УДК 004.652:622

ПРИМЕНЕНИЕ СУБД MONGODB В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ ГОРНОГО ПРОИЗВОДСТВА

Соколова А.С., ст. преп., Черненко Д.В., ст. преп., Куксевич В.Ф., ст. преп.

Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *В статье представлено краткое сравнение реляционных и документо-ориентированных моделей СУБД, рассмотрен вариант создания и заполнения базы данных в СУБД MongoDB в применении к процессам горного производства.*

Ключевые слова: программное обеспечение, автоматизация, горная промышленность, базы данных, нереляционные СУБД.

Программное обеспечение (ПО) производств горной промышленности за последние несколько лет серьезно усовершенствовалось, что закономерно привело к изменению характера деятельности горных компаний и к поступательному росту их производительности [1]. Тем не менее, большинство предприятий используют информационные технологии для улучшения отдельных технологических процессов, а не в целом производства. Много усилий тратится на автоматизацию старых методов и оборудования вместо кардинального их изменения.

Большинство современного ПО, применяемого в том числе в горном деле, использует базы данных (БД), построенные на основе реляционной модели, позволяющей представлять информацию о предметной области с помощью взаимосвязанных таблиц [2].

Как известно, любая реляционная БД имеет стандартную схему, отражающую количество таблиц и связи между ними. Совершенно новый подход к построению БД реализует документо-ориентированная система управления базами данных (СУБД) MongoDB, не требующая описания схемы связей таблиц, а значит и минимизирующая недостатки реляционной модели. Данная СУБД основана на коллекциях различных документов и относится к так называемым NoSQL СУБД. При этом количество полей, содержание и размер этих документов, может отличаться, то есть различные сущности не должны быть идентичными по структуре [3].

К основным преимуществам MongoDB относят: крайне понятная структура каждого объекта; легкость масштабирования; использование для хранения обрабатываемых в данный момент данных внутренней памяти, что позволяет получать более быстрый доступ; хранение данных в виде JSON-документов; поддержка динамических запросов документов (document-based query); отсутствие сложных JOIN-запросов и необходимости маппинга объектов приложения в объекты БД.

Таким образом, можно сделать вывод, что MongoDB является достаточно эффективным решением при использовании Big Data в автоматизации проектирования горных работ.

Так как MongoDB написана на C++, то она легко устанавливается на самые разные платформы: Windows, Linux, MacOS, Solaris.

В качестве примера рассмотрим создание и заполнение базы данных в нереляционной СУБД MongoDB в применении к процессам горного производства.

Создадим БД «Скважины» и добавим в неё следующие сущности: Скважины (Wells), Описание (Description), Диаметры (Diameters), Пробы (Samples).

```
# use Well_DB
# db.createCollection("Description")
# db.createCollection("Diameters")
# db.createCollection("Samples")
# db.createCollection("Wells")
```

Добавим документы в коллекции. Мы не указываем _id, из-за чего они будут сгенерированы автоматически.

Коллекция Description:

```
# db. Description.insertMany([
{ Power: 1.40, Grade: "АФМ"},
{ Power:1.20, Grade: "АФМ 15-3"},
{ Power: 1.00, Grade: "ФМ"}
])
```

Коллекция Diameters:

```
# db. Diameters.insertMany([
{ Deep:3.00, Diameter: 3.35},
{ Deep:4.40, Diameter: 4.30},
{ Deep:5.60, Diameter: 5.25}
])
```

Коллекция Samples:

```
# db. Samples.insertMany([
{ Number: "1923a", Fe_gen: 30.7, P2O5: 9.0,
  Description: [ObjectId("61949f8814e7b69fb36e4f3e")]},
{ Number: "1924a", Fe_gen: 22.2, P2O5: 9.0,
  Description: [ObjectId("61949f8814e7b69fb36e4f3f")]},
{ Number: "1925a", Fe_gen: 22.3, P2O5: 9.1,
  Description: [ObjectId("61949f8814e7b69fb36e4f3d")]}
])
```

