Тема 17. Введение в NoSQL и MongoDB.

Учебные вопросы:

1. Проблема масштабируемости и ограничения SQL.
2. Классификация NoSQL систем
3. Знакомство с MongoDB
4. Структура документа и динамические схемы

1. Проблема масштабируемости и ограничения SQL.

Реляционные базы данных (RDBMS) доминировали десятилетиями благодаря своей надежности и строгости.

Однако с приходом эпохи Web 2.0 (соцсети, онлайн-ритейл, IoT) архитекторы столкнулись с рядом проблем.

А. Масштабируемость.

Когда база данных перестает справляться с нагрузкой, есть два пути решения:

Вертикальное масштабирование (Scale-up):

- Суть: Покупка более мощного сервера (больше RAM, мощнее CPU, быстрее SSD).
- Минус: У этого пути есть физический предел (самый мощный сервер в мире все равно имеет границы) и экспоненциальная стоимость (сервер в 2 раза мощнее может стоить в 10 раз дороже).
- SQL-базы исторически создавались именно под этот тип.

Горизонтальное масштабирование (Scale-out):

- Суть: Добавление обычных, недорогих серверов в кластер. Нагрузка распределяется между ними.
- Проблема SQL: Реляционные базы данных очень сложно распределить по разным машинам из-за необходимости поддерживать связи (JOIN) и транзакции между серверами. Это требует огромных накладных расходов на синхронизацию.

Б. Жесткая схема (Rigid Schema)

В SQL вы должны заранее описать структуру таблицы (ALTER TABLE).

- Если у вас миллиард строк, любое изменение структуры (например, добавление новой колонки) может привести к блокировке всей базы на часы или даже дни.
- В современном мире данные меняются быстро: сегодня у товара 5 характеристик, завтра 50. SQL плохо адаптируется к такой изменчивости.

B. "Проклятие" связей (JOINs)

Реляционная модель требует нормализации — разбиения данных на множество таблиц для исключения дублирования.

- Чтобы собрать профиль пользователя со всеми его заказами, адресами и скидками, SQL должен выполнить операцию JOIN.
- Когда данных миллионы, выполнение множественных JOIN становится крайне медленным процессом, который «съедает» всю оперативную память сервера.

Г. Понятие Big Data и высокая доступность.

Современные приложения (как Telegram, Amazon или Uber) требуют:

- Низкой задержки (Low Latency): ответ за миллисекунды.
- Отказоустойчивости: если один сервер сгорел, пользователь не должен этого заметить.

Традиционный SQL-сервер обычно является "единой точкой отказа" (Single Point of Failure). NoSQL-системы изначально проектировались как распределенные системы, где данные дублируются на разных узлах.

Итог:

SQL идеально подходит для сложных финансовых операций, где важна абсолютная точность и связи.

Но он проигрывает NoSQL там, где нужно хранить огромные объемы данных, быстро менять их структуру и дешево масштабироваться на сотни серверов.

Что такое NoSQL?

NoSQL (Not Only SQL — «не только SQL») — это общее название для баз данных, которые не используют традиционную реляционную модель таблиц со строгими связями.

2. Классификация NoSQL систем

NoSQL — это не одна база данных, а целое семейство инструментов, каждый из которых решает свою специфическую задачу.

Классификация NoSQL систем

Базы данных NoSQL делятся на четыре основные группы в зависимости от того, как они хранят и структурируют данные.

1. Хранилища «Ключ-значение» (Key-Value Stores)

Самый простой и быстрый тип NoSQL. Данные хранятся как в обычном словаре (Dictionary в C#): есть уникальный ключ и связанное с ним значение (строка, число или целый объект).

- Особенности: Невероятная скорость работы. База «не знает», что лежит внутри значения, она просто отдает его по ключу.
- Когда использовать: Кэширование данных, хранение сессий пользователей, корзины в интернет-магазинах.
- Примеры: Redis, Amazon DynamoDB.

2. Документоориентированные БД (Document Databases)

Данные хранятся в виде «документов» (обычно это JSON или его бинарный аналог BSON). В отличие от Key-Value, база данных «понимает» структуру документа и может искать по его внутренним полям.

- Особенности: Максимальная гибкость. Каждый документ в одной коллекции может иметь свои уникальные поля. Данные хранятся в иерархическом (вложенном) виде.
- Когда использовать: Контент-менеджмент (CMS), блоги, профили пользователей, интернет-магазины со сложными характеристиками товаров.
- Примеры: MongoDB, CouchDB.

3. Столбцовые БД (Wide-Column Stores)

Вместо хранения данных строками (как в SQL), эти базы группируют данные по столбцам.

- Особенности: Позволяют очень быстро считывать огромные объемы данных по конкретным признакам (например, только цены всех товаров за год). Легко масштабируются на тысячи серверов.
- Когда использовать: Аналитика «Больших данных» (Big Data), хранение истории действий пользователей (логи), финансовые отчеты.
- Примеры: Apache Cassandra, HBase.

4. Графовые БД (Graph Databases)

Основаны на теории графов. Состоят из «узлов» (объектов) и «ребер» (связей между ними).

- Особенности: В SQL поиск связей между друзьями друзей требует тяжелых операций. В графовой БД связь — это физический объект, поэтому поиск по сложным цепочкам происходит мгновенно.
- Когда использовать: Социальные сети (поиск друзей, рекомендации), системы обнаружения мошенничества (связи между счетами), логистика.
- Примеры: Neo4j, JanusGraph.

3. Знакомство с MongoDB

MongoDB — это кроссплатформенная, документоориентированная база данных с открытым исходным кодом. Она хранит данные не в таблицах, а в структурах, очень похожих на JSON.

Почему MongoDB так популярна?

- Высокая производительность: Написана на C++, поддерживает быстрые индексы.
- Гибкость: Вы можете добавить новое поле в объект в любой момент без пересоздания базы.
- Удобство для разработчика: Объекты в коде (POCO-классы) почти один-в-один соответствуют документам в базе.

Основные термины MongoDB

1. Документ (Document)

Это основная единица записи в MongoDB. Документ — это набор пар «ключ-значение».

- Аналог в SQL: Строка (Row).
- Формат: Хранится в BSON, но выглядит как JSON.
- Пример: {"name": "Ivan", "age": 25}.

2. Коллекция (Collection)

Группа документов, хранящихся в одной базе данных.

Аналог в SQL: Таблица (Table).

