

サイバー大学IT総合学部

専門応用科目

JavaScriptフレームワークによるWebプログラミング

第7回 データベース実習

小園井康志

第7回 学習目標

- データベースの概念を理解し説明できる
- ドキュメントデータベース MongoDBについて理解し説明できる
- Node.jsからMongoDBへの接続を理解し説明できる
- Node.jsからMongoDBへの接続プログラムの作成ができる

第7回 授業構成

- 第1章 データベース概要
- 第2章 ドキュメントDB mongoDBについて
- 第3章 Node.jsからmongoDBへの接続

JavaScriptフレームワークによるWebプログラミング
第7回 データベース実習

第1章 データベース概要

第1章 学習目標

データベースについて理解し説明できる

- データベースとは何か（何ができるか）
- データベースの種類にはどのようなものがあるか
- それぞれのデータベースの種類には、
どのようなメリット・デメリットがあるか

REST API 前回のおさらい

作業内容

- REST APIについて学習
- Postmanについて学習
- Node.jsでREST APIを作成

データベースについて

この章では、データベースについて、以下の流れで説明する。

1. データベース (DB) とは
2. データベースでできること
3. SQLとは
4. データベースの種類
5. データベースの種類比較 (特徴、メリット、デメリット)

データベース（DB）とは

データベースとは、ある特定のデータを収集し、使いやすい形に格納した情報群のこと。データは何かしらの判断を行う際に非常に参考になる。しかし、適切に整理されていないままだと、情報を瞬時に取り出すことができない。

必要な情報を見やすい形で抽出することを可能にしたのが、データベース。データをそのまま表示することも可能であり、グラフのように視覚的に整えることもできる。

現代のビジネスにおいては、テクノロジーの発展によりビッグデータの解析など、さまざまなデータを処理してビジネスに役立てられるようになり、大量のデータを扱う機会が多くなっており、データベースの利用は必須といえる。

データベースでできること

データベースでは、膨大なデータの管理や必要なデータの検索、データのグループ化・編集が容易にできる。データの管理では、バラバラとなっている情報をデータベースにまとめることで、データの抽出が容易になる。またデータ検索では、必要に応じて文字順での並び替えやキーワード検索により、目的のデータを探ることができる。データ編集では、共通しているデータをグループ化することで、目的に応じてデータを編集することができる。

SQLとは

SQL、Structured Query Languageはデータベース（RDBMS）を操作するための言語。データベースにデータを挿入したり、検索したりする際に利用する。データベースのなかには、数万・数百万件ものデータが保存されているが、SQLを使うことで効率的に操作をすることが可能。

SQLは国際標準化されているため、さまざまなデータベースで利用できる。有名なデータベースとしては、Oracle、MySQL、PostgreSQL、SQLiteなどが、いずれもSQLで操作可能。

SQLとは

SQLでデータベースに対して行える操作として、主に次のようなものがある。
データベースに関わるあらゆる操作をSQLで行う。

- ・ データの検索
- ・ データの追加
- ・ データの更新
- ・ データの削除
- ・ テーブルの作成
- ・ テーブルの削除
- ・ テーブルの主キーの設定
- ・ ユーザー権限の付与

データベースの種類

主要なデータベースには、以下の4種類がある。

- 階層型
- ネットワーク型
- 関係型（リレーショナルデータベース）
- NoSQL

データベースの種類：階層型

- 階層型データベースの特徴

- 会社の組織図のように階層ごとに分かれている。
- 特徴的なのはツリー（木）構造で、一本の幹から多数の枝が分かれ、さらに多数の葉をつけているようなデータベースを構築している。
- 親データと子データが「1対多数」の関係を持つ。

メリット

検索する際に、検索ルートが一つだけに限られるため、速度が速い

デメリット

常に検索ルートが一つのため、重複したデータが登録されることがある点。
さらに非常に速いデータベースでは、データを登録する度にルートを再構築する場合もある。

データベースの種類：ネットワーク型

- ネットワーク型データベースの特徴
 - 先ほどの階層型データベースでは難しい「多数対多数」のデータベースを構築することができる。
 - 例えば、ある社員が異なる2事業部を兼務しているイメージ。
 - ツリー構造の階層型データベースとは異なり、網目状になる。

メリット

階層型データベースで問題になっていたデータの重複を避けられる。
子ノードが複数の親ノードを持てるようになっており、複数の要素を持つ子ノードを1つのノードだけで表現でき、データの重複登録を防げる。そのため、階層型データベースよりも自然な形でデータを格納できる。

デメリット

階層型データベースと同じくデータの柔軟な取り扱いが難しいプログラムがデータ構造に依存していて、データ構造を理解していないとデータへのアクセスが難しいため、ネットワーク型データベースの使用には、高度な知識が必要。

データベースの種類：関係型

- 関係型（リレーショナルデータベース）データベースの特徴
 - Microsoft Excelのような表で構成されたデータベース。
 - 関係型データベースは行と列を持ち、表形式でデータの関係性を示す。
 - データ検索のほかにSQLを用いたデータのアクセスが可能になる。

