MongoDB — документоориентированная система управления базами данных, не требующая описания схемы таблиц.
Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных.
Написана на языке C++.
Лицензия: Server Side Public License (SSPL), ранее GNU AGPL (СУБД) и Apache License (драйверы)

СУБД MongoDB

1. Быстрая база данных
2. Устоявшийся проект
3. Open-sourse, но разрабатывается и поддерживается компанией MongoDB, Inc (уст. 10gen)
4. Одно из наиболее универсальных решений

Документо-ориентированная СУБД

-имеют более сложную структуру, чем хранилища «ключ-значение».
-Имеют иерархическую структуру.
-Как правило хранят данные в одном из известных форматов: XML, JSON, BSON, YAML.

Примеры: CouchDB, Couchbase, MarkLogic, MongoDB

Варианты использования MongoDB

- хранение и регистрация событий;
- системы управления документами и контентом;
- электронная коммерция;
- игры;
- данные мониторинга, датчиков;
- мобильные приложения;
- хранилище операционных данных веб-страниц (например, хранение комментариев, рейтингов, профилей пользователей, сеансы пользователей).

Поддержка MongoDB языками

Официальные драйверы:

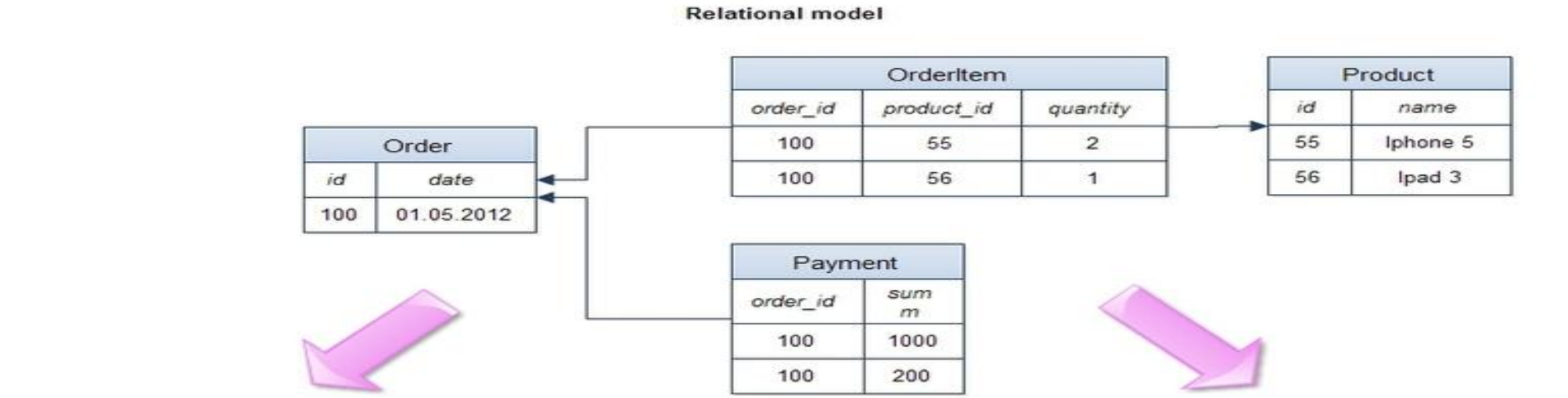
- PERL
- PHP
- PYTHON
- RUBY
- C/C++
- JAVA
- .NET
- Javascript
- Erlang
- Scala
- Haskell

Драйверы open-sourse:

- ActionScript
- Clojure
- Delphi
- Node.js
- F#
- Go
- Groovy
- Lua
- и т.д.

Кто использует MongoDB ?

Представление данных в виде агрегатов (aggregates)



Aggregate model 1

```
// Order document
{
"id": 100,
"customer_id": 1000,
"date": 01.05.2012,
"order_items": [
        {
        "product_id": 55,
        "product_name": Iphone5,
        "quantity": 2
        },
        {
        "product_id": 56,
        "product_name": Ipad3
        "quantity": 1
        }
],
"payments":[
        {
        "sum": 1000,
        "date": 03.05.2012
        }
]
}
// Product document here
{...}
```

Aggregate model 2

```
// Order document
{
"id": 100,
"customer_id": 1000,
"date": 01.05.2012,
"order_items": [
        {
        "product_id": 55,
        "product_name": Iphone5,
        "quantity": 2
        },
        {
        "product_id": 56,
        "product_name": Ipad3
        "quantity": 1
        }
]
}
// Payment document
{
"order_id": 100,
"sum": 1000,
"date": 03.05.2012
}
// Product document here
{...}
```

