Университет ИТМО

Курсовая работа

по дисциплине: «Системы баз данных»

MongoDB

Выполнили:
студенты III курса
группы 3125
Припадчев Артём
Чурсин Никита
Логунов Илья

Проверит:

Беликов П.А.

Санкт-Петербург

2015

Оглавление

Описание предметной области

Задачей построения базы данных является зачисление абитуриентов в ВУЗы.

До того, как ВУЗ начинает прием документов, он формирует список документов, которые регламентируют прием. Правила приема могут изменяться ежегодно, т.к. список специальностей обучения динамичен, также изменяются законы РФ в части предоставления льгот гражданам при поступлении в ВУЗы.

Разработка базы данных абитуриентов поможет приемной комиссии большую часть работы автоматизировать. В отличии от стеллажей с огромным количеством личных дел, БД позволяет осуществлять мгновенный доступ к информации о любом абитуриенте, а также без ошибок формировать списки людей по различным критериям.

Процесс зачисления в ВУЗ осуществляется в несколько этапов. Начинается всё с приема документов у абитуриента. При приеме документов абитуриент заявляет о своём желании участвовать в конкурсе на конкретную специальность, а если точнее, то на учебную программу. Выбирать специальностей можно несколько, выставляя для каждой свой приоритет. В таком случае, если студент не проходит по конкурсу на специальность с 1 приоритетом, то начинает участвовать в конкурсе на специальности со 2 приоритетом и так далее.

Помимо информации о выбранных специальностях и экзаменах, абитуриент предоставляет некоторые дополнительные сведения (ФИО, паспортные данные, гражданство и так далее), а также (при наличии) документы, которые подтверждают право на льготы.

Следующий этап – зачисление абитуриентов с самым высоким рейтингом. В связи с тем, что по законодательству России выделяются разные категории граждан, имеющих льготы при поступлении, приемная комиссия формирует несколько приказов со списками абитуриентов, зачисленных по определенному нормативному документу. Однако основная часть поступающих не имеет дополнительных льгот и участвует в общем конкурсе. Именно эта категория требует автоматизированной обработки.

Описание базы данных

Что такое NoSQL?

NoSQL (Not only SQL) — это ряд технологий, подходов, проектов направленных на реализацию моделей баз данных, имеющих существенные отличия от традиционных СУБД, работающих с языком SQL. Концепция NoSQL не отрицает SQL, она лишь стремится решить проблемы и вопросы, с которыми недостаточно хорошо справляется РСУБД. Чаще всего данные в NoSQL решении представляются в виде хеш-таблиц, деревьев, документов и пр.

Классические реляционные базы данных страдают проблемами при работе с данными очень большого объема и при высокой нагрузке, поэтому, рано или поздно возникает необходимость распределенных вычислений. Так, например, часто трудно или не реально добиться изолированности, расщеплённости данных в рамках одной БД. Согласно теореме Брюера, в любой реализации распределенных вычислений возможно добиться не более двух их трех свойств: согласованность данных, доступность, устойчивость к разделению.

Концепция NoSQL предоставляет высокую доступность и устойчивость данных к разделению, но при этом не все гладко с согласованностью данных. Этот подход призван решать собственный класс задач.

Что такое MongoDb?

MongoDb — это документо-ориентированная база данных. Т.е. каждая запись – это документ без жестко заданной схемы, который может содержать вложенные документы.

MongoDb хороша высокой скоростью записи / чтения, масштабированностью, но опять же сохранность и целостность данных не так хороша. Так, например, в Монго есть отличная реализация репликации, которая довольно легко устанавливается и настраивается или же шардинг (возможность разнести данные по нескольким серверам), который тоже довольно прост в установке. В комплексе, можно получить систему распределенных вычислений с высокой отказоустойчивостью.

MongoDb использует JSON нотацию для хранения и управления документами, а также свой достаточно удобный язык запросов.
Основные возможности данной СУБД:

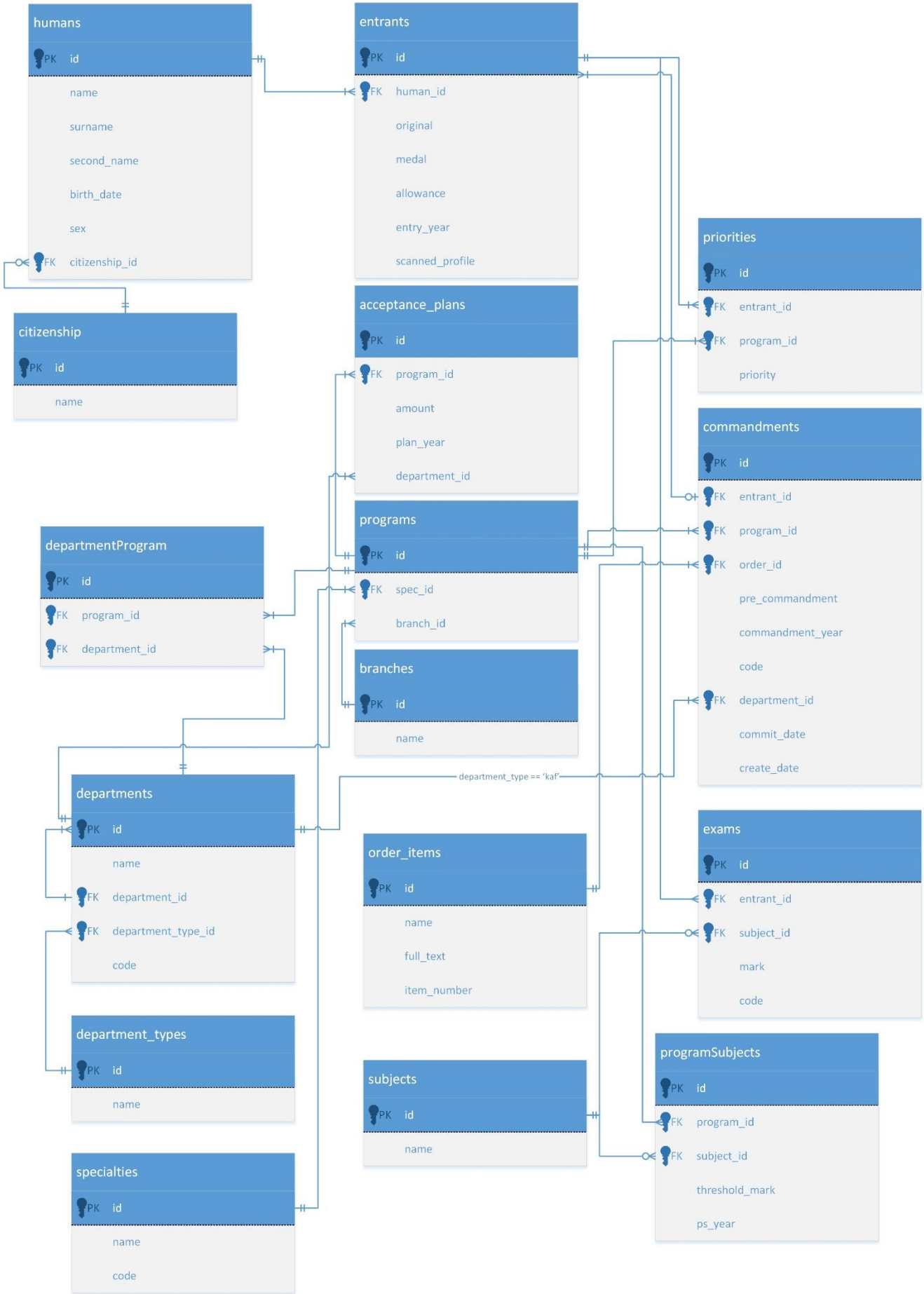
- Документо-ориентированное хранилище (простая и мощная JSON-подобная схема данных)
- Достаточно гибкий язык для формирования запросов
- Динамические запросы
- Полная поддержка индексов
- Профилирование запросов

- Быстрые обновления «на месте»
- Эффективное хранение двоичных данных больших объёмов, напр., фото и видео
- Журналирование операций, модифицирующих данные в БД
- Поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и шардинг
- Может работать в соответствии с парадигмой MapReduce