メリット

柔軟なデータの取り扱いができることや、複雑なデータに関連性を扱えること、データ処の一貫性を保てることなどのメリットがある。特定のデータを操作したり、条件を付けてデータを検索することが可能。また、複数の表と組み合わせ、より複雑なデータの取り扱いもできる。

デメリット

プログラムが複雑化しやすく、処理速度が遅くなる傾向にあるというデメリットがある。複雑な関係性を持つデータの格納ができるゆえ、大規模なデータを扱ったりする際に、プログラムが複雑になってしまう。

データベースの種類：NoSQL

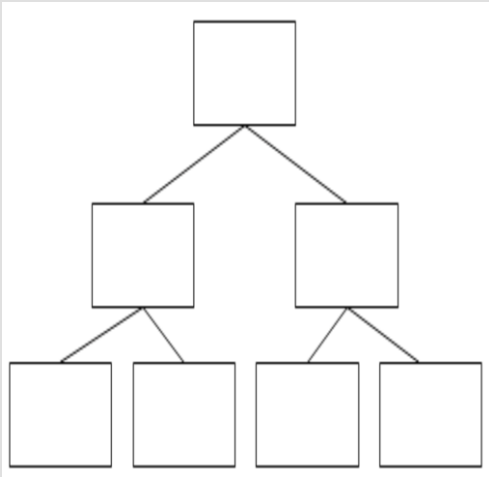
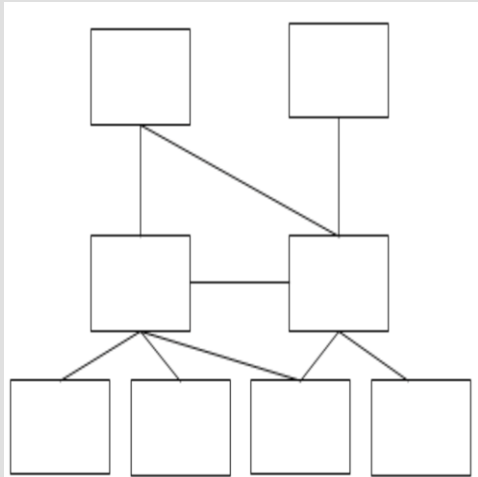
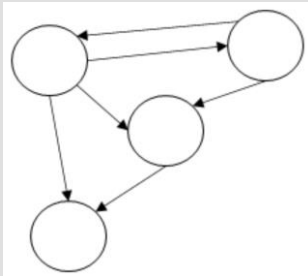

- NoSQLデータベースの特徴

- 比較的新しいデータベースで、膨大なデータをより速い速度で処理するために開発されたもの。
- NoSQLの「No」は、否定の意味ではなく「Not only」の意味で使用されている。
- このデータベースでは処理能力の重点を置いているため、データの整合性よりも処理パフォーマンスを重視する場合に使用される

リレーショナルデータベースよりも大規模なデータを扱えることや、リアルタイムに近い応答性能があることなどのメリットがある。大規模なデータを扱えることからビッグデータの処理などにも使用されている。また、高度なデータベースチューニングの技術がなくてもデータの高速な処理が可能で、高い技術を持つエンジニアがいなくても使用できる。

NoSQLは、同時実行制御のための機能が緩く、大量のデータに素早く対処するために、データの整合性を保つのが難しくなっている。また、NoSQLはSQL言語を使用しないので、複雑な条件を指定した検索などができず、データの加工が難しくなっている。

データベースの種類イメージ図

| 階層型 | ネットワーク型 | 関係型 (リレーショナルデータベース) | NoSQL | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---|--|-------|----|-------|-------|---|------|----|-----|---|------|----|-----|---|-----|----|-----|---|-----|-------|-----|-------|-----|-------|-----|-------|
|  |  | <table><tr><th>ID</th><th>名前</th><th>組織コード</th><th>役職コード</th></tr><tr><td>1</td><td>山田太郎</td><td>10</td><td>100</td></tr><tr><td>2</td><td>高橋花子</td><td>20</td><td>200</td></tr><tr><td>3</td><td>鈴木隆</td><td>30</td><td>300</td></tr></table> | ID | 名前 | 組織コード | 役職コード | 1 | 山田太郎 | 10 | 100 | 2 | 高橋花子 | 20 | 200 | 3 | 鈴木隆 | 30 | 300 | <table><tr><th>key</th><th>value</th></tr><tr><td>key</td><td>value</td></tr><tr><td>key</td><td>value</td></tr><tr><td>key</td><td>value</td></tr></table>   | key | value | key | value | key | value | key | value |
| ID | 名前 | 組織コード | 役職コード | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 山田太郎 | 10 | 100 | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 高橋花子 | 20 | 200 | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 鈴木隆 | 30 | 300 | | | | | | | | | | | | | | | | | | | | | | | | |
| key | value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| key | value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| key | value | | | | | | | | | | | | | | | | | | | | | | | | | | |
| key | value | | | | | | | | | | | | | | | | | | | | | | | | | | |

