Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Метелева Алиса Андреевна

# Сравнение производительности PostgreSQL и MongoDB СУБД

Курсовая работа

Научный руководитель:
д. ф.-м. н., профессор Новиков Б. А.

Санкт-Петербург
2016

# Оглавление

# Введение

В современном мире информационных технологий базы данных используются повсеместно: от приложений вроде записных книжек, до интернет-магазинов.

Существуют два наиболее распространенных типа систем управления данными: реляционные и NoSQL СУБД, которые отличаются в таких аспектах работы, как надежность, гибкость, согласованность данных и масштабируемость.

В последнее время NoSQL подход набирает все большую популярность [7], которая частично обуславливается тем, что решения NoSQL отличаются проектированием с ориентацией на дальнейшее масштабирование и обладают некоторыми полезными свойствами, такими как возможность применения различных типов хранилищ, возможность разработки базы данных без задания схемы, линейная масштабируемость базы (добавление процессоров увеличивает производительность).

Благодаря таким преимуществам, мы получаем приложения, использующие NoSQL СУБД(к примеру MongoDB, Reddis, Cassandra) даже там, где реляционный подход были бы эффективнее, так как, при выборе СУБД важно ориентироваться не только на структуру базы данных, но и на то, какие задачи мы будем ставить перед ней, то есть обработка каких запросов нам будет нужна.

Цель данной работы - сравнить два типа СУБД по производительности, но не на уровне самих систем, тестируя одну и ту же функциональность для рСУБД и для NoSQL, а на уровне задач, рассматривая разные способы выполнения запросов и разные схемы баз данных и выбирая лучшие случаи, чтобы выяснить, в какой системе определенные задачи решаются эффективнее наилучшим образом,

# 1. Постановка задачи

Задача состоит в сравнении производительности систем управления баз данных, опирающихся на SQL и NoSQL подходы для конкретного класса задач.

Для данной курсовой работы в качестве NoSQL СУБД рассматривается MongoDB 3.2[5], в качестве SQL СУБД - PostgreSQL 9.5[6].

Ниже представлена примерная схема базы данных(рис. 1) , включающая в себя информацию о группах, предметах и результатах студентов за промежуточные аттестации.

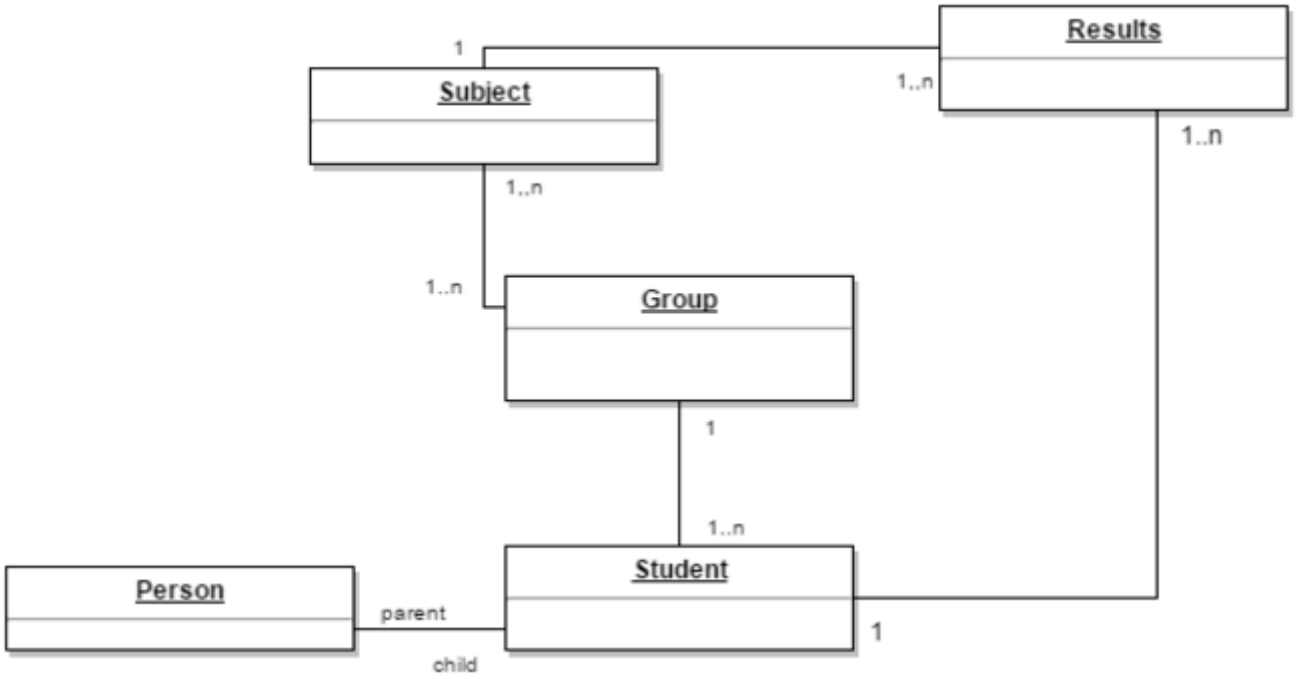



Рис. 1: Схема БД

Реализовав данную схему, можно рассмотреть пласт типичных задач. В данной работе это измерение времени вставки и обновления информации, а также задачи, реализация которых требует просмотра всей базы данных при выполнении аналитических запросов:

- вывести студентов,у которых студента у которого 3 по наименее преподаваемому предмету;
- вывести студентов - отличников/хорошистов;

- вывести группы, средний бал по которым больше 4.2;
- вывести группы, в которых больше всего студентов-хорошистов;
- вывести наименование наименее преподаваемого предмета;
- вывести телефоны всех студентов, с именем Dolly;
- вывести группы, id которых меньше одного числа, но больше другого.