Организация данных

SQL	MongoDB
База данных	База данных
Таблица	Коллекция
Строка/запись	Документ

Изменение мышления

Навыки работы с SQL не подходят для NoSQL

1. Нет JOIN
2. Контроль данных из приложения, а не из БД
3. Избыточность данных вместо нормализации

Различия схемы БД в RDBMS и MongoDB

Схема БД в RDBMS

В MongoDB, имеем одну коллекцию и структуру(без схемы) документа:

```
{
 _id: POST_ID
  title: TITLE_OF_POST,
 description: POST_DESCRIPTION,
 by: POST_BY,
 url: URL_OF_POST,
 tags: [TAG1, TAG2, TAG3],
 likes: TOTAL_LIKES,
 comments: [
            { user:'COMMENT_BY', message: TEXT, dateCreated: DATE_TIME, like: LIKES },
            { user:'COMMENT_BY', message: TEXT, dateCreated: DATE_TIME, like: LIKES }
            ]
}
```

Различия схемы БД в MongoDB

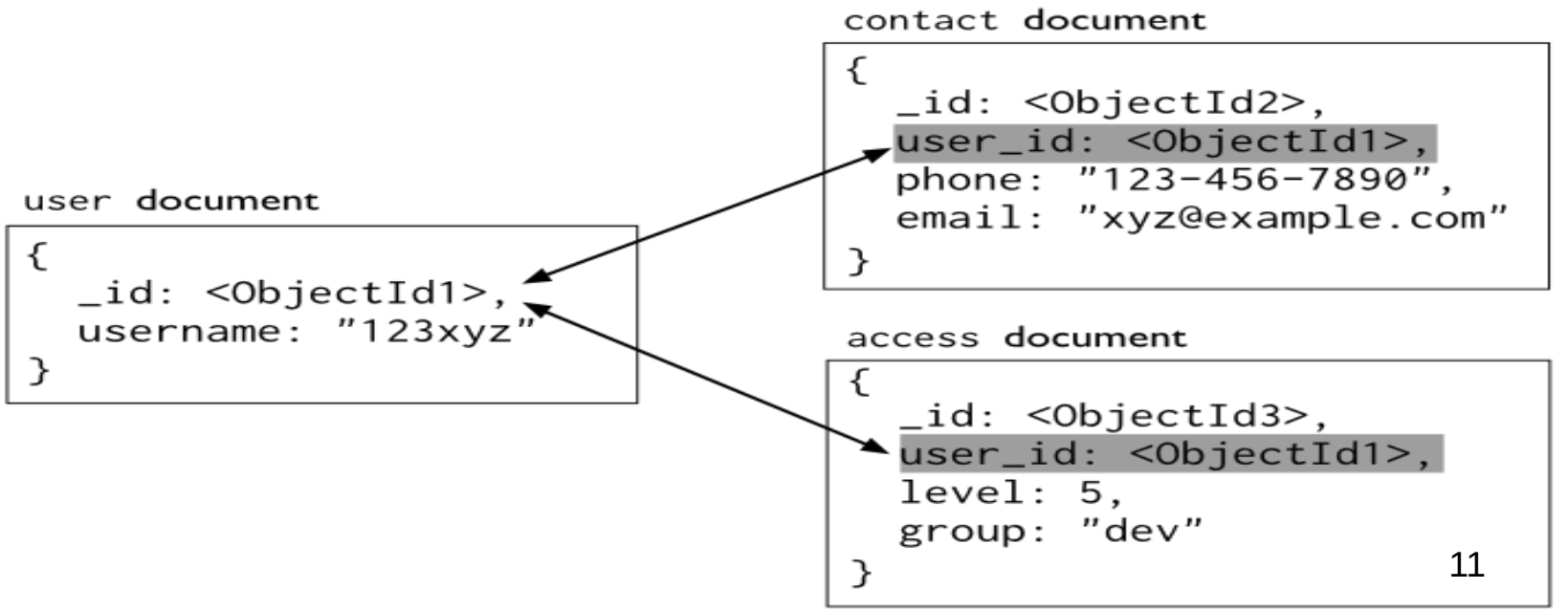
Структура документа с вложенными документами (денормализованная модель)