Особенность: Коллекции не требуют строгой схемы. В одной коллекции "Users" один документ может иметь поле "отчество", а другой — нет.

3. Поле (Field)

Единица данных в документе.

Аналог в SQL: Столбец/Колонка (Column).

Типы данных: Строки, числа, даты, массивы и даже другие документы (вложенные объекты).

4. Database (База данных)

Контейнер верхнего уровня, который объединяет несколько коллекций. Обычно одно приложение использует одну базу данных.

5. _id (ObjectId)

Каждый документ обязан иметь уникальное поле _id, выполняющее роль первичного ключа.

Если вы не укажете его при вставке, MongoDB создаст его автоматически.

Тип данных ObjectId содержит в себе временную метку, идентификатор машины и счетчик, что делает его уникальным во всей сети.

6. Вложенные документы (Embedded Documents)

Это возможность хранить один объект внутри другого.

Зачем: Чтобы не делать JOIN. Например, адрес клиента (улица, дом, город) можно хранить прямо внутри документа «Клиент», а не в отдельной таблице «Адреса».

7. Индекс (Index)

Специальная структура данных, которая позволяет базе быстро находить документы по значению определенного поля (например, по Email), не просматривая всю коллекцию целиком.

Формат данных: BSON

Хотя мы говорим, что MongoDB работает с JSON, внутри она использует формат BSON (Binary JSON).

- BSON — это бинарное представление JSON.
- Зачем он нужен? Он работает быстрее, занимает меньше места и поддерживает больше типов данных (например, Date, Decimal128 и Binary data), которых нет в обычном JSON.

Ключевое поле _id

В каждом документе MongoDB обязательно есть поле _id.

- Это первичный ключ (Primary Key).
- По умолчанию MongoDB использует тип ObjectId — это 12-байтовое уникальное значение, которое генерируется автоматически.
- Оно гарантирует, что даже в распределенной системе (на разных серверах) идентификаторы не совпадут.

Инфраструктура (Где запускать?)

- MongoDB Community Server: Бесплатная версия для установки на свой ПК/Сервер.
- MongoDB Atlas: "Облако", где база данных работает на серверах MongoDB.
- MongoDB Compass: Официальная программа с графическим интерфейсом для просмотра данных.

4. Структура документа и динамические схемы

Динамическая схема (Dynamic Schema)

В реляционных БД схема фиксирована. Если вы решили добавить поле Discount в таблицу Orders, вы должны изменить структуру всей таблицы.

В MongoDB действует принцип schemaless:

Документы в одной коллекции могут иметь разные наборы полей.

Поля могут добавляться «на лету» самим приложением.

Это идеально подходит для итеративной разработки (Agile), когда требования к данным меняются каждую неделю.

Два подхода к моделированию данных

В SQL у нас один путь — нормализация (разнос по разным таблицам). В MongoDB у разработчика есть выбор: Вложение или Ссылки.

1. Вложение (Embedding) — «Всё своё ношу с собой»

Связанные данные хранятся прямо внутри родительского документа.

- Пример: В документе User лежит массив объектов Addresses.
- Плюсы: Невероятная скорость чтения. Одним запросом вы получаете пользователя и все его адреса. Нет никаких JOIN.
- Минусы: Если адресов станут тысячи, документ может превысить лимит (16 МБ). Тяжело обновлять данные, если они дублируются (например, если один и тот же адрес у 100 пользователей).

2. Ссылки (Referencing) — «Как в SQL»

В документе хранится только _id другого документа.

- Пример: В документе Order хранится user_id.
- Плюсы: Данные не дублируются. Удобно для связей «многие-ко-многим».
- Минусы: Чтобы собрать полную информацию, приложению нужно сделать два запроса к базе (или использовать оператор $lookup, который работает медленнее, чем простое чтение).

Главное правило моделирования в MongoDB

Данные, которые используются вместе, должны храниться вместе.

Если вы на сайте всегда показываете комментарии вместе со статьей — вкладывайте комментарии в документ статьи. Если комментарии живут своей жизнью и их миллионы — делайте ссылки.

5. Основы работы (CRUD операции)

CRUD — это акроним: Create (Создание), Read (Чтение), Update (Обновление) и Delete (Удаление).

1. Create (Создание документов)

Для добавления данных используются методы **insertOne** (для одного объекта) или **insertMany** (для массива объектов).

Команда:

```
db.users.insertOne({
  name: "Алексей",
  age: 20,
  skills: ["C#", "SQL"]
})
```

Особенность: Если коллекции users не существовало, MongoDB создаст её автоматически в момент вставки первого документа.

Из чего состоит команда:

- db: Обращение к текущей базе данных (той, которую вы выбрали командой use myDatabase).
- users: Имя коллекции. Если такой коллекции еще нет, MongoDB создаст её автоматически.
- insertOne: Метод (функция), который говорит базе: «Возьми этот один объект и сохрани его».

2. Read (Чтение/Поиск документов)

Метод **find** возвращает документы, соответствующие фильтру.

Найти всех: **db.users.find()**

Поиск по условию: **db.users.find({ name: "Алексей" })**

NoSQL использует специальные ключи для сравнения:

$gt (greater than) — больше.

$lt (less than) — меньше.

$in — входит в список.

Пример: Найти пользователей старше 18 лет:
db.users.find({ age: { $gt: 18 } })

3. Update (Обновление документов)

Важно помнить: метод **updateOne** или **updateMany** требует использования оператора $set, иначе документ будет полностью перезаписан (заменен) на новый, и старые поля пропадут.

Правильное обновление:

```
db.users.updateOne(
  { name: "Алексей" },          // 1. Фильтр (кого обновляем)
  { $set: { age: 21 } }         // 2. Что меняем
)
```

4. Delete (Удаление документов)

Удаление работает аналогично поиску — вы передаете фильтр документов, которые нужно стереть.

- Удалить одного: db.users.deleteOne({ name: "Алексей" })
- Удалить всех, кто младше 18: db.users.deleteMany({ age: { $lt: 18 } })

Ограничения

При проектировании нужно помнить о физических лимитах MongoDB:

- Максимальный размер документа: 16 Мегабайт. (Для 99% задач этого более чем достаточно, но хранить внутри одного документа всю историю действий пользователя за 10 лет — плохая идея).
- Глубина вложенности: До 100 уровней (на практике редко используют больше 3-4).

6. Установка MongoDB и создание первой коллекции.