Всего в базах располагается информация о 500 000 студентов, 25 000 предметов, 25 000 групп, 4 000 000 результатов для базы данных размеров 812 Мб.

# 2. Обзор

В ходе изучения темы, было выявлено, что сравнение производительности уже проводились для MongoDB и PostgreSQL, но в отличии от нашей работы это было сравнение на уровне самих систем, а не на уровне задач. В этих сравнениях рассматривали одни и те же функции для PostgreSQL и MongoDB, возможно, игнорируя более удачные способы расположения информации в Mongo и иные варианты написания запросов. Стоит также отметить, что одно из исследований проводилось разработчиками PostgreSQL, поэтому есть вероятность того, что они не были целиком беспристрастны. Тогда как в нашей работе мы не отталкиваемся от наших симпатий, и не задаемся целью выставить ту или иную СУБД в худшем, чем она есть свете. Но, тем не менее, я считаю, что результаты анализов стоит привести, чтобы предположить чего стоит ожидать. Для PostgreSQL 8.4.13 и MongoDB 2.4.3 в результате тестирования производительности были получены следующие графики в 2013 году[3] (рис. 2) :

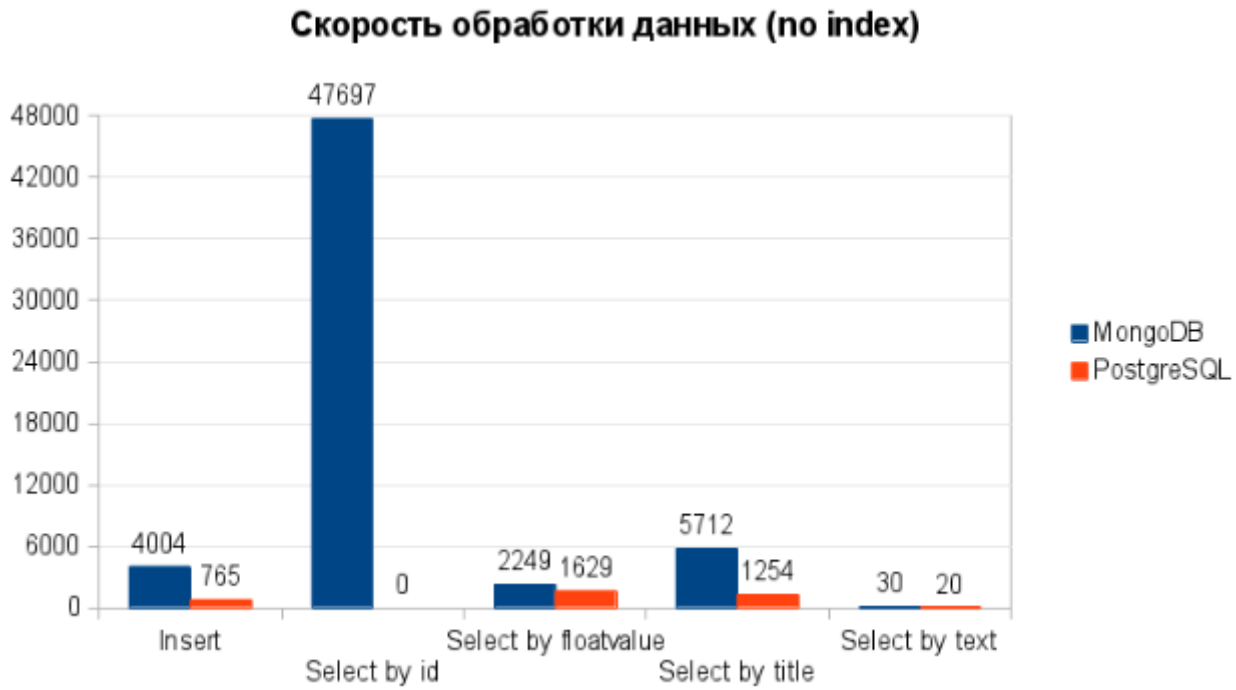



Рис. 2: Выпонение запросов 2013г

Сравнительно недавно были получены следующие результаты компанией Enterprise[2] (рис. 3).