データベースの種類比較:特徴

| 種類 | 特徴 |
|------------------------|--|
| 階層型 | 階層型データベースは、データのことをノードと呼び、あるノードから別のノードを派生させて、ツリー上に展開してデータを保存しているデータベースのこと。階層型データベースでは、派生前のデータを親ノード、派生後のデータを子ノードと呼び、親ノードは複数個のノードを持つことができ、子ノードは1つだけ親ノードを持つ。 |
| ネットワーク型 | ネットワーク型データベースは、階層型データベースと同じようにデータをノードで表し、あるノードから別のノードを派生させて、ノードのつながりが網目状になるように展開されるデータベースのこと。階層型データベースとは異なり、1つの子ノードが複数の親ノードを持てるような構造で、1つの親ノードも複数の子ノードを持てる。 |
| 関係型 (リレーショナルデータベース) | リレーショナルデータベースは、レコードという行とフィールドという列で構成され、テーブルと呼ばれる表形式のデータベース。表形式のデータを互いに関連付けて、データ同士の関連性から新しい表を作成したりでき、エクセルを扱うように使用できるので、リレーショナルデータベースはさまざまな場面で使用されている。 |
| NoSQL | NoSQLとは、データベースを操作するための言語であるSQLを使用せず、リレーショナルデータベースで扱うのが難しい大規模なデータに対応するためのデータベースのこと。NoSQLにはさまざまな種類のものがあり、キーバリュ型やカラム指向型、ドキュメント指向型、グラフ型などの種類がある。 |

データベースの種類比較: メリット

| 種類 | メリット |
|------------------------|--|
| 階層型 | 階層型データベースには、アクセスするための速度が速いというメリットがある。階層型データベースでは、データの参照などのデータを操作する際に、データにアクセスするためのノードのルートが限定されるので、データにアクセスするための速度が速くなる。関連性の高いデータの格納や、素早いデータの検索が必要なサービスの構築などに役立つ。 |
| ネットワーク型 | ネットワーク型データベースは、階層型データベースで問題になっていたデータの重複を避けられるというメリットがある。ネットワーク型データベースは、子ノードが複数の親ノードを持てるようになっており、複数の要素を持つ子ノードを1つのノードだけで表現でき、データの重複登録を防げる。そのため、階層型データベースよりも自然な形でデータを格納できる。 |
| 関係型 (リレーショナルデータベース) | リレーショナルデータベースには、柔軟なデータの取り扱いができることや、複雑なデータの関連性を扱えること、データ処理の一貫性を保てることなどのメリットがある。特定のデータを操作したり、条件を付けてデータを検索することが可能。また、複数の表と組み合わせて、より複雑なデータの取り扱いもできる。 |
| NoSQL | NoSQLには、リレーショナルデータベースよりも大規模なデータを扱えることや、リアルタイムに近い応答性能があることなどのメリットがある。リレーショナルデータベースよりも大規模なデータを扱えるため、ビッグデータの処理などに使用されている。また、高度なデータベースチューニングの技術がなくてもデータの高速な処理が可能で、高い技術を持つエンジニアがいなくても使用できる。 |

データベースの種類比較:デメリット

| 種類 | デメリット |
|------------------------|--|
| 階層型 | 階層型データベースのデメリットとして、複数の親ノードが必要なデータの入力の難しさや、データを管理する際の柔軟性に欠けていることなどがある。基本的に親ノードは子ノードに対して1つだけなので、複数の親ノードが必要な場合には重複登録が必要で、不自然なデータになってしまう。また、データの追加や削除などの際に、ルートを再登録する必要があり、データを管理するための柔軟性にも欠けている。 |
| ネットワーク型 | ネットワーク型データベースのデメリットには、階層型データベースと同じく、データの柔軟な取り扱いが難しいことなどがある。プログラムがデータ構造に依存していて、データ構造を理解していないとデータへのアクセスが難しいという問題がある。そのためネットワーク型データベースの使用には、高度な知識が必要。 |
| 関係型 (リレーショナルデータベース) | リレーショナルデータベースには、プログラムが複雑化しやすく、処理速度が遅くなる傾向にあるというデメリットがある。複雑な関係性を持つデータの格納ができるゆえ、大規模なデータを扱ったりする際に、プログラムが複雑になってしまう。 |
| NoSQL | NoSQLのデメリットとしては、データの整合性が保てないことや、データの加工が難しいことなどが挙げられる。NoSQLは、同時実行制御のための機能が緩く、大量のデータに素早く対処するために、データの整合性を保つのが難しくなっている。また、NoSQLはSQL言語を使用しないので、複雑な条件を指定した検索などができず、データの加工が難しくなっている。 |

最近のデータベースのトレンドや進化

最近のデータベースのトレンドや進化には以下のようなものがある。これらのトレンドは、データの増加や多様化に対応し、高可用性、スケーラビリティ、セキュリティ、柔軟性などのニーズに応えるために発展している。