Шаги установки (Локальный вариант):

Скачивание: Скачайте установщик с официальнго сайта MongoDB:

https://www.mongodb.com/products/self-managed/community-edition

Установка: При установке выберите тип Complete.

- Важно: Убедитесь, что стоит галочка "Install MongoDB as a Service". Это позволит базе запускаться автоматически при включении ПК.
- MongoDB Compass: В конце установщик предложит установить Compass — это официальный графический интерфейс (GUI). Обязательно соглашайтесь.

Проверка: После установки база будет доступна по адресу localhost:27017.

www.mongodb.com MongoDB Community Edition | MongoDB

Спросить Алису AI

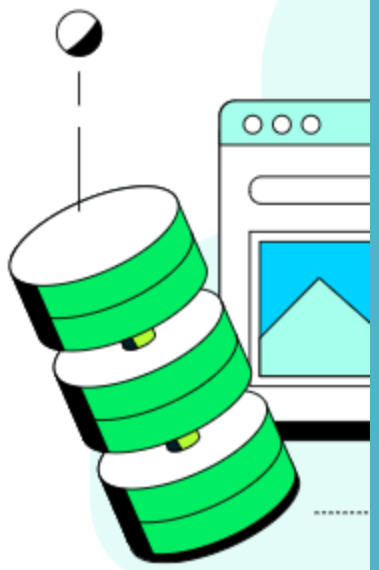
Community Edition

Overview Features Getting Started Resources

COMMUNITY EDITION