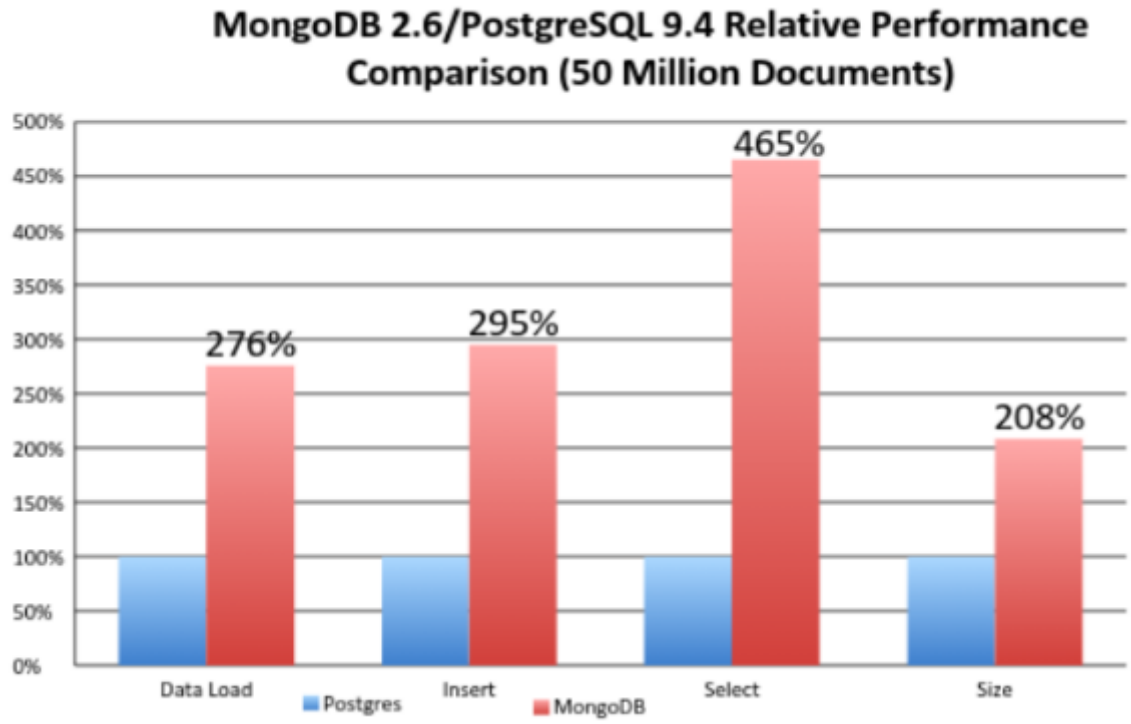



Рис. 3: Результаты Enterprise

Также в 2014 году для PostgreSQL 9.1 и MongoDB 2.4.9 студенты Universidade Regionaldo Noroeste do Estado do Rio Grande do Sul (UNIJU´I) [1] получили устаревшие, на данный момент результаты, которые были получены при измерении времени выполнения простых и сложных запросов, ставя в приоритете сравнение на уровне самих СУБД, а не задач, которые перед ними ставят(рис. 4, рис. 5, рис. 6).

Получаем, что за последнее время PostgreSQL догнала и перегнала MongoDB на тестах. Стоит помнить, что сравнения производились на простейших схемах БД, и рано делать выводы о том, что PostgreSQL покажет себя так же хорошо, как и на графике компании Enterprise, на более реальном примере со сложной структурой данных. Более вероятно, что результаты будут более сопоставимыми.

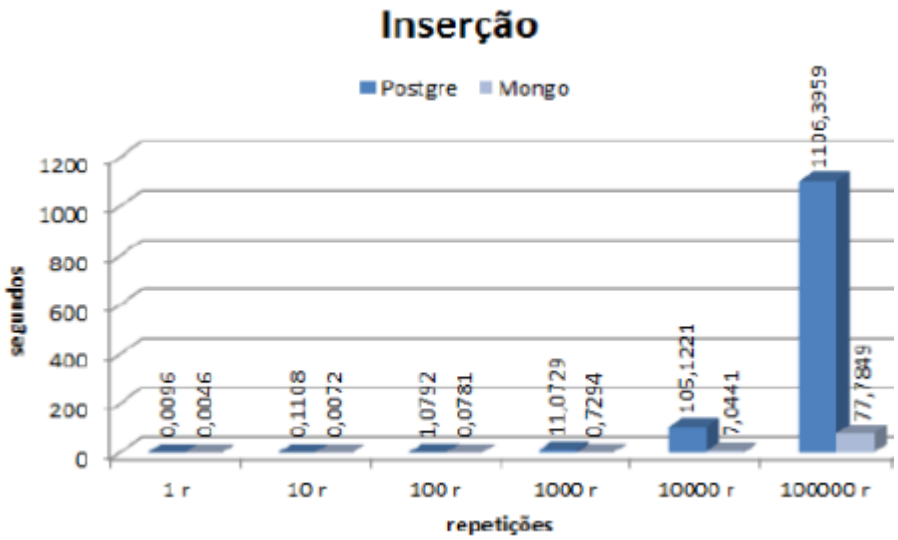



Рис. 4: Результаты UNIJU´I времени вставки

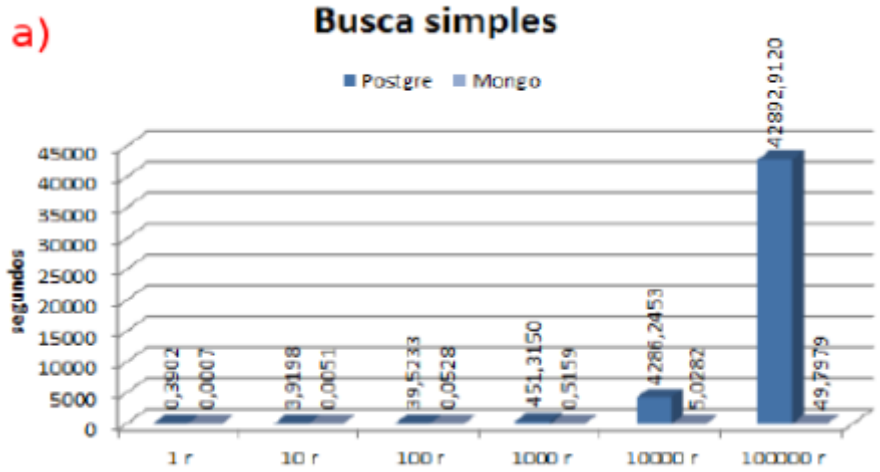



Рис. 5: Результаты UNIJU´I выполнения простых запросов

# 3. Инструменты

В данном параграфе определяются выбранные ранее СУБД :MongoDB и PostgreSQL, а также вспомогательные инструменты, необходимые для генерации данных и удобного пользование СУБД.