- **NoSQLデータベースの普及**

NoSQLデータベースは、従来のリレーショナルデータベースに代わる新しい種類のデータベース。

NoSQLデータベースは、高速でスケーラブルであることが特徴であり、非構造化データやビッグデータを扱うことができる。

- **クラウドデータベースの増加**

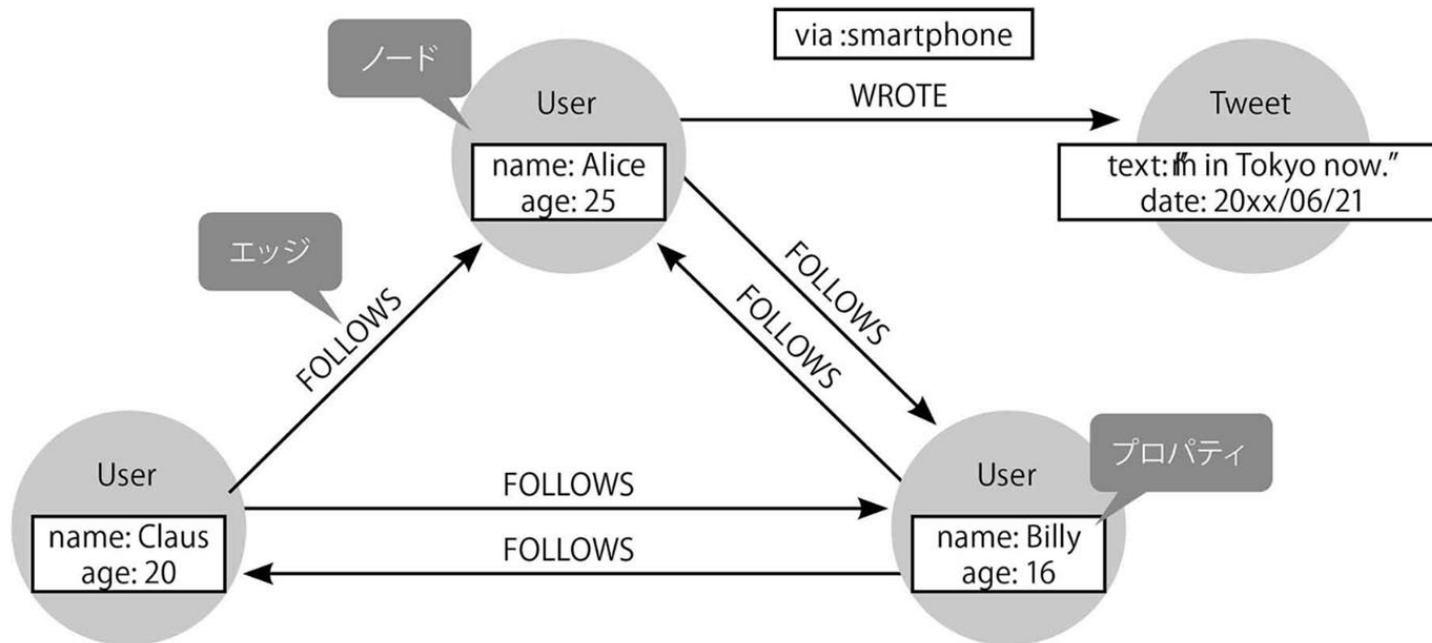
クラウドデータベースは、クラウド上にあるデータベースサービスであり、クラウドの利点を活用することで、高可用性、スケーラビリティ、セキュリティ、柔軟性などのメリットを提供。

- **グラフデータベースの登場**

グラフデータベースは、ノードとエッジの関係性を中心にデータを表現するデータベース。

グラフデータベース

ここで言うグラフとはグラフ理論の「グラフ」であり、「ネットワーク」という言葉のイメージに近いものです。ノード（頂点）とエッジ（辺）とプロパティ（属性）の3つの要素から構成され、ノード間の関係を管理することに特化したデータベースです。



最近のデータベースのトレンドや進化

•分散データベースの普及

分散データベースは、複数のノードにデータを分散させ、高可用性とスケーラビリティを実現するデータベース。

•AI/機械学習に対応したデータベース

AI/機械学習に対応したデータベースは、大量のデータを扱い、高速な分析と予測を可能にするデータベース。

•サーバーレスデータベースの普及

サーバーレスデータベースは、データベースの運用やスケーリングに必要なインフラストラクチャを提供するクラウドサービスであり、データベースの管理が不要であるため、開発者はアプリケーションの開発に注力することができる。

•マルチモデルデータベースの登場

マルチモデルデータベースは、リレーショナルデータベース、グラフデータベース、ドキュメントデータベースなど、複数のデータベースモデルをサポートするデータベース。

最近のデータベースの代表的なもの

2023年3月時点

最近のデータベースの代表的な製品・サービスには以下のようなものがある。

- NoSQLデータベース: MongoDBやCassandra、Couchbase。
- クラウドデータベース: AWSのDynamoDB、AzureのCosmos DB、Google CloudのFirestore。
- グラフデータベース: Neo4jやAmazon Neptune。
- 分散データベース: Apache CassandraやApache HBase、Amazon DynamoDB。
- AI/機械学習に対応したデータベース: Google CloudのBigQuery ML、Amazon Redshift ML、Oracle Autonomous Database。
- サーバーレスデータベース: AWSのDynamoDBやAurora Serverless、Microsoft AzureのCosmos DB。
- マルチモデルデータベース: ArangoDB、MarkLogic、OrientDB