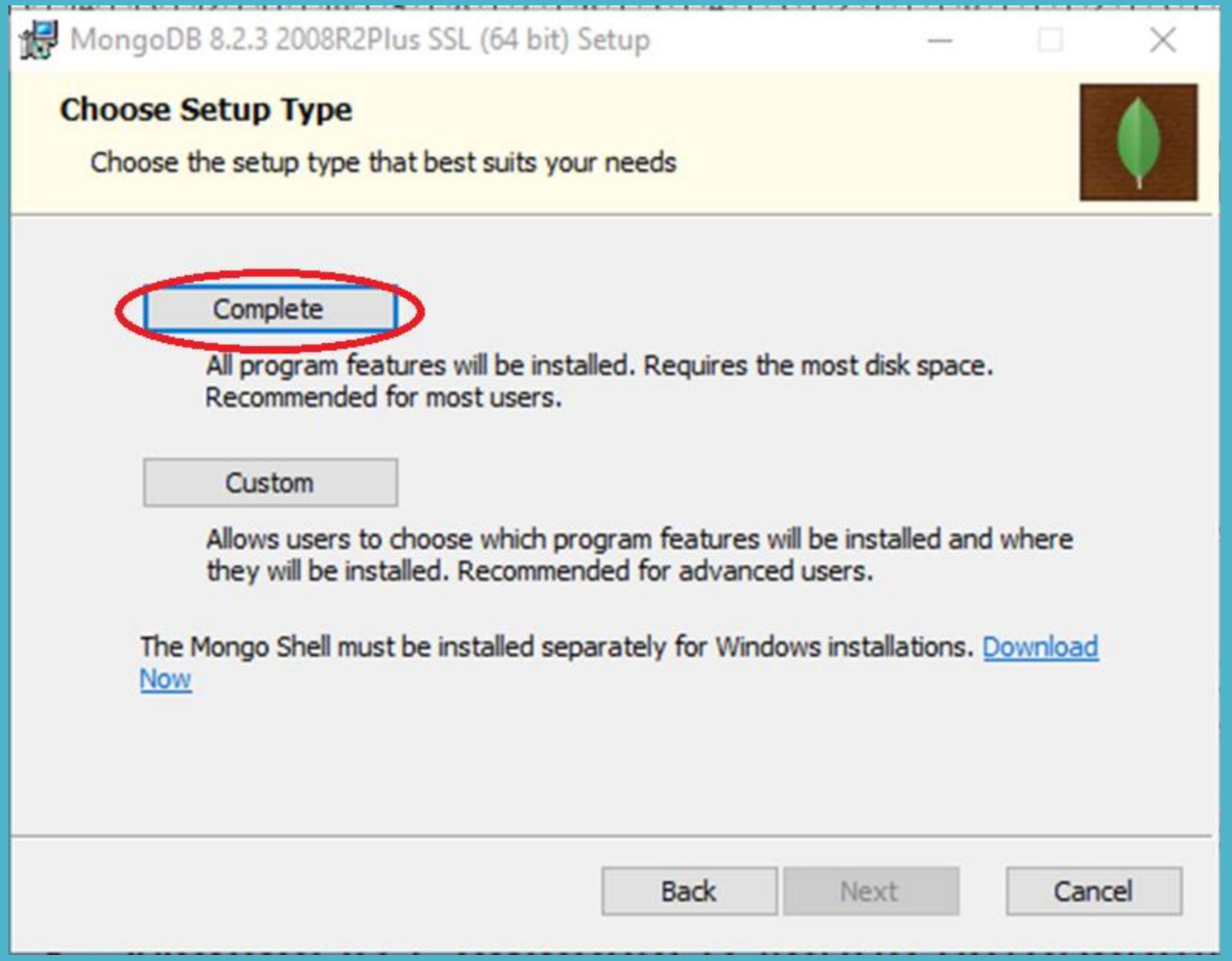
MongoDB Community Edition

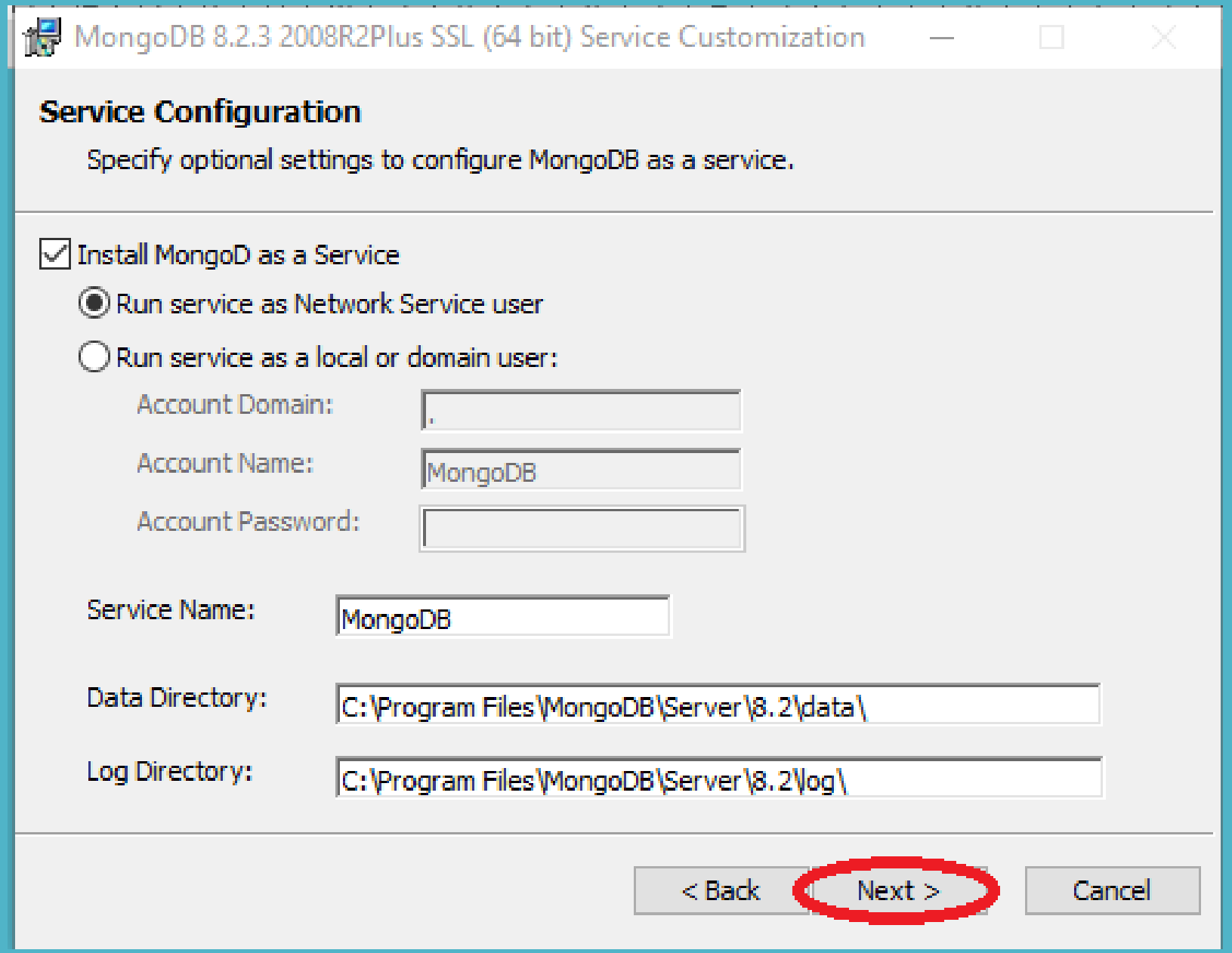
The source-available, free-to-use version of MongoDB's document database. Ideal for learning, experimenting, and building applications in a self-managed environment.

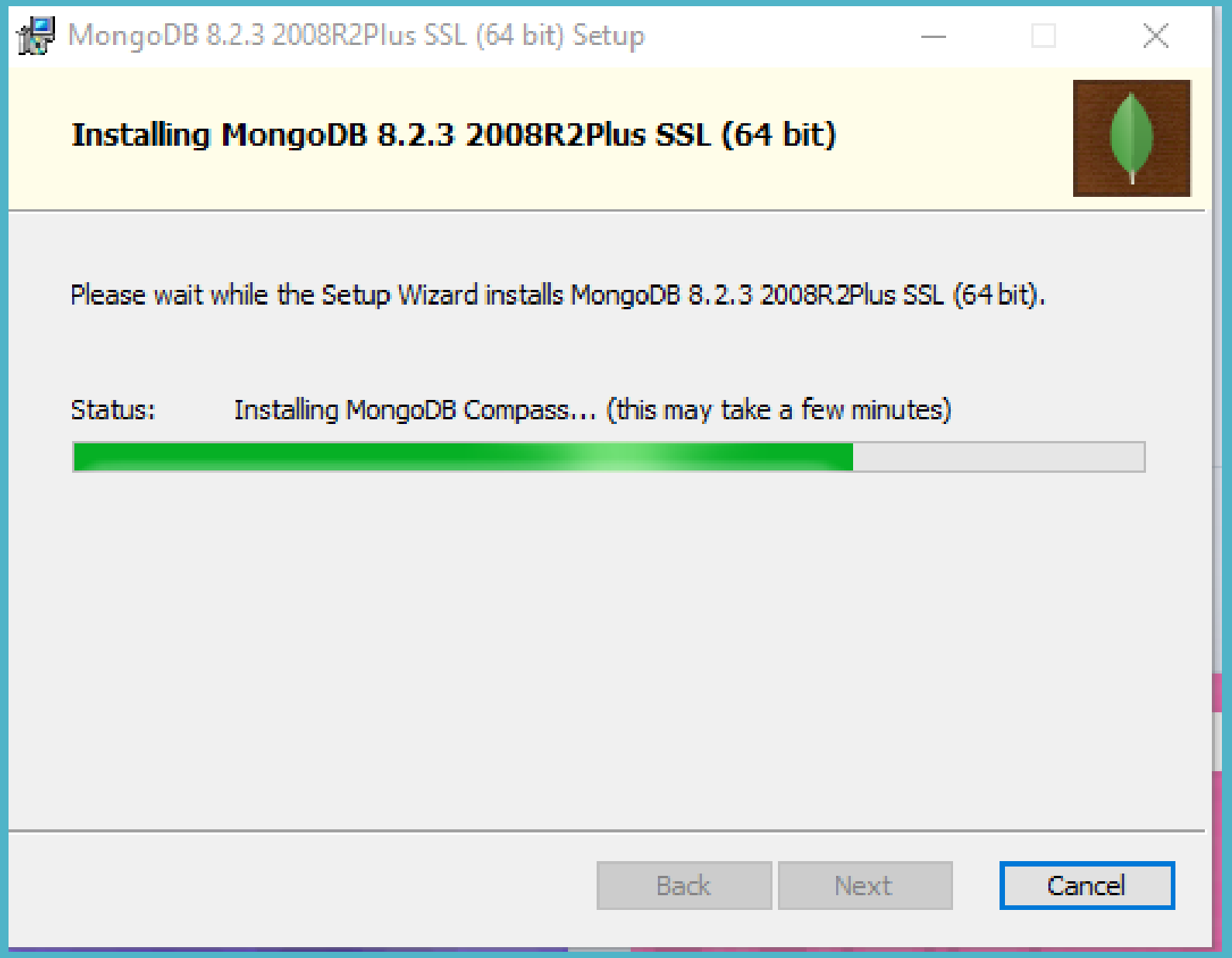


Download Community

Explore fully managed MongoDB →


MongoDB 8.2.3 2008R2Plus SSL (64 bit) Setup
Choose Setup Type
Choose the setup type that best suits your needs
Complete
All program features will be installed. Requires the most disk space.
Recommended for most users.
Custom
Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.
The Mongo Shell must be installed separately for Windows installations. Download Now
Back
Next
Cancel