- **MongoDB 3.0**
  Документо-ориентированная система управления базами данных. Хранит данные в виде коллекций документов, состоящих из набора полей. Этот набор может различаться в документах одной коллекции благодаря «бессхемности» таких СУБД. Допускаются

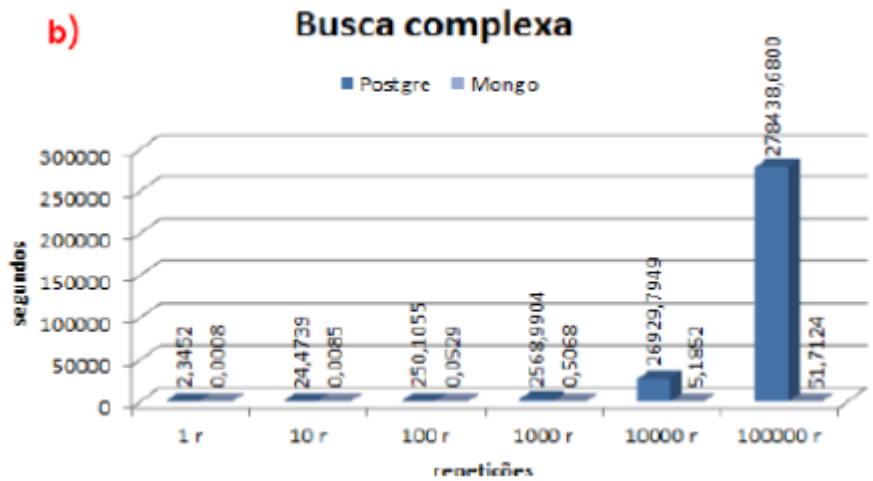



Рис. 6: Результаты UNIJU´I выполнения более сложных запросов

вложенные документы и сложные типы значений полей (массивы, ссылки и т.п.). СУБД управляет наборами JSON-подобных документов, хранимых в двоичном виде в формате BSON.

- **PostgreSQL 9.4.5**
  Объектно-реляционная система управления базами данных, базируется на языке SQL и поддерживает многие из возможностей стандарта SQL.

- **MongoChef**
  MongoDB GUI. Позволяет редактировать любые поля любого типа, включена подсветка синтаксиса, быстрый поиск по базе данных, управление пользователями и ролями. В работе использовалась в качестве удобного GUI

- **Ruby-gem Faker**.
  **RubyGems** — система управления пакетами для языка программирования Руби, который предоставляет стандартный формат для программ и библиотек Руби (“gems”), инструменты, предназначенные для простого управления установкой «gems», и сервер для их

распространения.

**Faker gem** - gem, предназначенный для генерирования случайных данных, возвращаемым значениям не гарантируется уникальность.

Библиотека была нужна для генерирования данных в таблицы PostgreSQL и коллекции MongoDB.

# 4. Реализация

В ходе выполнения курсовой работы была разработана схема базы данных, которая была реализована в виде таблиц для PostgreSQL (см. Приложение 1). В MongoDB были созданы коллекции (см. Приложение 2), в документах которых были размещены данные для хранения той же самой информации. Для более тщательного исследования были заполнены 3 варианта базы для MongoDB, чтобы учесть вероятность того, что на разных схемах какие-то запросы будут давать лучшие результаты. Отметим также, что сравнение производительности выполнялось для баз данных размерами 812 Мб, 1.5 Гб и 2Гб.

## 4.1. Заполнение базы данных в PostgreSQL

После создания таблиц в PostgreSQL потребовалось сгенерировать и загрузить в них данные. Для генерации данных использовался Faker gem. Небольшое приложение, написанное на Ruby загружало данные в файлы с расширением sql, которые затем загружались в базу. Для выгрузки в базу использовалась команда в консоли

**psql -h localhost -U postgres -d study -f 'C:/filepath/file.sql'**

Эта команда подключается к базе study и выполняет скрипт file.sql. В скрипте написаны команды для вставки:

INSERT INTO table VALUES(),(),()...();

## 4.2. Заполнение баз данных в MongoDB

Аналогичным образом создаются данные и заполняются коллекции в MongoDB. Но в отличии от PostgreSQL (где мы заполянем 7 таблиц), здесь нам нужно сгенерировать данные только для 3х, 2х или 1 коллекции. А так как MongoDB предоставляет нам возможность вложения одних документов в другие, то в целом мы вставляем меньшее количество записей, хотя и большего объема.

Сгенерированные данные сохраняются в документ формата .js. В нем прописаны строки в формате:

db.collection.insert([{},{},{},...{}]).

Следует отметить, что для заполнения базы в MongoDB мы не должны вручную создавать коллекции. Если они не созданы, то они создаются при вызове db.collection.insert().

Загрузка данных происходит через консоль, когда мы прописываем в mongo-shell команду

load('C:/filepath/file.js');

Изначально был выбор между тем, загружать данные из файла, в котором за одну команду Insert происходит вставка n записей или загружать данные из файла, в котором на n записей приодится n Insert. Практика показала, что первый вариант работает в разы быстрее.

## 4.3. Запросы

В рамках поставленной задачи были придуманы и реализованы запросы, требующие просмотра всей базы данных. Замеры проходили на "горячей" базе, так как проверка эффективности на таких запросах менее трудоемка, чем на "холодной", которая требует перезапуска СУБД после каждого раза, и более приближена к жизни.

Ниже перечислены запросы, результаты сравнения которых можно увидеть в следующем параграфе данной работы.Для примера код представлен только для запроса 7, как наиболее полно показывающего структуру запросов и способы их реализации для MonogDB.

- Insert
    1. db.collection.insert({},...)
    2. insert into tablename values(value1,value2,...)
- Select

1. Вывести отличников
2. Вывести название наименее преподаваемого предмета (предмета, у которого меньше всего часов в базе)
3. Посчитать количество групп, у которых id $>$ N или $<$ M( N и M - натуральные)
4. Вывести студентов, с именем ‘Dolly’
5. Вывести студентов, у которых 3 по наименее преподаваемому предмету
6. Вывести группы, в которой средний бал учеников $>$ 4.2
7. Вывести группу в которой больше всего студентов со средним баллом $>=$ 4

1. Запрос для MongoDB базы данных из трех коллекций(см. Приложение):

```
gr_ids = db.student.aggregate(
  {$unwind: "$student.results"},
  {$group: {
          _id: {id: "$_id",
          gr_id: "$student.groups_id"},
          mark: {$avg: "$student.results.mark"}}},
  {$match:  {mark: {$gt: 4}}},
  {$group: {_id: "$_id.gr_id",
            students: {$sum: 1}}}).
  result.map(function(d){
    return parseInt(d._id)});

db.groups.find({_id: {$in: gr_ids}})
```