Коллекция Wells:

```
# db. Wells.insertMany([
{Name: "100", Date: "2022-04-21T18:25:43-05:00", Horizon: 262,
  Diameters_id: [ObjectId("61949fa714e7b69fb36e4f40")],
Samples _id: [ObjectId("6194a0c314e7b69fb36e4f46")]},
{Name: "100c", Date: "2021-04-21T18:25:43-05:00", Horizon: 250,
  Diameters_id: [ObjectId("61949fa714e7b69fb36e4f40")],
  Samples_id: [ObjectId("6194a0c314e7b69fb36e4f47")]},
{Name: "101", Date: "2022-05-21T18:25:43-05:00", Horizon: ,
  Diameters_id: [ObjectId("61949fa714e7b69fb36e4f41")],
Samples_id: [ObjectId("6194a0c314e7b69fb36e4f48")]},
{Name: "101c", Date: "2021-05-21T18:25:43-05:00", Horizon: 250,
  Diameters_id: [ObjectId("61949fa714e7b69fb36e4f41")],
Samples_id: [ObjectId("6194a0c314e7b69fb36e4f47")]},
{Name: "102", Date: "2021-04-12T18:25:43-05:00", Horizon: 262,
  Diameters_id: [ObjectId("61949fa714e7b69fb36e4f42")],
Samples_id: [ObjectId("6194a0c314e7b69fb36e4f48")]}
])
```

Для тестирования созданной БД выполним ряд запросов выборки данных.

А) Выберем пробы, номер которых содержит 24:

```
# db. Samples.find({Nomer: /24/i})
```

Б) Выведем таблицу проб вместе с документом «Описание», id которого указан в документе проб при помощи оператора $lookup:

```
#db.Samples.aggregate([{
    $lookup:  {
from: "Description",
    localField: " Description_id",
    foreignField: "_id",
    as: "Description"
    }
}]),
```

В) Аналогично запросу Б, выведем таблицу скважин.

```
#db.Wells.aggregate([{
    $lookup:  {
                from: "Diameters",
                localField: "Diameters_id",
                foreignField: "_id",
                as: "Diameters"
            }
},
{ $lookup: {
                from: "Samples",
                localField: "Samples_id",
                foreignField: "_id",
```

```
                    as: "Samples"
                }
    }])
```

Г) К предыдущему запросу В добавим фильтрацию. Выведем скважины с горизонтом меньше 260:

```
    # db.Wells.aggregate([
    {$match:
    { Horizon: {$lt: 260}}
    },
    { $lookup:  {
                    from: "Diameters",
                    localField: "Diameters_id",
                    foreignField: "_id",
                    as: "Diameters"
                }
    },
    { $lookup: {
                    from: "Samples",
                    localField: "Samples_id",
                    foreignField: "_id",
                    as: "Samples"
                }
    }])
```

Таким образом, представленный вариант БД может быть эффективно использован в составе ПО предприятий горного производства.

Список использованных источников

1. 302r-Краткий обзор ПО [Электронный ресурс]. – Режим доступа: http://masters.donntu.org/2003/fvti/petrovskaya/lib/review.htm. – Дата доступа: 13.05.2022.
2. Реляционные базы данных: достоинства и недостатки. – GOSy VMKSS [Электронный ресурс]. – Режим доступа: https://sites.google.com/site/gosyvmkss12/bazy-dannyh/07-relacionnye-bazy-dannyh-dostoinstva-i-nedostatki. – Дата доступа: 13.05.2022.
3. Руководство по MongoDB. Преимущества – PROSELYTE [Электронный ресурс]. – Режим доступа: https://proselyte.net/tutorials/mongodb/advantages/. – Дата доступа: 13.05.2022.

УДК 004.4'236

МОДЕЛИРОВАНИЕ ЭЛЕКТРОСТАТИЧЕСКИХ ПОЛЕЙ ВОКРУГ ЧЕЛОВЕКА С ПРИМЕНЕНИЕМ КРОССПЛАТФОРМЕННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Белицкая О.А., к.т.н., доц.

Российский государственный университет им. А.Н. Косыгина, г. Москва, Российская Федерация

Реферат. *Статья посвящена практическому применению кроссплатформенного программного обеспечения для анализа методом конечных элементов COMSOL Multiphysics, которое является популярной у специалистов в сфере автоматизированных инженерных расчётов. Пользовательский интерфейс и инструментарий позволяет моделировать распространение электростатических разрядов в воздухе вокруг заряженного человека по всей геометрии тела.*

Ключевые слова: метод конечных элементов, COMSOL Multiphysics, электростатические разряды, электростатическое поле, моделирование.

С развитие современной вычислительной техники процесс проектирования и исследования различных конструкций был заметно облегчен с внедрением программ на