第1章 まとめ

データベースについて学習した。

- データベース（DB）とは
- データベースでできること
- SQLとは
- データベースの種類
- データベースの種類比較（特徴、メリット、デメリット）

JavaScriptフレームワークによるWebプログラミング
第7回 データベース実習

第1章 データベース概要

終わり

JavaScriptフレームワークによるWebプログラミング
第7回 データベース実習

第2章 ドキュメントDB MongoDB概要

第2章 学習目標

ドキュメントDB MongoDBについて理解し説明できる

- MongoDBとその特徴
- MongoDBを使用したアプリケーション開発とその用途
- MongoDBのデータベース構造
- MongoDBのクラウドサービス

MongoDBとその特徴

- MongoDBは、オープンソースのドキュメント指向データベース管理システム。MongoDBは、リレーショナルデータベースではなく、ドキュメント指向データベース。
ドキュメント指向データベースは、JSON形式で表現されるドキュメントを保存するためのデータベース。
- MongoDBの主な特徴は、水平方向にスケラブルであることであり、データベースを分散処理することができる。また、MongoDBは、高速な読み取りおよび書き込み操作を提供するため、大量のデータを処理するのに適している。MongoDBは、データの柔軟なスキーマ設計が可能であり、アプリケーションの要件に合わせた柔軟なデータ構造を設計することができる。

スケーラビリティ

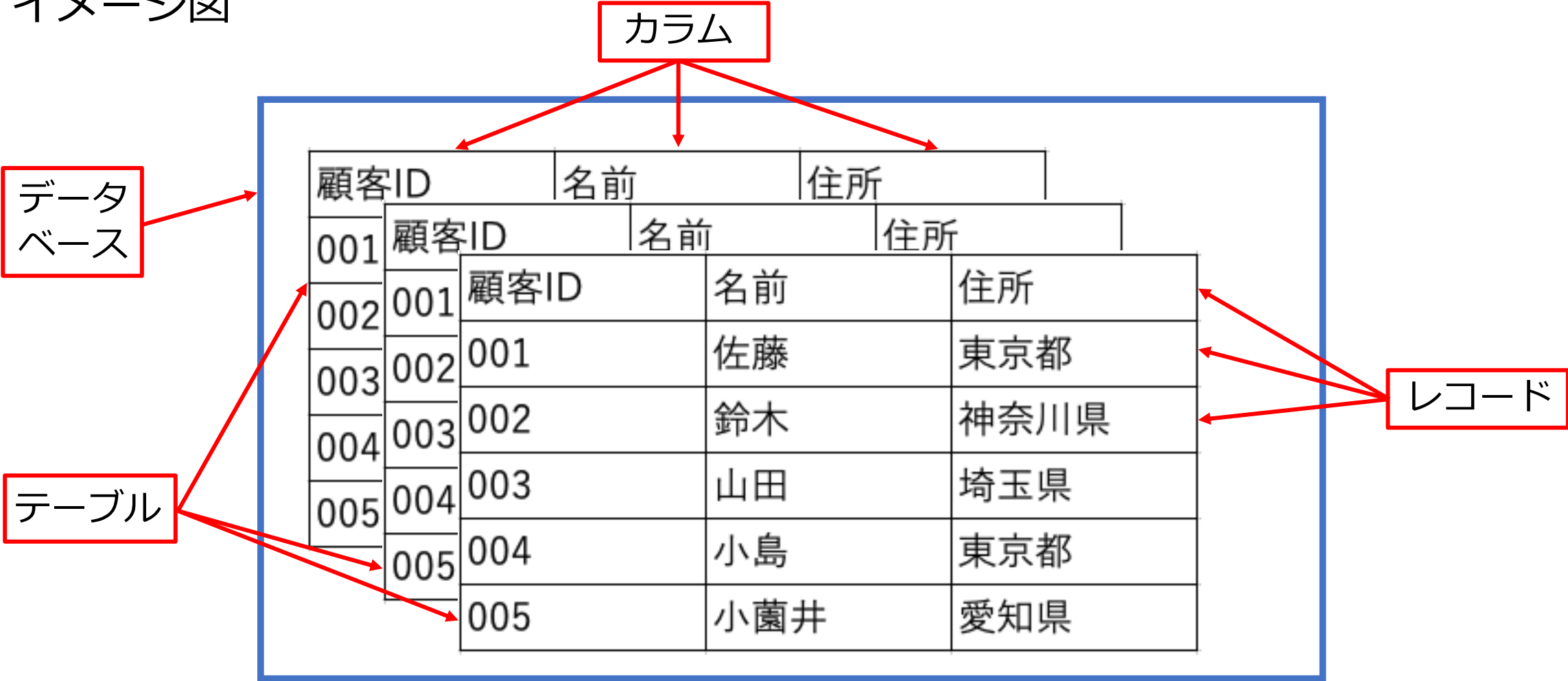
- スケーラビリティとは、システムの規模（スケール）の変化に柔軟に対応できる度合いのこと
- 垂直方向（スケールアップ）
負荷の増加に対してシステムのハードウェアの性能を向上させて対応する拡張方法
- 水平方向（スケールアウト）
負荷の増加に対してシステムのハードウェアの数を増やす方向で対応するもの

