
}
String getEdition() {
        return edition;
}
void setEdition(String edition) {
        this.edition = edition;
}
String getCity() {
        return city;
}
void setCity(String city) {
        this.city = city;
}

public String getEvent() {
        return event;
}
```

```
	public void setEvent(String event) {
		this.event = event;
	}
}
```