2. Запрос для MongoDB базы данных из двух коллекций(см. Приложение 2.б):

```
db.student.aggregate({$unwind: "$student_results"},
  {$group: {
        _id: "$_id" ,
        gr: {$addToSet: "$groups_id"},
        mark: {$avg: "$student_results.mark"}}},
  {$match: {"mark": {$gte: 4.0}}},
  {$group: {_id:"$gr",
            count: {$sum: 1}}},
  {$sort: {count: -1}},
  {$limit: 1})
```

3. Запрос для MongoDB базы данных из одной коллекции(см. Приложение 2.в):

```
var map = function(){
  for (var s = 0; s < this.gr.syl.subj.length; s++)
  {
    var key = {key : this._id, gr : this.gr.gr_id}
    var value = {
      count : 1,
          group : this.gr.gr_id,
      mark  : this.gr.syl.subj[s].mark
        };
    emit(key,value);
  }
};

var reduce = function(key,countObjVals){
  reducedVal = {count : 0, mark : 0};
  for (var i = 0; i < countObjVals.length; i++)
  {
    reducedVal.count += countObjVals[i].count;
    reducedVal.mark += countObjVals[i].mark;
```

```
    }
    return reducedVal;
}

var finalizeFunction = function (key, reducedVal) {
    reducedVal.avg = reducedVal.mark/reducedVal.count;
    return reducedVal;
    };


db.student.mapReduce( map,reduce,
                      {out: "out",
                      finalize: finalizeFunction});

db.out.aggregate({$match : {"value.avg" : {$gte : 4}}},
                 {$group : {_id : "$_id.group",
                            count : {$sum : 1}}},
                {$sort : {count : -1}},
                {$limit : 1})
```

4. Запрос для PostgreSQL базы данных:

```
SELECT group_id,count(student_id) FROM
  (SELECT student.group_id,student.student_id, AVG(mark)
    FROM
      student, studygroup, student_result
    WHERE
      student.student_id = student_result.student_id and
      student.group_id = studygroup.group_id
     GROUP BY student.group_id, student.student_id
    ) AS "p"
WHERE
  p.AVG > 4
GROUP BY group_id
```

```
ORDER BY count(student_id) DESC
LIMIT 1;
```

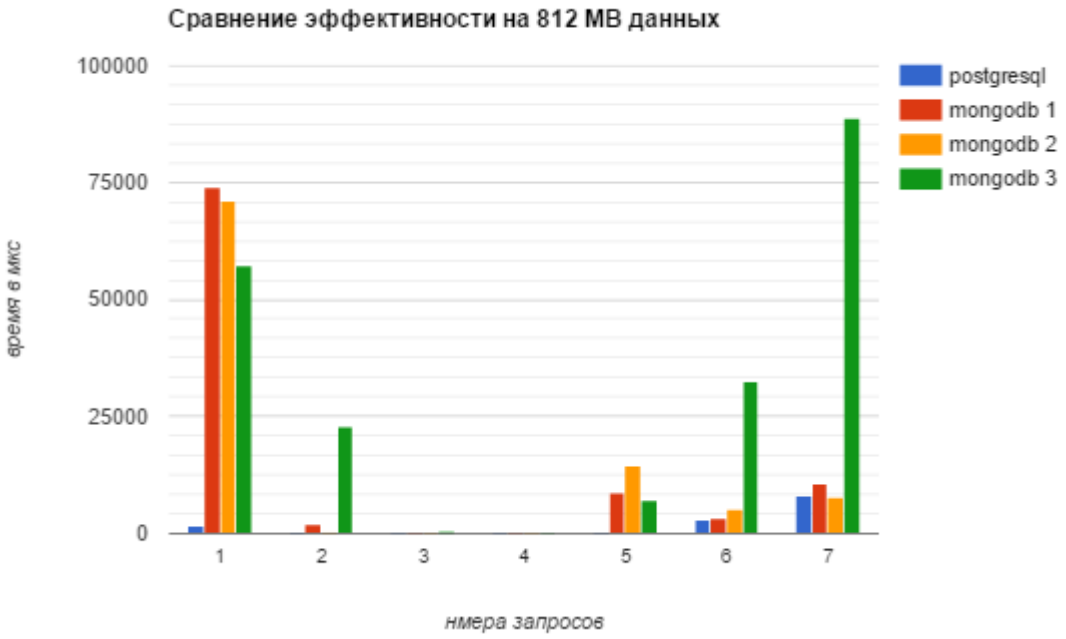
Для MongoDB рассматривалось несколько вариантов выполнения некоторых запросов: через aggregation framework или через MapReduce. Выбирался вариант, который давал лучшие результаты.

Так как для выполнения запросов в MongoDB мы предварительно никак не отсеивали инфомарцию, то есть в aggregation framework запросах не использовалсь функции $project, $limit, $skip, то и индексирование нам никак не пригодилось[4].