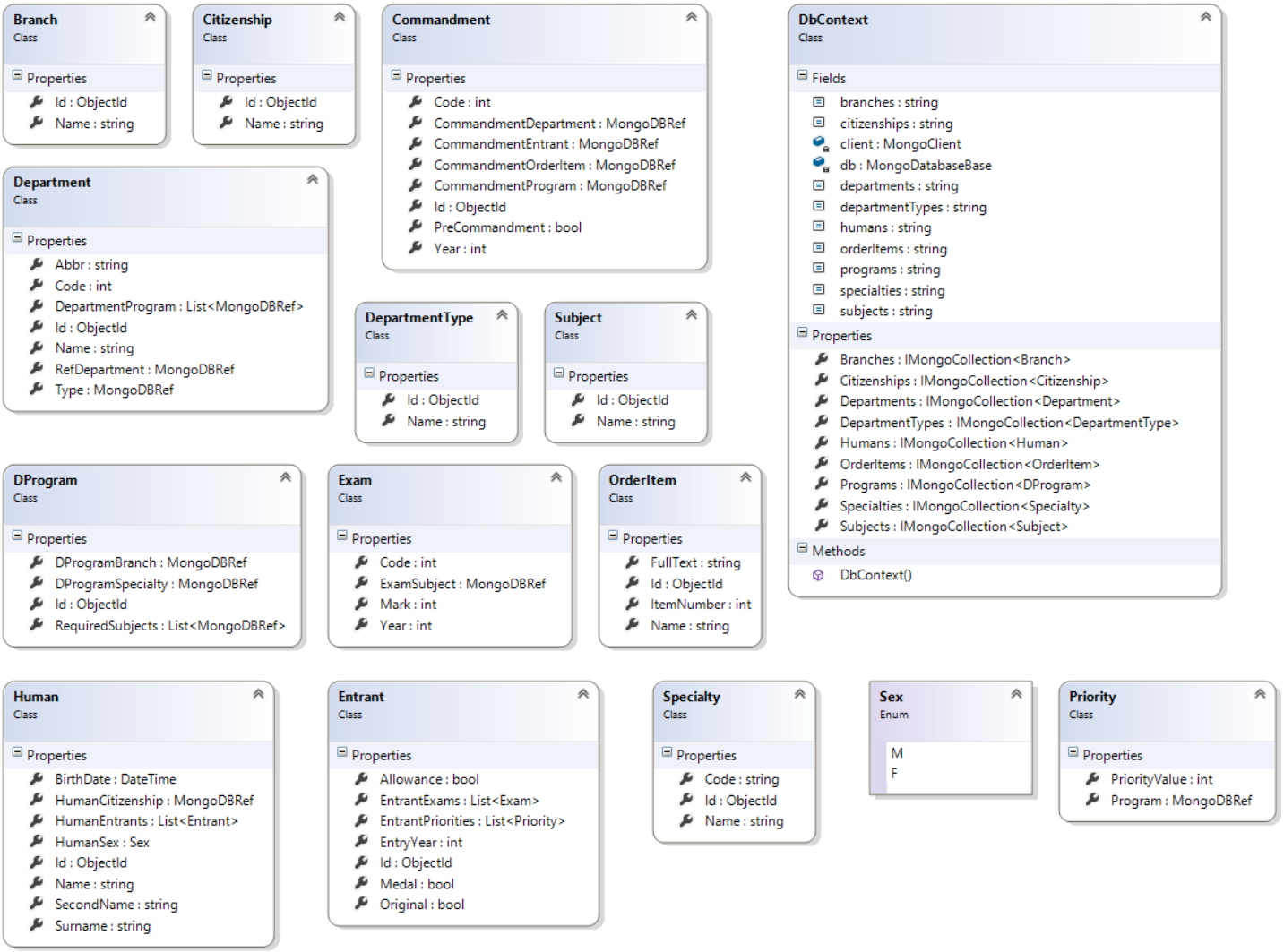
Недостатки:

- Отсутствует оператор «join». Обычно данные могут быть организованы более денормализованным способом, но на разработчиков ложится дополнительная нагрузка по обеспечению непротиворечивости данных.
- Нет такого понятия, как «транзакция». Атомарность гарантируется только на уровне целого документа, т.е. частичное обновление документа произойти не может.
- Отсутствует понятие «изоляции». Любые данные, которые считываются одним клиентом, могут параллельно изменяться другим клиентом.