MongoDB 8.2.3 2008R2Plus SSL (64 bit) Service Customization
Service Configuration
Specify optional settings to configure MongoDB as a service.
Install MongoD as a Service
Run service as Network Service user
Run service as a local or domain user:
Account Domain:
.
Account Name:
MongoDB
Account Password:
Service Name:
MongoDB
Data Directory:
C:\Program Files\MongoDB\Server\8.2\data\
Log Directory:
C:\Program Files\MongoDB\Server\8.2\log\
< Back
Next >
Cancel


MongoDB 8.2.3 2008R2Plus SSL (64 bit) Setup
Installing MongoDB 8.2.3 2008R2Plus SSL (64 bit)
Please wait while the Setup Wizard installs MongoDB 8.2.3 2008R2Plus SSL (64 bit).
Status: Installing MongoDB Compass... (this may take a few minutes)
Back
Next
Cancel

Welcome to Compass

Build aggregation pipelines, optimize queries, analyze schemas, and more. All with the GUI built by - and for - MongoDB.

Start

To help improve our products, anonymous usage data is collected and sent to MongoDB in accordance with MongoDB's privacy policy.

Manage this behaviour on the Compass Settings page.

MongoDB Compass

Connections Edit View Help

Compass

My Queries

Data Modeling

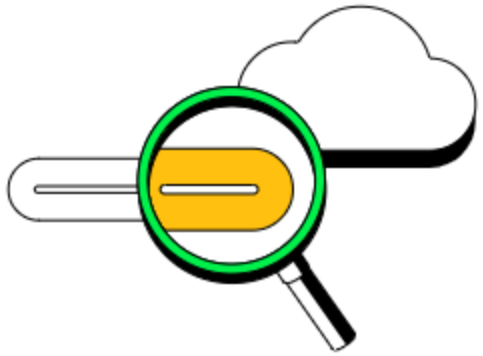
CONNECTIONS

Search connections

You have not connected to any deployments.

+ Add new connection

Welcome



Welcome to MongoDB Compass

To get started, connect to an existing server or

+ Add new connection

New to Compass and don't have a cluster?

If you don't already have a cluster, you can create one for free using MongoDB Atlas

CREATE FREE CLUSTER

New Connection

Manage your connection settings

URI

Edit Connection String

mongodb://localhost:27017/

How do I find my connection string in Atlas?

If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.
See example

Name

newconnect

Color

No Color

Favorite this connection

Favoriting a connection will pin it to the top of your list of connections

How do I format my connection string?