MongoDBを使用したアプリケーション開発とその用途

- MongoDBは、多くのプログラミング言語で使用されているため、アプリケーション開発に適している。MongoDBには、豊富な機能があり、インデックス、レプリケーション、シャーディングなどがサポートされている。
- MongoDBは、Webアプリケーション、ゲーム、モバイルアプリケーション、IoTデバイス、メッセージングアプリケーションなど、多様なアプリケーションに使用されている。

RDBについて

イメージ図



MongoDBのデータベース構造

MongoDBのデータベース構造は以下の図のようになっている。
RDBと同じく、MongoDBでも最上位の要素はデータベース。
RDBのテーブルに該当するものがコレクション。
コレクションには、ドキュメント（JSON）を複数保存することができるが、
テーブルとは違いスキーマ定義はない。そのため、保存するのはどんな構造
のJSONでもOK。RDBの行に該当するものがドキュメント（JSON）。
JSONをそのまま挿入でき、フィールド単位の更新が可能。

| RDB | MongoDB |
|-------------|------------------|
| データベース | データベース |
| テーブル | コレクション |
| レコード (行) | ドキュメント (JSON) |
| カラム (列) | フィールド |



JSON:

```
[  
  {"id" : "1",  
   "name" : "osonoi"},  
  {"id" : "2",  
   "name" : "kojima"}  
]
```

上記は2つのドキュメントが
あってid, nameという
フィールドがある

MongoDBのクラウドサービス

MongoDBのクラウドサービスには、以下のようなものがある。これらのクラウドサービスは、MongoDBのインスタンスを簡単にセットアップし、スケーラブルで信頼性の高いデータベース環境を提供するための機能を提供している。

- **MongoDB Atlas - MongoDB Atlas**

MongoDBの公式クラウドサービス。Atlasは、MongoDBのクラウド版を提供し、デプロイメント、管理、モニタリング、バックアップ、セキュリティなどの機能を提供している。Atlasは、AWS、Azure、GCP、Oracle Cloudなどの主要なクラウドプロバイダーで利用可能。

- **mLab – mLab**

MongoDBのクラウドホスティングサービスであり、MongoDBのインスタンスをすばやく簡単にセットアップすることができる。mLabは、AWS、Azure、GCPなどの主要なクラウドプロバイダーで利用可能。

MongoDBのクラウドサービス

- **ScaleGrid – ScaleGrid**

MongoDBや他のデータベースの管理、モニタリング、自動スケーリング、高可用性などの機能を提供するマネージドデータベースサービス。ScaleGridは、AWS、Azure、GCP、DigitalOceanなどの主要なクラウドプロバイダーで利用可能。

- **Compose – Compose**

MongoDBなどのデータベースをホストするマネージドデータベースサービスであり、バックアップ、レプリケーション、高可用性などの機能を提供している。Composeは、AWS、Azure、GCP、IBM Cloudなどの主要なクラウドプロバイダーで利用可能。

第2章 まとめ

ドキュメントDB MongoDBについて学習した。

- MongoDBとその特徴
- MongoDBを使用したアプリケーション開発とその用途
- MongoDBのデータベース構造
- MongoDBのクラウドサービス