ER-модель реляционной базы данных



Модель NoSQL базы данных



Создание коллекций и заполнение данными

```
#region Inserts

        static void insertHumans()
        {
            List<DProgram> programs =
context.Programs.Aggregate().ToListAsync().Result;
            var humanCollection = context.Humans;
            for (int i = 0; i < 100000; i++)
            {
                humanCollection.InsertOneAsync(getRandomHuman(programs)).Wait();
            }
        }

        static void insertCitizenships()
        {
            string[] citiz = { "Россия", "Китай", "Казахстан" };
            var col = context.Citizenships;
            foreach (var c in citiz)
                col.InsertOneAsync(new Citizenship { Name = c }).Wait();
        }

        static void insertSubjects()
        {
            string[] subjects = {"Русский язык", "Математика", "Физика",
"Информатика",
                                 "Английский язык", "История", "Обществознание",
"Химия",
                                 "Биология", "Литература"};
            var collection = context.Subjects;
            foreach (var s in subjects)
            {
                collection.InsertOneAsync(new Subject { Name = s }).Wait();
            }
        }

        static void insertBranches()
        {
            string[] branches = { "Дневное", "Вечернее", "Заочное" };
            var collection = context.Branches;
            foreach (var b in branches)
            {
                collection.InsertOneAsync(new Branch { Name = b }).Wait();
            }
        }

        static void insertOrderItems()
        {
            var collection = context.OrderItems;
            collection.InsertOneAsync(new OrderItem { Name = "Победители и призеры
олимпиад", ItemNumber = 1 }).Wait();
            collection.InsertOneAsync(new OrderItem { Name = "Имеющие льготы",
ItemNumber = 2 }).Wait();
            collection.InsertOneAsync(new OrderItem { Name = "Основной прием",
ItemNumber = 3 }).Wait();
        }

        static void insertSpecialties()
        {
```

```
            var collection = context.Specialties;
            collection.InsertOneAsync(new Specialty { Name = "Прикладная математика и
информатика", Code = "01.03.02" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Информатика и
вычислительная техника", Code = "09.03.01" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Прикладная
информатика", Code = "09.03.03" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Программная инженерия",
Code = "09.03.04" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Информационная
безопасность", Code = "10.03.01" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Приборостроение", Code
= "12.03.01" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Оптотехника", Code =
"12.03.02" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Техническая физика",
Code = "16.03.01" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Менеджмент", Code =
"38.03.02" }).Wait();
            collection.InsertOneAsync(new Specialty { Name = "Интеллектуальные
системы в гуманитарной сфере", Code = "38.03.02" }).Wait();
        }

        static void insertDepartmentTypes()
        {
            var collection = context.DepartmentTypes;
            collection.InsertOneAsync(new DepartmentType { Name = "Университет"
}).Wait();
            collection.InsertOneAsync(new DepartmentType { Name = "Факультет"
}).Wait();
            collection.InsertOneAsync(new DepartmentType { Name = "Кафедра"
}).Wait();
        }

        static void insertPrograms()
        {
            var spec = context.Specialties;
            List<Specialty> specialties = spec.Find(x => x.Code !=
"").ToListAsync().Result;

            var programs = context.Programs;
            foreach (var specialty in specialties)
            {
                if (Array.Exists(new[] { "12.03.01", "16.03.01", "01.03.02",
"12.03.02" }, x => x == specialty.Code))
                {
                    programs.InsertManyAsync(getProgramsForInsert(specialty, new[] {
"Физика", "Математика", "Русский язык" })).Wait();
                }
                else
                    if (Array.Exists(new[] { "09.03.01", "09.03.04", "10.03.01",
"09.03.03" }, x => x == specialty.Code))
                    {
                        programs.InsertManyAsync(getProgramsForInsert(specialty,
new[] { "Информатика", "Математика", "Русский язык" })).Wait();
                    }
                    else
                        if (Array.Exists(new[] { "38.03.02", "39.03.02" }, x => x ==
specialty.Code))
                        {
```

```
                            programs.InsertManyAsync(getProgramsForInsert(specialty,
new[] { "Обществознание", "Математика", "Русский язык" })).Wait();
                        }
            }
        }

        static void insertDepartments()
        {
            var dep = context.Departments;
            dep.InsertOneAsync(new Department
                {
                    Name = "Университет ИТМО",
                    Type = new MongoDBRef(DbContext.departmentTypes,
                        context.DepartmentTypes.Find(x => x.Name ==
"Университет").FirstAsync().Result.Id),
                    Code = 777,
                    Abbr = "ИТМО"
                }).Wait();

            //Faculties
            var id = context.Departments.Find(x => x.Code ==
777).FirstAsync().Result.Id;
            var type = context.DepartmentTypes.Find(x => x.Name ==
"Факультет").FirstAsync().Result.Id;
            dep.InsertOneAsync(getFaculty("Естественно научный факультет", 100,
"ЕНФ", id, type)).Wait();
            dep.InsertOneAsync(getFaculty("Факультет информационных технологий и
программирования", 150, "ФИТиП", id, type)).Wait();
            dep.InsertOneAsync(getFaculty("Факультет компьютерных технологий и
управления", 200, "ФКТиУ", id, type)).Wait();
            dep.InsertOneAsync(getFaculty("Факультет фотоники и оптоинформатики",
250, "ФФиОИ", id, type)).Wait();
            dep.InsertOneAsync(getFaculty("Факультет оптико-информационных систем и
технологий", 300, "ФОИСТ", id, type)).Wait();
            dep.InsertOneAsync(getFaculty("Инженерно физический факультет", 350,
"ИФФ", id, type)).Wait();

            //Chairs
            type = context.DepartmentTypes.Find(x => x.Name ==
"Кафедра").FirstAsync().Result.Id;
            dep.InsertManyAsync(
                new[] {
                        getChair("Кафедра Высшей математики - 1",
                            new[] {"01.03.02"},
                            "ЕНФ",type,101,"ВМ-1"
                            ),
                        getChair("Кафедра компьютерных технологий",
                            new[] {"01.03.02"},
                            "ФИТиП",type,151,"КТ"
                            ),
                        getChair("Кафедра вычислительной техники",
                            new[] {"09.03.01","10.03.01"},
                            "ФКТиУ",type,201,"ВТ"
                            ),
                        getChair("Кафедра компьютерной фотоники и видеоинформатики",
                            new[] {"09.03.03","12.03.02"},
                            "ФФиОИ",type,251,"КФиВИ"
                            ),
                        getChair("Кафедра информационных систем",