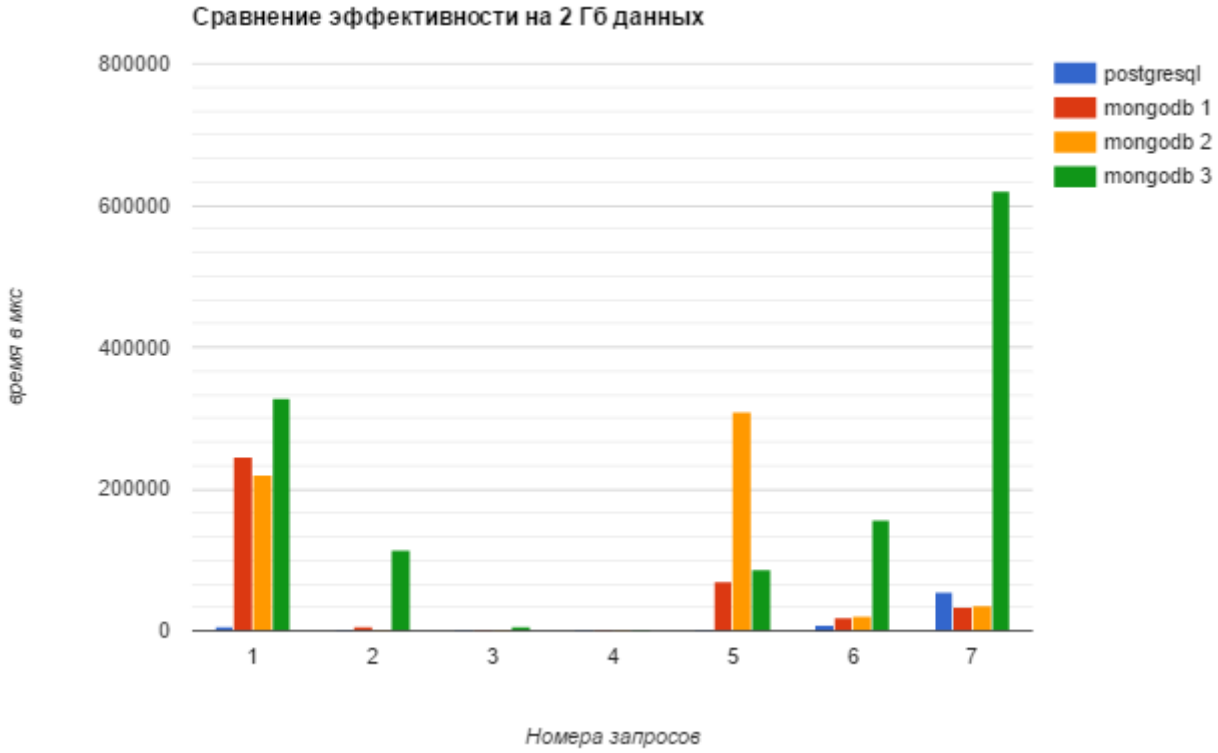
# 5. Результаты

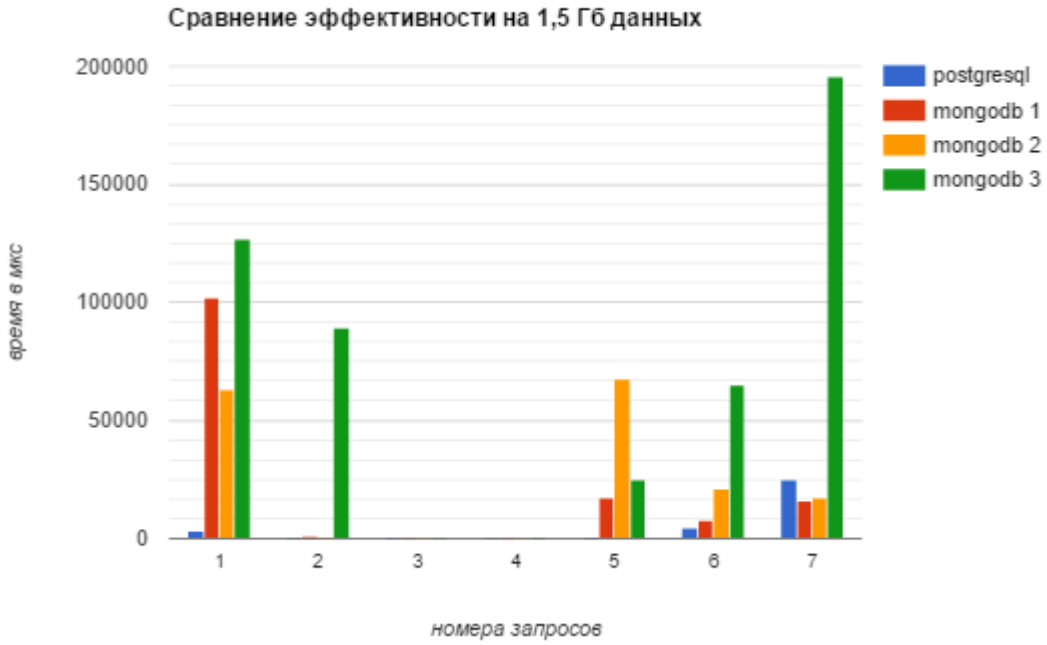
Как упоминалось выше, результаты измерений получены для баз данных размерами 812 Мб,1.5 Гб и 2 Гб. Всего было произведено по 10 измерений для каждого запроса. Измерения проводились на ”горячей” базе данных, так как для обеспечения измерений на ”холодной” базе приходилось бы после каждого замера перезапускать сервер.

На диаграммах postgresql - база данных PostgreSQL, mongodb 1 - база данных MongoDB с тремя коллекциями, mongodb 1 - база данных MongoDB с двумя коллекциями, mongodb 1 - база данных MongoDB с одной коллекцией (см. Приложение).