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
             level: 5,
             group: "dev"
           }
}
```

Embedded sub-document

Embedded sub-document

Пример нормализованной модели с использованием ссылок между документами



Сравнение с другими СУБД: чем лучше?

Преимущества и недостатки

Нормализация данных	Данные в виде агрегатов
• Целостность информации при обновлении (меняем запись в одной таблице, а не в нескольких) • Ориентированность на широкий спектр запросов к данных	• Оптимизация только под определенный вид запросов • Сложности при обновлении денормализованных данных
• Неэффективна в распределенной среде • Низкая скорость чтения при использовании объединений (joins) • Несоответствие объектной модели приложения физической структуре данных (impedance mismatch, решается с помощью Hibernate etc.)	• Лучший способ добиться большой скорости на чтение в распределенной среде • Возможность хранить физически объекты в том виде, в каком с ними работает приложение (легче кодировать и меньше ошибок при преобразовании) • Родная (native) поддержка атомарности на уровне записей

Организация данных

Типы данных BSON

- String
- Integer
- Double
- Date
- Byte array (бинарные данные)
- Boolean
- Null
- BSON Object

Ключ

Каждому добавленному документу автоматически предоставляется уникальный ключ

_id: ObjectId(“47cc67093475061e3d95369d”)

C - CREATE
R - READ
U - Update
D - DELETE

Интерфейс

- Основным интерфейсом к базе данных была командная оболочка **mongo**.
- С версии MongoDB 3.2 в качестве графической оболочки поставляется «MongoDB **Compass**», имеется панель командной оболочки **mongosh**
- Существуют продукты и сторонние проекты, которые предлагают инструменты с графическим интерфейсом для администрирования и просмотра данных.
- записи в MongoDB с Robomongo 0.8.5.

CREATE

SQL

```
CREATE DATABASE vldc;
CREATE TABLE vldc.users (`id` INT AUTO_INCREMENT PRIMARY
KEY, `first_name` VARCHAR(50), `last_name` VARCHAR(50));
INSERT INTO vldc.users SET first_name = “Oleg”;
```

MongoDB

```
use vldc
db.users.insert({ first_name: “Oleg” })
```

READ

SQL

```
SELECT * FROM users
SELECT first_name FROM users
```

MongoDB

```
db.users.find()
db.users.find({}, { first_name: 1 })
```

READ

SQL

```
SELECT * FROM users
WHERE first_name = “Oleg” ORDER BY id DESC LIMIT 1,10
```

MongoDB

```
db.users.find({first_name: “Oleg”}).sort({ _id: -1 }).skip(1).limit(10)
```

Операторы условий

- $gt, $lt, $gte, $lte
- $ne
- $in, $nin
- $mod
- $all
- $size
- $exists
- $type
- $not
- $where

Update

SQL

```
UPDATE users SET last_name = "Kachan" WHERE first_name= "Oleg"
```

MongoDB

```
db.users.update({ first_name: "Oleg" }, { last_name: "Kachan" })
db.users.update({ first_name: "Oleg" }, { $set: { last_name: "Kachan" } })
```

Операторы модификации

- $set
- $unset
- $inc
- $push
- $pushAll
- $addToSet
- $pop
- $pull
- $pullAll

DELETE

SQL

```
DELETE FROM users WHERE id = 1
DELETE FROM users WHERE first_name = “Oleg”
```

MongoDB

```
db.users.remove({ _id: ObjectId(“4df8fb81ed4cadd6271c0000”) })
db.users.remove({ first_name: “Oleg”})
```

Создание индексов

SQL

```
ALTER TABLE `users` ADD INDEX (`first_name`)
```

MongoDB

```
db.users.ensureIndex({ first_name: 1 }) // по возрастанию
db.users.ensureIndex({ first_name: -1 }) // по убыванию
```

Гео-индекс

MongoDB

```
db.places.ensureIndex({location: "2d"})
```

Поиск при помощи операторов

- $near – поиск объектов с сортировкой, самые близкие - первые
- $box – поиск объектов в заданном квадрате
- $center – поиск объектов в заданном радиусе

Литература

1. Фаулер М., Салерадж П.Д. NoSQL: новая методология разработки нереляционных бах данных. : Пер. с англ. М.: ООО «И.Д.Вильямс», 2015. 192 с.: ил.
2. Редмонд Э., Уилсон Д.Р. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. Под редакцией Жаклин Картер / Пер. с англ. Слинкин А.А. М.: ДМК Пресс, 2013. 384 с.: ил.