                            new[] {"09.03.03"},
                            "ФИТиП",type,152,"ИС"
```

```
                        ),
                    getChair("Кафедра информатики и прикладной математики",
                        new[] {"09.03.04"},
                        "ФКТиУ",type,202,"ИПМ"
                        ),
                    getChair("Кафедра безопасных информационных технологий",
                        new[] {""},
                        "ФКТиУ",type,203,"БИТ"
                        ),
                    getChair("Кафедра интеллектуальных технологий в гуманитарной
сфере",
                        new[] {"45.03.04"},
                        "ЕНФ",type,102,"ИТГС"
                        )
                    }).Wait();

        }

        #endregion

        #region Get some data for inserts

        static IEnumerable<ObjectId> getProgramIdBySpecialtyCode(string code)
        {
            if (code == "") return null;
            var specId = context.Specialties.Find(s => s.Code ==
code).FirstAsync().Result.Id;
            var programsId = context.Programs.Find(p => p.DProgramSpecialty.Id ==
specId).ToListAsync().Result;
            return programsId.Select(p => p.Id);
        }

        static Department getChair(string Name, IEnumerable<string> specCodes, string
facultyAbbr, ObjectId typeId, int code, string abbr)
        {
            List<MongoDBRef> departmentPrograms = new List<MongoDBRef>();
            foreach (var specCode in specCodes)
            {
                try
                {
                    foreach (var pCode in getProgramIdBySpecialtyCode(specCode))
                    {
                        departmentPrograms.Add(new MongoDBRef(DbContext.programs,
pCode));
                    }
                }
                catch (Exception e)
                {
                    departmentPrograms = null;
                }
            }
            return new Department
            {
                Name = Name,
                DepartmentProgram = departmentPrograms,
                RefDepartment = new MongoDBRef(DbContext.departments,
context.Departments.Find(d => d.Abbr == facultyAbbr).FirstAsync().Result.Id),
                Type = new MongoDBRef(DbContext.departmentTypes, typeId),
                Code = code,
                Abbr = abbr
            };
```

```
        }

        static Department getFaculty(string Name, int code, string abbr, ObjectId
refDepId, ObjectId typeId)
        {
            return new Department
            {
                Name = Name,
                RefDepartment = new MongoDBRef(DbContext.departments, refDepId),
                Type = new MongoDBRef(DbContext.departmentTypes, typeId),
                Code = 100,
                Abbr = abbr
            };
        }

        static DProgram[] getProgramsForInsert(Specialty specialty, string[] subj)
        {
            DProgram[] pr = new DProgram[3];
            List<MongoDBRef> reqSubjects = new List<MongoDBRef>();

            foreach (var s in subj)
            {
                reqSubjects.Add(new MongoDBRef(DbContext.subjects,
context.Subjects.Find(x => x.Name == s).FirstAsync().Result.Id));
            }


            pr[0] = new DProgram
            {
                DProgramSpecialty = new MongoDBRef(DbContext.specialties,
specialty.Id),
                RequiredSubjects = reqSubjects,
                DProgramBranch = new MongoDBRef(DbContext.branches,
context.Branches.Find(x => x.Name == "Дневное").FirstAsync().Result.Id)
            };
            pr[1] = new DProgram
            {
                DProgramSpecialty = new MongoDBRef(DbContext.specialties,
specialty.Id),
                RequiredSubjects = reqSubjects,
                DProgramBranch = new MongoDBRef(DbContext.branches,
context.Branches.Find(x => x.Name == "Вечернее").FirstAsync().Result.Id)
            };
            pr[2] = new DProgram
            {
                DProgramSpecialty = new MongoDBRef(DbContext.specialties,
specialty.Id),
                RequiredSubjects = reqSubjects,
                DProgramBranch = new MongoDBRef(DbContext.branches,
context.Branches.Find(x => x.Name == "Заочное").FirstAsync().Result.Id)
            };

            return pr;
        }

        static IEnumerable<Exam> getAllRandomExams()
        {
            List<Exam> exams = new List<Exam>();
            int year = gen.Next(2010, 2015);
            foreach (var s in context.Subjects.Aggregate().ToListAsync().Result)
            {
```

```
                int mark = gen.Next(30, 100);
                Exam exam = new Exam
                {
                    ExamSubject = new MongoDBRef(DbContext.subjects, s.Id),
                    Mark = mark,
                    Year = year,
                    Code = gen.Next(100000000, 900000000) + year + mark
                };
                exams.Add(exam);
            }
            return exams;
        }

        static IEnumerable<Priority> getRandomPriorities(IEnumerable<DProgram>
programs)
        {
            List<Priority> priorities = new List<Priority>();
            priorities.AddRange(new[]
                {
                    new Priority
                    {
                        Program = new
MongoDBRef(DbContext.programs,programs.ElementAt(gen.Next(programs.Count())).Id),
                        PriorityValue = 1
                    },
                    new Priority
                    {
                        Program = new
MongoDBRef(DbContext.programs,programs.ElementAt(gen.Next(programs.Count())).Id),
                        PriorityValue = 2
                    },
                    new Priority
                    {
                        Program = new
MongoDBRef(DbContext.programs,programs.ElementAt(gen.Next(programs.Count())).Id),
                        PriorityValue = 3
                    }
                });

            return priorities;
        }

        static Human getRandomHuman(IEnumerable<DProgram> programs)
        {
            List<Entrant> entrants = new List<Entrant>();
            entrants.Add(getRandomEntrant(programs));
            Human human = new Human
            {
                Name = GetRandomTextLine(textCh, 6),
                Surname = GetRandomTextLine(textCh, 6),
                SecondName = GetRandomTextLine(textCh, 6),
                BirthDate = RandomDay(),
                HumanSex = (gen.Next(2) == 0) ? Sex.M : Sex.F,
                HumanCitizenship = new MongoDBRef(DbContext.citizenships,
                    context.Citizenships.Find(c => c.Name ==
"Россия").FirstAsync().Result.Id),
                HumanEntrants = entrants
            };
            return human;
        }
```

```
static Entrant getRandomEntrant(IEnumerable<DProgram> programs)
{
    List<Exam> exams = getAllRandomExams().ToList<Exam>();
    Entrant entrant = new Entrant
    {
        Id = ObjectId.GenerateNewId(DateTime.Now),
        Original = (gen.Next(2) == 0),
        Medal = (gen.Next(2) == 1),
        Allowance = (gen.Next(2) == 1),
        EntrantExams = exams,
        EntrantPriorities = getRandomPriorities(programs).ToList<Priority>(),
        EntryYear = exams.ElementAt(0).Year
    };
    return entrant;
}

#endregion
```