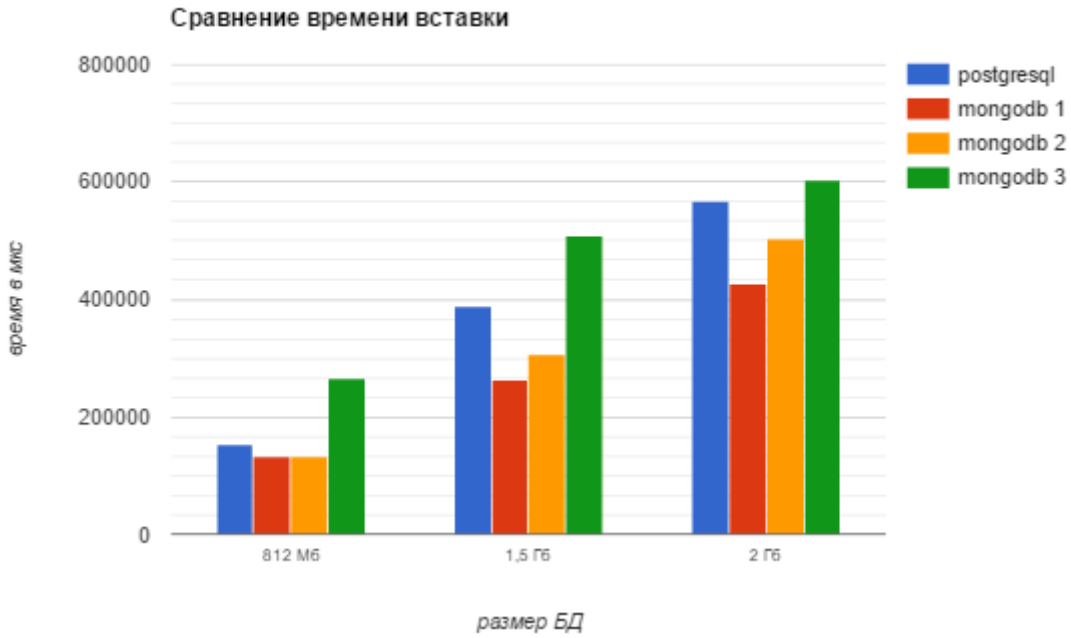



На получившихся диаграммах сравнения эффективности мы видим, что MongoDB уступает PostgreSQL практически на всех запросах, выигрывая у последнего только во времени вставки, и имея сравнимые результаты для простейших запросов (найти всех Долли, посчитать количество групп с заданным id). Также заметно, что время выполнения запросов растет практически линейно. Наиболее плохо себя показала коллекция 3 из MongoDB, что, возможно, случилось из-за больших объемов документов, которые включают в себя всю информацию о студенте (от его имени до количества часов преподаваемого предмета у него).




Из-за структуры запросов и того, что в mongoDB существуют ограничения на объем документов 16Mb, мы не смогли их обработать aggregation framework, а для всех запросов использовали технолгию MapReduce, которая, возможно, немного проигрывает aggregation в других случаях. Лучше всего в Mongo проявила себя база данных, в которой информа-




ция была распределена по трем коллекциям, что позволило избежать дублирования данных (кроме хранения массива id сопряженных объектов). В целом, возможно, MongoDb проигрывает PostgreSQL из-за особенностей построения запросов и отсутствия возможности в данных обстоятельствах использовать индексирование.

## 5.1. Обоснование использования результатов

Для каждого запроса получена выборка нормально-распределенных случайных величин, для которых посчитано среднеквадратичное отклонение по формуле

$$s = \sqrt{\frac{\sum_{i=1}^{n} (X - \bar{X})}{n}}, \tag{1}$$

где X - случайная величина, $\bar{X}$ - среднее арифметическое выборки, n - количество измерений, s - среднеквадратичное отклонение. Значения всех величины удовлетворяют правилу "трех сигм", т.е лежат в интервале

$$(\bar{x} - 3\sigma; \bar{x} + 3\sigma)\,, \tag{2}$$

где в качестве сигмы выступает s из формулы (1). Выполнение этого правила дает нам право использовать полученные результаты.

# Заключение

Была разработана и реализована схема базы данных в PostgreSQL и MongoDB, позволяющая реализовать сложные запросы и сравнить вышеупомянутые СУБД на уровне задач, в качестве которых выступили аналитические запросы, просматривающие всю базу данных. Были получены результаты измерений времени выполнения запросов на “горячей” базе данных, которые показывают, что для данного класса задач PostgreSQL является более эффективной, чем MongoDB, уступая последней только на этапе вставки данных.

# Список литературы


[1] Cristiano Politowski Vinicius Maran. Comparacao de Performance entre PostgreSQL e MongoDB. — URL: https://www.researchgate.net/publication/261871960_Comparacao_de_Performance_entre_PostgreSQL_e_MongoDB#pfa.

[2] Linster Marc. Postgres Outperforms MongoDB and Ushers in New Developer Reality // EnterpriseDB.

[3] Michael. Сравнение производительности MongoDB vs PostgreSQL // Хабрахабр. — 2013. — URL: https://habrahabr.ru/post/197590/.

[4] MongoDB. Aggregation Pipeline Optimization. — 2016. — URL: https://docs.mongodb.com/v3.0/core/aggregation-pipeline-optimization/.

[5] MongoDB. The MongoDB 3.0 Manual. — 2016. — URL: https://docs.mongodb.org/manual/?_ga=1.133199502.1653685077.1445621960.

[6] PostgreSQL. The PostgreSQL 9.4.5 Documentation. — 2016. — URL: http://www.postgresql.org/docs/9.4/interactive/index.htm.

[7] Свинарев Сергей. Еще раз о росте популярности NoSQL. — 2015. — URL: http://www.jetinfo.ru/stati/silnye-i-slabye-storony-nosql.