JavaScriptフレームワークによるWebプログラミング
第7回 データベース実習

第2章 ドキュメントDB MongoDB概要

終わり

JavaScriptフレームワークによるWebプログラミング
第7回 データベース実習

第3章

Node.jsからmongodbへの接続

第3章 学習目標

- Node.jsからMongoDBへの接続を理解し説明できる
- Node.jsからMongoDBへの接続プログラムの作成ができる

今回の概要

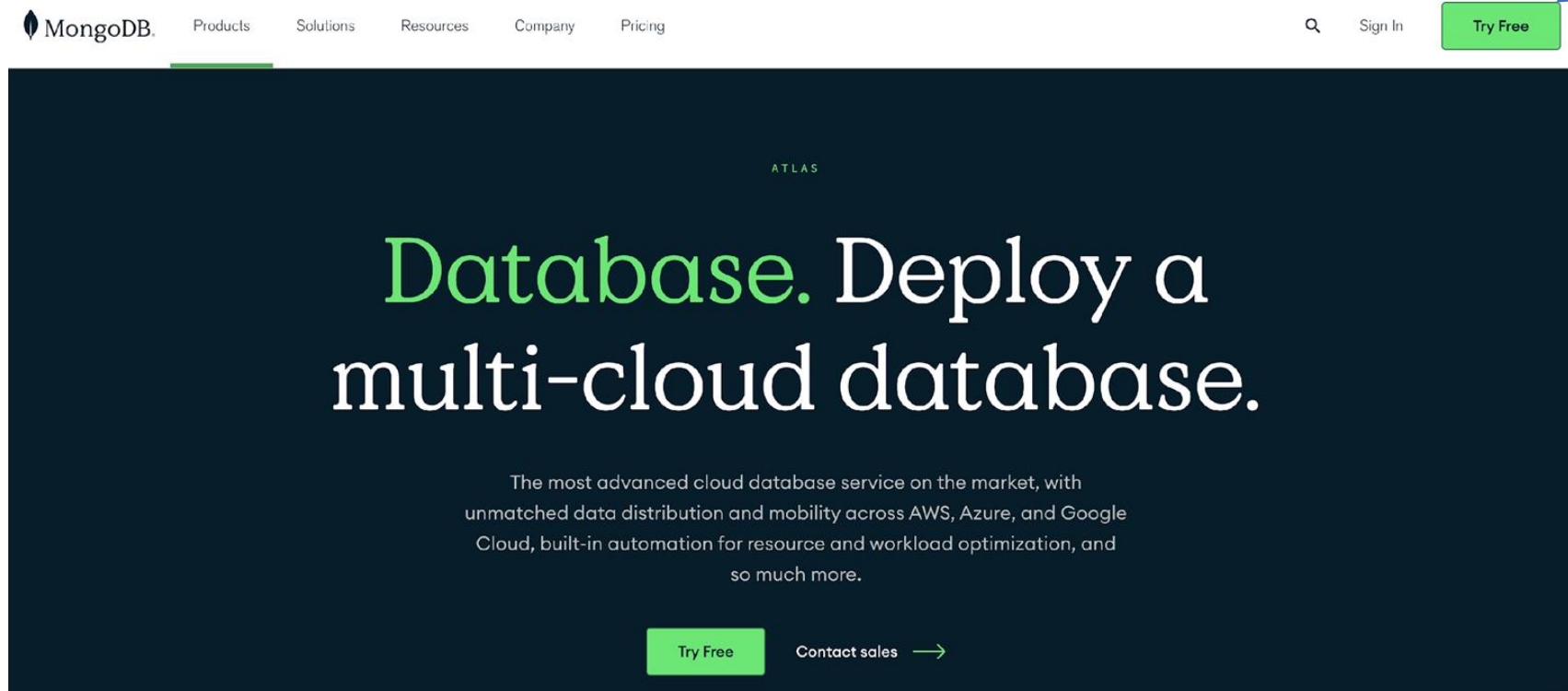
- Javascriptのプログラム（node.js）からネット上のMongoDBにアクセス
- データベースの中身进行操作（作成、編集、削除）



Atlas MongoDB

- ・ ネット上のサービス（SaaS）
- ・ 無料で利用可能

ここから
登録する



<https://www.mongodb.com/atlas/database>

事前準備

- MongoDBAtlasにアクセス
(事前にMongoDBAtlasの無料アカウント作成しデータベースを作成しておく)

左のメニューから、
DataBaseをクリック。
Connectをクリック。

The screenshot shows the MongoDB Atlas interface. In the left sidebar, under the 'DEPLOYMENT' section, the 'Database' option is highlighted with a red box. In the main content area, under the 'Cluster0' section, the 'Connect' button is highlighted with a red box. The page displays various metrics and a table of database deployments.