Зачетные задания

Задание 1.

```
        const string English =
"qwertyuiop[]asdfghjkl;'zxcvbnm,.QWERTYUIOP{}ASDFGHJKL:\"ZXCVBNM<>?";
        const string Russian =
"йцукенгшщзхъфывапролджэячсмитьбюЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ,";

//Task 1*
        static string convertEngToRus(string input)
        {
            var result = new StringBuilder(input.Length);
            int index;
            foreach (var symbol in input)
                result.Append((index = English.IndexOf(symbol)) != -1 ? Russian[index] :
symbol);
            return result.ToString();
        }

        static void getAggregateData()
        {
            var humanCollection = context.Humans;
            var subjectCollection = context.Subjects;

            List<Human> humans =
humanCollection.Aggregate().Skip(100).Limit(3).ToListAsync().Result;
            foreach (var human in humans)
            {
                Console.WriteLine("Name:\t" + human.Name);
                Console.WriteLine("Surname:\t" + human.Surname);
                Console.WriteLine("Second name:\t" + human.SecondName);

                Console.WriteLine("===== Entrants =====");
                foreach (var entrant in human.HumanEntrants)
                {
                    Console.WriteLine("Year: " + entrant.EntryYear);
                    Console.Write("Exams: ");
                    foreach (var exam in entrant.EntrantExams)
                    {
                        Console.Write(context.Subjects.Find(s => s.Id ==
exam.ExamSubject.Id).FirstAsync().Result.Name + ":");
                        Console.Write(exam.Mark + "\t");
                    }
                }
                Console.WriteLine();
                Console.WriteLine();
            }
        }
static List<Human> getHumansWithRegex()
        {
            var humansCollection = context.Humans;

            Regex regex = new Regex(".*fu.*");

            var humans = humansCollection.Aggregate().Match(c =>
regex.IsMatch(c.Name)).Limit(10).ToListAsync().Result;

            return humans;
        }
```