# Приложение

- Схемы коллекций MongoDB

  (как выглядят те же данные в виде коллекций и документов)

  а) База данных из 3х коллекций

```
Person{
        _id  :
        first_name
        last_name
        patronymic_name
        phone
        student:{
              budget_or_not:
              group_id:
              results:[{
                    subject_id  :
                    mark  :} ,... {}]
}
}


Group{
        _id:
        form_of_education:
        admission_year:
        graduation_year:
        department_board:
        group_number:
        title_of_education_program:
        subjects[id1 ,id2 ...]
        students[id1 ,id2 ,...]
}
```

```
Subjects{
    _id
    title_of_subject
    term_number
    form_of_exam
    hours_per_semestr
}
```

б) База данных из 2х коллекций

```
Student{
    _id:    ,
    first_name:    ,
    last_name:     ,
    patronymic_name:  ,
    phone:    ,
    budget_or_not:    ,
    groups_id:    ,
    student_results:  [
        {
            subjects_id:    ,
            mark:
        },
    ]}
```

```
Group{
    _id:  ,
    form_of_education:  ,
    admission_year:  ,
    graduation_year:  ,
```

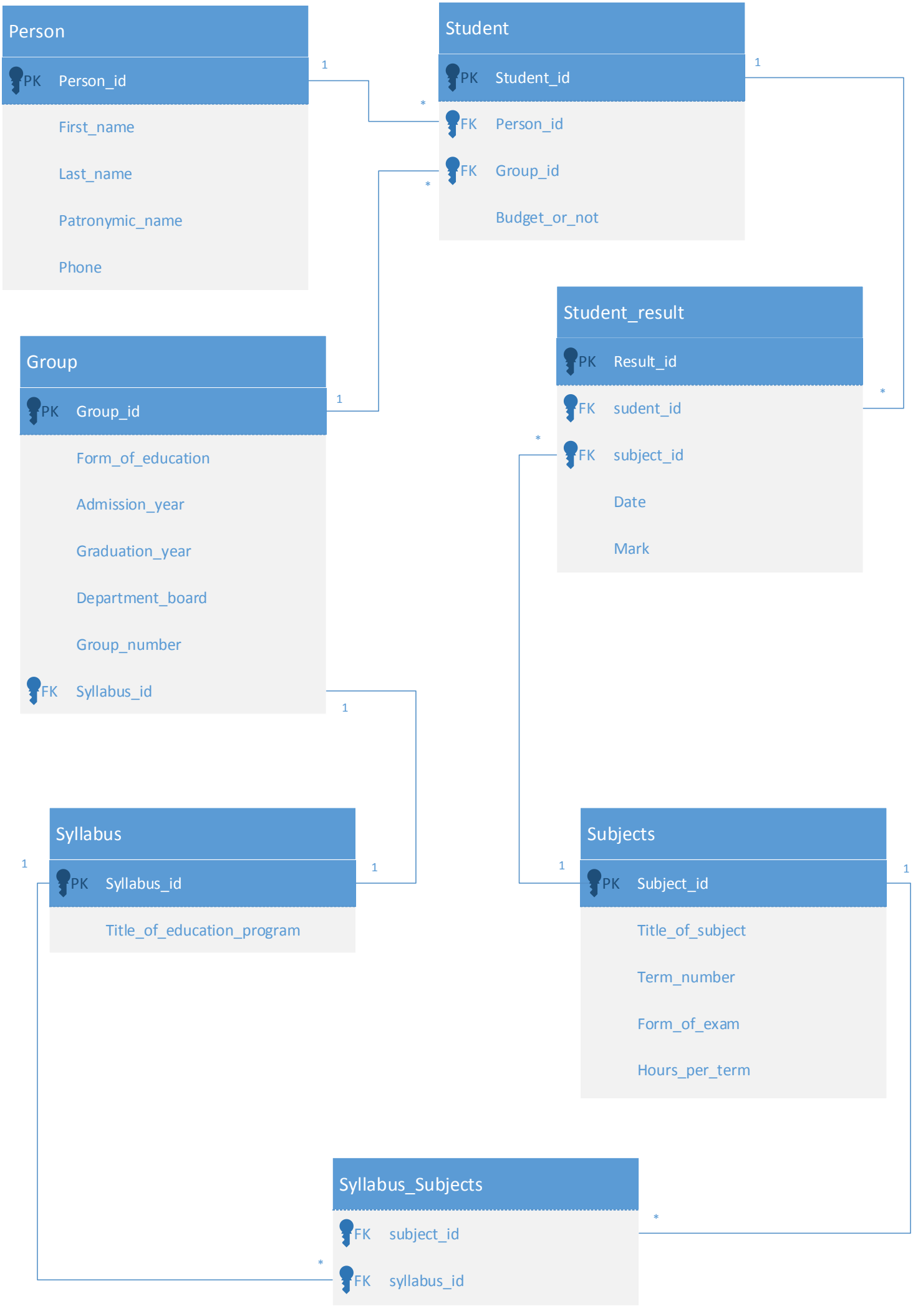
```
    department_board: ,
    group_number: ,
    syllabus: {
        title_of_education_program: ,
        subjects: [{
                subjects_id: ,
                title_of_subject: ,
                term_number: ,
                form_of_exam: ,
                hours_per_semestr: },{} ]
    }
}
```

в) База данных из 1й коллекции

```
Student{
    _id: ,
    first_name: ,
    last_name: ,
    patronymic_name: ,
    phone: ,
    group: {
        group_id: ,
        budget_or_not: ,
        form_of_education: ,
        admission_year: ,
        graduation_year: ,
        department_board: ,
        group_number: ,
        syllabus: {
            title_of_education_program: ,
            subjects: [
                {
                    subjects_id: ,
```

```
                    title_of_subject:  ,
                    term_number:  ,
                    form_of_exam:  ,
                    hours_per_semestr:  ,
                    mark:
                },{}
            ]
        }
    }
}
```

- Схема таблиц PostgreSQL


Person
PK Person_id
First_name
Last_name
Patronymic_name
Phone
Student
PK Student_id
FK Person_id
FK Group_id
Budget_or_not
Student_result
PK Result_id
FK sudent_id
FK subject_id
Date
Mark
Group
PK Group_id
Form_of_education
Admission_year
Graduation_year
Department_board
Group_number
FK Syllabus_id
Syllabus
PK Syllabus_id
Title_of_education_program
Subjects
PK Subject_id
Title_of_subject
Term_number
Form_of_exam
Hours_per_term
Syllabus_Subjects
FK subject_id
FK syllabus_id
1
*