See example

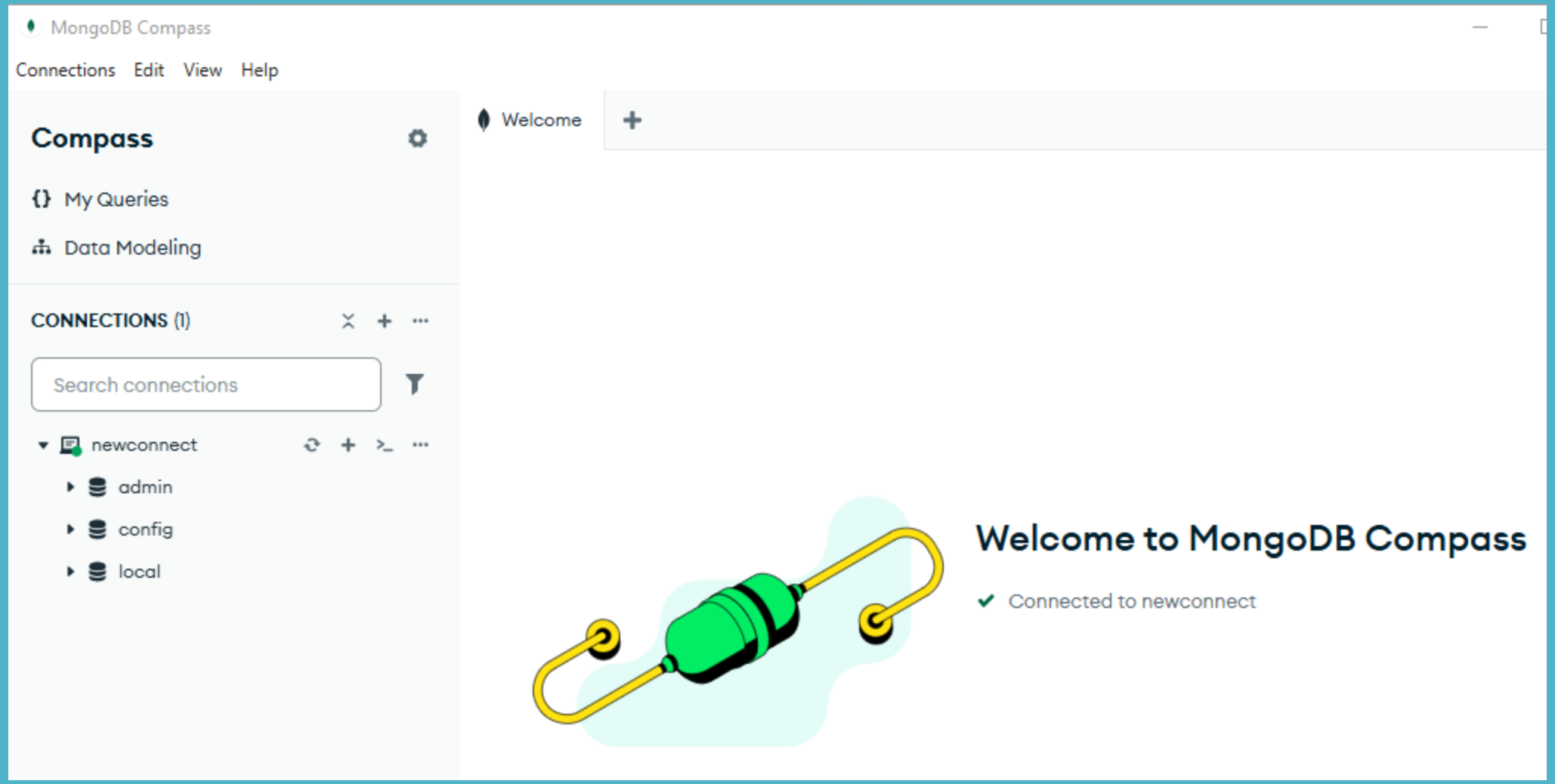
Advanced Connection Options

Cancel

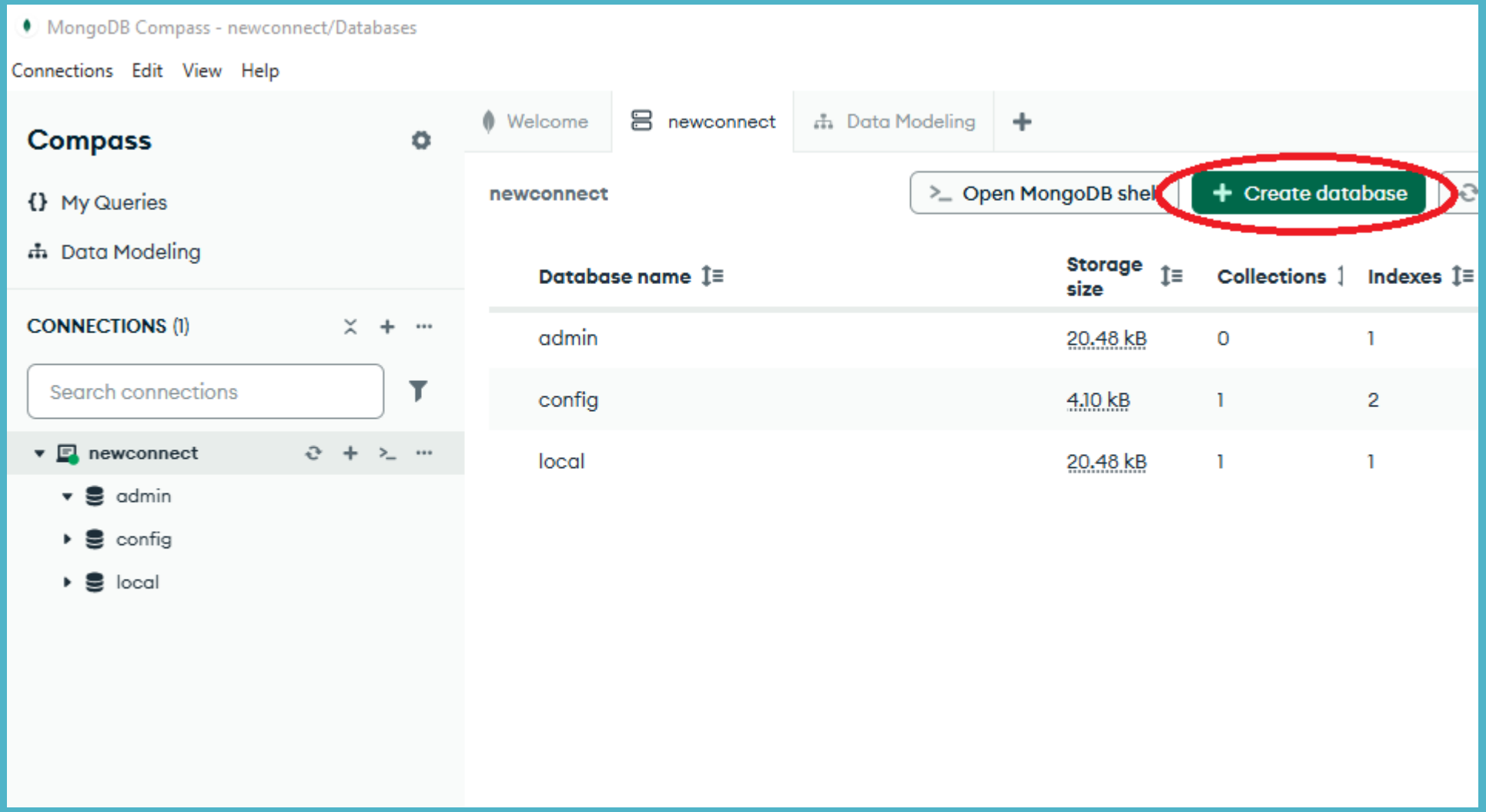
Save

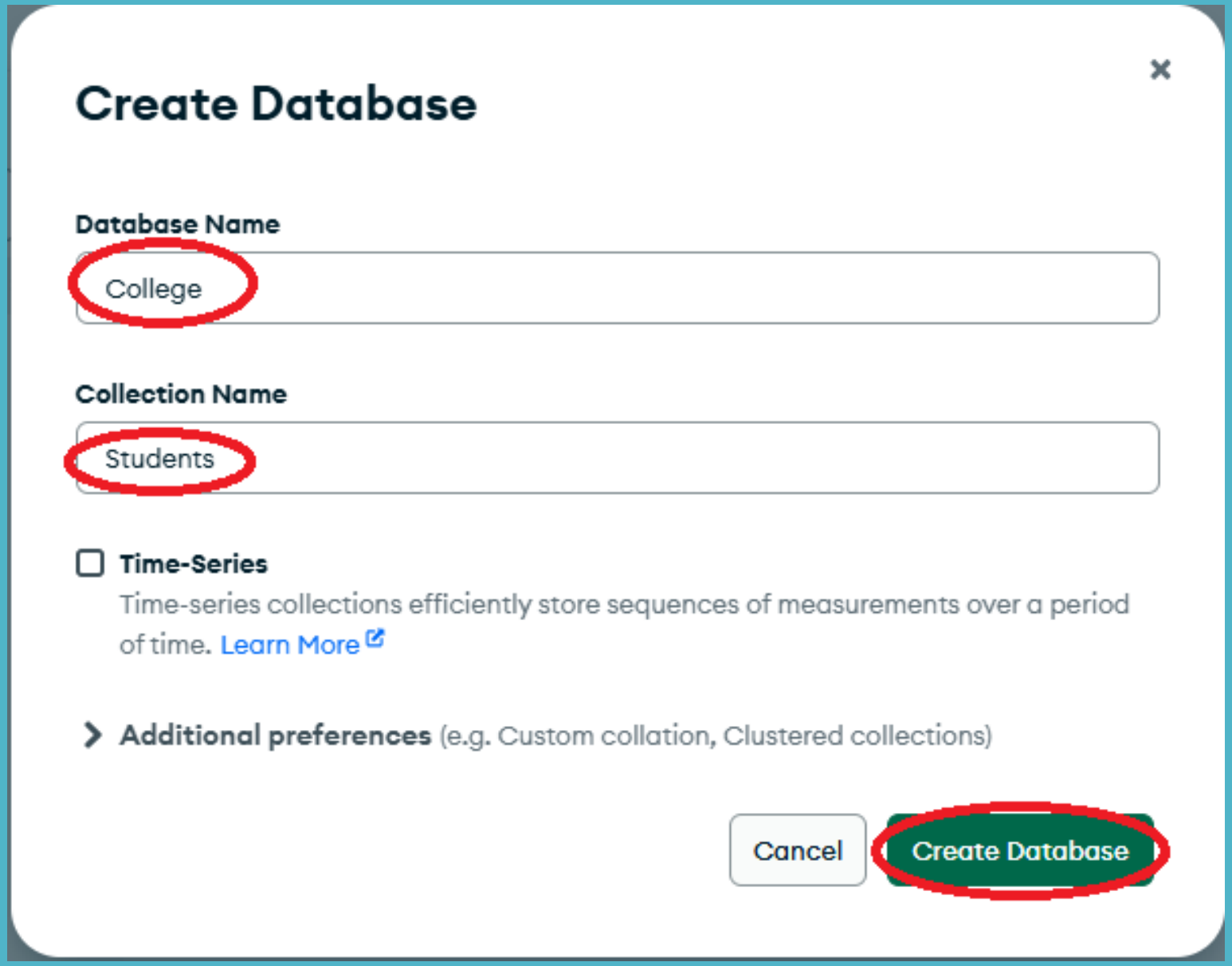
Connect

Save & Connect

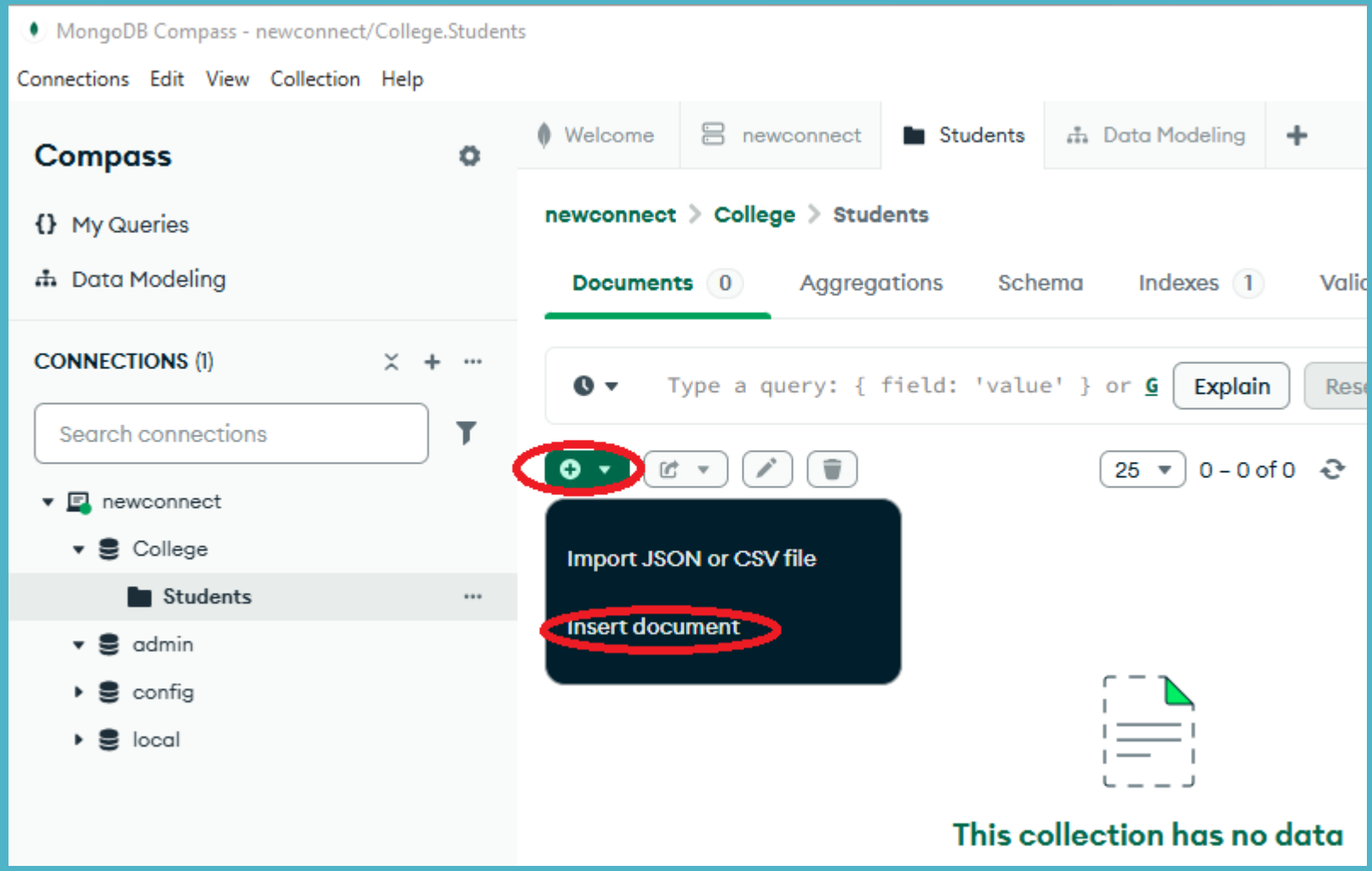

MongoDB Compass
Connections Edit View Help
Compass
My Queries
Data Modeling
CONNECTIONS (1)
Search connections
newconnect
admin
config
local
Welcome
Welcome to MongoDB Compass
Connected to newconnect

Создание базы данных «College» «Students»