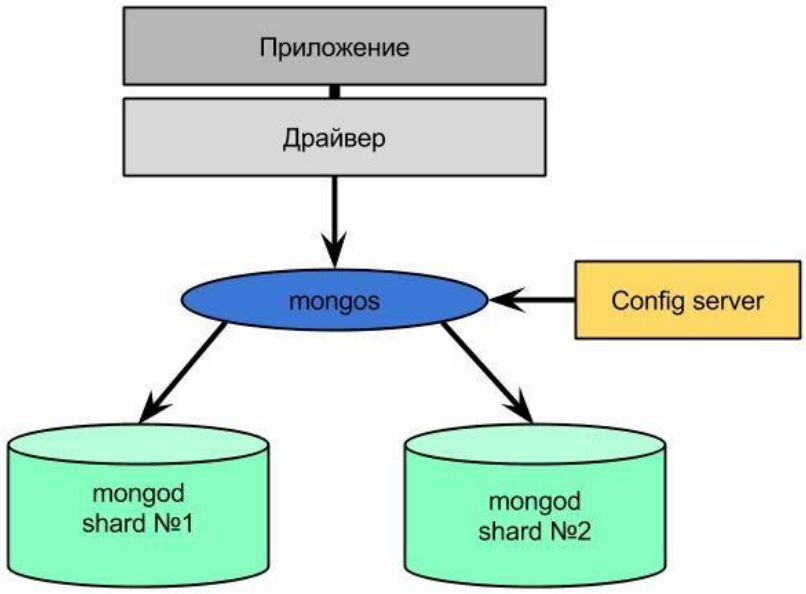
Задание 2.

```
//Task 2**
        static void getDataWithFuzzySearch()
        {
            string line1 = "Расия";
            string line2 = "Россия";

            Console.WriteLine("Levenshtein distance between {0} and {1}: {2}",
                line1, line2, FuzzySearch.LevenshteinDistance(line1, line2));

            Console.WriteLine("Jaro Winkler distance between {0} and {1}: {2}",
                line1, line2, FuzzySearch.JaroWinklerDistance(line1, line2));

            line1 = "Рассия";
            Console.WriteLine("Hamming distance between {0} and {1}: {2}",
                line1, line2, FuzzySearch.GetHammingDistance(line1, line2));

            Console.WriteLine();

            var humans = getHumansWithRegex();

            double maxMatch = 0;
            string line = "sradfu";
            Dictionary<string, double> matches = new Dictionary<string, double>();
            foreach (var human in humans)
            {
                double match = FuzzySearch.JaroWinklerDistance(human.Name, line);
                if (maxMatch < match) maxMatch = match;
                matches.Add(human.Name, match);
            }

            foreach(var d in matches)
                Console.WriteLine("Name: {0} distance: {1}",d.Key,d.Value);
        }
public static class FuzzySearch
    {
        public static int LevenshteinDistance(string string1, string string2)
        {
            if (string1 == null) throw new ArgumentNullException("string1");
            if (string2 == null) throw new ArgumentNullException("string2");
            int diff;
            int[,] m = new int[string1.Length + 1, string2.Length + 1];

            for (int i = 0; i <= string1.Length; i++) m[i, 0] = i;
            for (int j = 0; j <= string2.Length; j++) m[0, j] = j;

            for (int i = 1; i <= string1.Length; i++)
                for (int j = 1; j <= string2.Length; j++)
                {
                    diff = (string1[i - 1] == string2[j - 1]) ? 0 : 1;

                    m[i, j] = Math.Min(Math.Min(m[i - 1, j] + 1,
                                               m[i, j - 1] + 1),
                                               m[i - 1, j - 1] + diff);
                }

            return m[string1.Length, string2.Length];
        }

        /* The Winkler modification will not be applied unless the
     * percent match was at or above the mWeightThreshold percent
     * without the modification.
     * Winkler's paper used a default value of 0.7
```

```
    */
        private static readonly double mWeightThreshold = 0.7;
        /* Size of the prefix to be concidered by the Winkler modification.
         * Winkler's paper used a default value of 4
         */
        private static readonly int mNumChars = 4;

        public static double JaroWinklerDistance(string string1, string string2)
        {
            //return 1.0 - proximity(string1, string2);
            return proximity(string1, string2);
        }

        private static double proximity(string aString1, string aString2)
        {
            int lLen1 = aString1.Length;
            int lLen2 = aString2.Length;
            if (lLen1 == 0)
                return lLen2 == 0 ? 1.0 : 0.0;

            int lSearchRange = Math.Max(0, Math.Max(lLen1, lLen2) / 2 - 1);

            bool[] lMatched1 = new bool[lLen1];
            for (int i = 0; i < lMatched1.Length; i++)
            {
                lMatched1[i] = false;
            }
            bool[] lMatched2 = new bool[lLen2];
            for (int i = 0; i < lMatched2.Length; i++)
            {
                lMatched2[i] = false;
            }

            int lNumCommon = 0;
            for (int i = 0; i < lLen1; ++i)
            {
                int lStart = Math.Max(0, i - lSearchRange);
                int lEnd = Math.Min(i + lSearchRange + 1, lLen2);
                for (int j = lStart; j < lEnd; ++j)
                {
                    if (lMatched2[j]) continue;
                    if (aString1[i] != aString2[j])
                        continue;
                    lMatched1[i] = true;
                    lMatched2[j] = true;
                    ++lNumCommon;
                    break;
                }
            }
            if (lNumCommon == 0) return 0.0;

            int lNumHalfTransposed = 0;
            int k = 0;
            for (int i = 0; i < lLen1; ++i)
            {
                if (!lMatched1[i]) continue;
                while (!lMatched2[k]) ++k;
                if (aString1[i] != aString2[k])
                    ++lNumHalfTransposed;
                ++k;
            }
            // System.Diagnostics.Debug.WriteLine("numHalfTransposed=" +
numHalfTransposed);
            int lNumTransposed = lNumHalfTransposed / 2;
```

```
            // System.Diagnostics.Debug.WriteLine("numCommon=" + numCommon + "
numTransposed=" + numTransposed);
            double lNumCommonD = lNumCommon;
            double lWeight = (lNumCommonD / lLen1
                              + lNumCommonD / lLen2
                              + (lNumCommon - lNumTransposed) / lNumCommonD) / 3.0;

            if (lWeight <= mWeightThreshold) return lWeight;
            int lMax = Math.Min(mNumChars, Math.Min(aString1.Length, aString2.Length));
            int lPos = 0;
            while (lPos < lMax && aString1[lPos] == aString2[lPos])
                ++lPos;
            if (lPos == 0) return lWeight;
            return lWeight + 0.1 * lPos * (1.0 - lWeight);

        }

        public static int GetHammingDistance(string source, string target)
        {
            if (source.Length != target.Length)
            {
                throw new Exception("Strings must be equal length");
            }

            int distance =
                source.ToCharArray()
                .Zip(target.ToCharArray(), (c1, c2) => new { c1, c2 })
                .Count(m => m.c1 != m.c2);

            return distance;
        }
    }
```

Задание 3.




```
mongod --configsvr --dbpath /mongod/config --port 27002
mongos --configdb json.cf:27002 --port 27100

mongo --port 27100

sh.addShard("json.cf:27017")
sh.addShard("logiv.koding.io:27017")
sh.addShard("artemvirused.koding.io:27017")


use admin
sh.enableSharding("test")

use test
db.humans.ensureIndex({_id:1})
use admin
db.runCommand({shardCollection: "test.humans", key: {_id: 1}})

db.runCommand( { movePrimary : "test", to : "shard0000" } )
db.runCommand( { removeShard: "shard0001" } )
db.printShardingStatus()


MongoDB shell version: 3.0.2
connecting to: 127.0.0.1:27100/test
Server has startup warnings:
2015-04-27T16:21:21.338+0300 I CONTROL  ** WARNING: You are running this process as
the root user, which is not recommended.
2015-04-27T16:21:21.338+0300 I CONTROL
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("553d31ab51a440c86a8487e1")
}
  shards:
```

```
        {  "_id" : "shard0000",  "host" : "json.cf:27017" }
        {  "_id" : "shard0002",  "host" : "logiv.koding.io:27017" }
        {  "_id" : "shard0003",  "host" : "artemvirused.koding.io:27017" }
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                9 : Success
                71 : Failed with error 'moveChunk failed to engage TO-shard in the
data transfer: exception: remote client json.cf:34830 tried to initialize this host
as shard shard0003, but shard name was previously initi
alized as shard0001', from shard0000 to shard0003
                1 : Failed with error '_recvChunkCommit failed!', from shard0001 to
shard0002
  databases:
        {  "_id" : "admin",  "partitioned" : false,  "primary" : "config" }
        {  "_id" : "test",  "partitioned" : true,  "primary" : "shard0000" }
                test.humans
                        shard key: { "_id" : 1 }
                        chunks:
                                shard0000       2
                                shard0002       2
                                shard0003       2
                        { "_id" : { "$minKey" : 1 } } -->> { "_id" :
ObjectId("553ce7e6712cb11db8b976c3") } on : shard0003 Timestamp(10, 0)
                        { "_id" : ObjectId("553ce7e6712cb11db8b976c3") } -->> { "_id"
: ObjectId("553ce843712cb11db8b9f7c7") } on : shard0003 Timestamp(9, 0)
                        { "_id" : ObjectId("553ce843712cb11db8b9f7c7") } -->> { "_id"
: ObjectId("553ce89f712cb11db8ba78cb") } on : shard0000 Timestamp(9, 1)
                        { "_id" : ObjectId("553ce89f712cb11db8ba78cb") } -->> { "_id"
: ObjectId("553ce8fb712cb11db8baf9cf") } on : shard0002 Timestamp(10, 1)
                        { "_id" : ObjectId("553ce8fb712cb11db8baf9cf") } -->> { "_id"
: ObjectId("553ce956712cb11db8bb7ad3") } on : shard0000 Timestamp(7, 0)
                        { "_id" : ObjectId("553ce956712cb11db8bb7ad3") } -->> { "_id"
: { "$maxKey" : 1 } } on : shard0002 Timestamp(8, 0)
```

Использованная литература

1. http://habrahabr.ru/search/?q=Mongodb
2. http://docs.mongodb.org/manual/
3. http://jsman.ru/mongo-book/
4. https://www.google.com