Create Database
Database Name
College
Collection Name
Students
Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. Learn More
Additional preferences (e.g. Custom collation, Clustered collections)
Cancel
Create Database

Добавление документа в коллекцию:



Insert Document

To collection College.Students

VIEW {}

```
1   _id: ObjectId('695bccb4124e03d7131a0e1a')        ObjectId
2   name : "Petroff"                                 String
3   age : 20                                         Int32
```

Cancel Insert

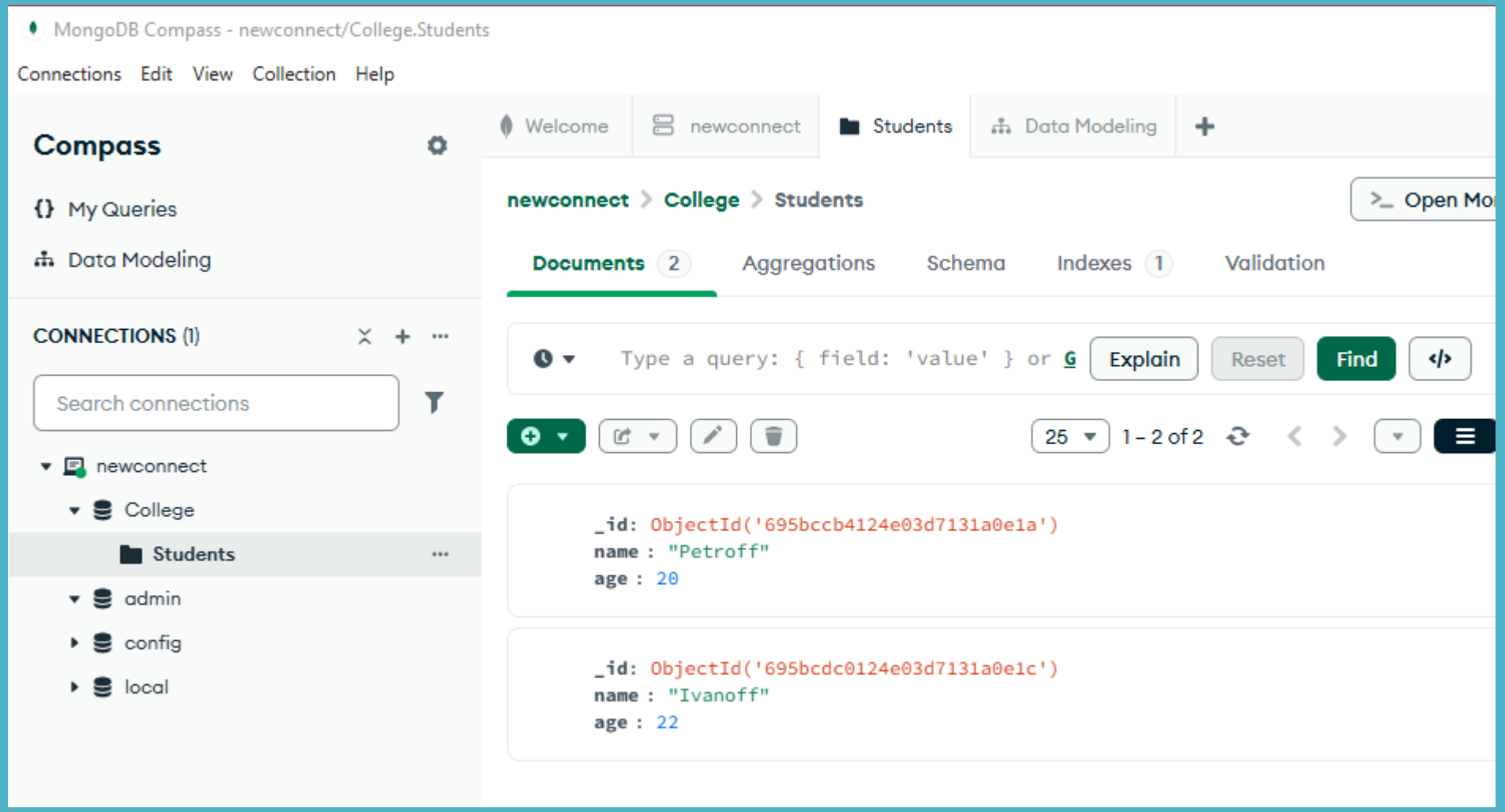
Insert Document

To collection College.Students

VIEW {}

```
{
  "name": "Ivanoff",
  "status": 22
}
```

Cancel Insert


MongoDB Compass - newconnect/College.Students
Connections Edit View Collection Help
Compass
My Queries
Data Modeling
CONNECTIONS (1)
Search connections
newconnect
College
Students
admin
config
local
Welcome
newconnect
Students
Data Modeling
newconnect > College > Students
Open Mo
Documents 2
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or G
Explain
Reset
Find
25
1 – 2 of 2
_id: ObjectId('695bccb4124e03d7131a0e1a')
name : "Petroff"
age : 20
_id: ObjectId('695bcdc0124e03d7131a0e1c')
name : "Ivanoff"
age : 22

Список литературы:

1. Видеокурс C#.
2. https://metanit.com/sharp/windowsforms/3.1.php
3. Видеокурс C# Windows Forms

Материалы лекций:

https://github.com/ShViktor72/Education2025

Обратная связь:

colledge20education23@gmail.com

Задание на дом:

Задание 1. Рисование примитивов

Создайте новый проект Windows Forms.

При загрузке формы, должны отобразиться:

- Синяя горизонтальная линия длиной 200 пикселей.
- Красный прямоугольник (100x50 пикселей).
- Зеленый закрашенный круг (диаметром пикселей).
- Черный текст "Графика в C#" шрифтом Arial, 16 px.
- Прямоугольник с линейным градиентом от синего к белому
- Желтый треугольник

Требования:

- Рисование должно выполняться в обработчике события Paint.
- Все фигуры должны быть нарисованы с помощью Graphics.

Задание 2. Отображение изображения

Создайте форму с компонентом PictureBox, который загружает изображение из файла.

Добавьте кнопку "Выбрать изображение", при нажатии которой пользователь может выбрать файл (JPG/PNG).

Загруженное изображение должно отображаться в PictureBox в режиме Zoom.

Требования:

- Использовать OpenFileDialog для выбора файла.
- Загружать изображение в PictureBox программно.
- Настроить SizeMode так, чтобы изображение вписывалось в PictureBox без искажений.