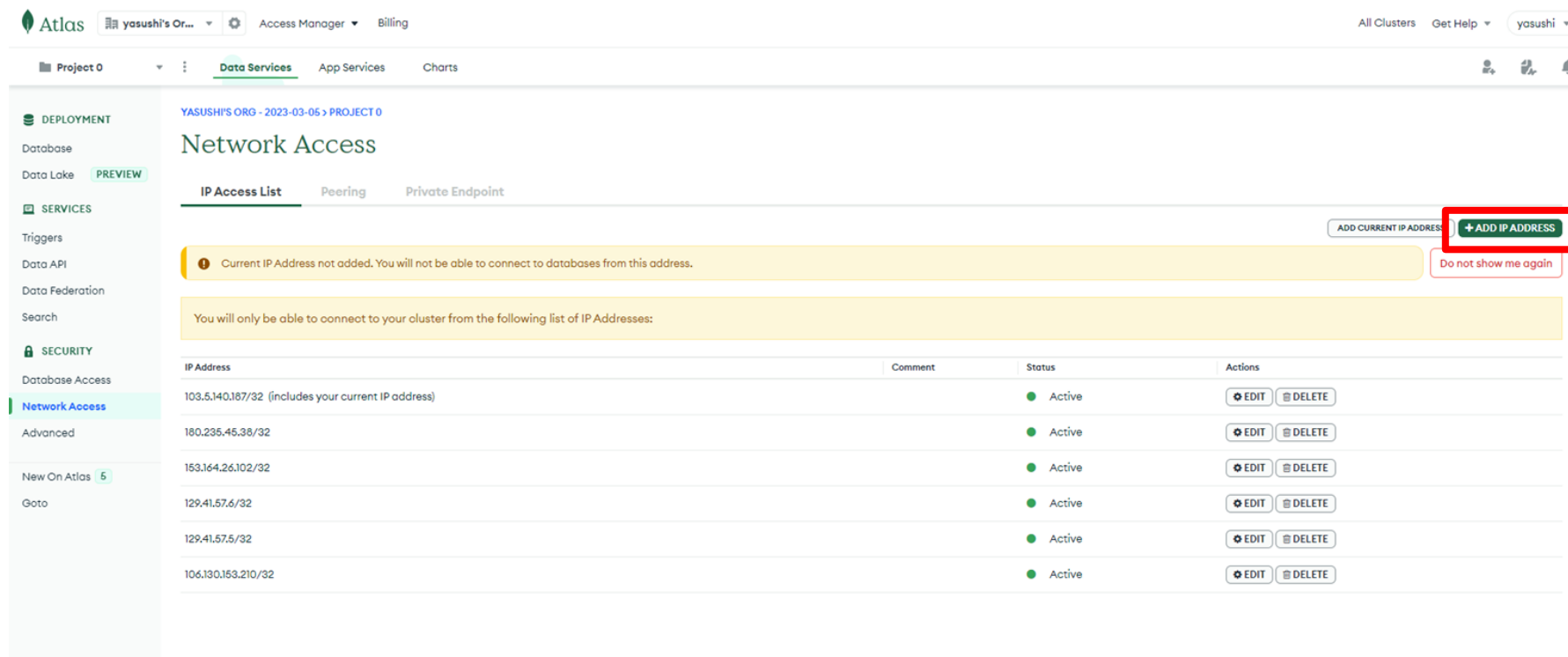
| VERSION | REGION | CLUSTER TIER | TYPE | BACKUPS | LINKED APP SERVICES | ATLAS SEARCH |
|---------|------------------------------|----------------------|-----------------------|----------|---------------------|------------------------------|
| 5.0.15 | AWS / Tokyo (ap-northeast-1) | M0 Sandbox (General) | Replica Set - 3 nodes | Inactive | None Linked | Create Index |

<https://www.youtube.com/watch?v=JBCykXLfvv0>

事前準備

- MongoDBAtlasへのアクセスを許可

デフォルトではサービスの外部からアクセスできないようになっている
自分のPCからMongoDBにアクセスできるように変更



The screenshot shows the MongoDB Atlas interface for a project named 'Project 0'. The left sidebar contains navigation options: DEPLOYMENT, SERVICES, and SECURITY. The main content area is titled 'Network Access' and shows the 'IP Access List' tab. A yellow banner at the top of the list states: 'Current IP Address not added. You will not be able to connect to databases from this address.' Below this, a table lists the allowed IP addresses. The first row is highlighted and includes the comment '(includes your current IP address)'. The '+ADD IP ADDRESS' button is highlighted with a red box.

| IP Address | Comment | Status | Actions |
|--------------------|------------------------------------|--------|---|
| 103.5.140.187/32 | (includes your current IP address) | Active | EDIT DELETE |
| 180.235.45.38/32 | | Active | EDIT DELETE |
| 153.164.26.102/32 | | Active | EDIT DELETE |
| 129.41.57.6/32 | | Active | EDIT DELETE |
| 129.41.57.5/32 | | Active | EDIT DELETE |
| 106.330.153.210/32 | | Active | EDIT DELETE |

<https://www.youtube.com/watch?v=JBCykXLfvv0>

事前準備

- MongoDBAtlasへのアクセスを許可

前ページのメッセージが出ない場合は左下の”Network Access”からも設定可能

Atlas yasushi's Or... Access Manager Billing All Clusters Get Help yasushi

Project 0 Data Services App Services Charts

DEPLOYMENT Database Data Lake PREVIEW SERVICES Triggers Data API Data Federation Search SECURITY Database Access **Network Access** Advanced New On Atlas 5 Goto

YASUSHI'S ORG - 2023-03-06 > PROJECT 0

Network Access

IP Access List Peering Private Endpoint

+ ADD IP ADDRESS

You will only be able to connect to your cluster from the following list of IP Addresses:

| IP Address | Comment | Status | Actions |
|---|---------|--------|---|
| 103.5.140.187/32 | | Active | EDIT DELETE |
| 180.235.45.38/32 (includes your current IP address) | | Active | EDIT DELETE |
| 153.164.26.102/32 | | Active | EDIT DELETE |
| 129.41.57.6/32 | | Active | EDIT DELETE |
| 0.0.0.0/0 (includes your current IP address) | | Active | EDIT DELETE |
| 129.41.57.5/32 | | Active | EDIT DELETE |
| 106.130.153.210/32 | | Active | EDIT DELETE |

<https://www.youtube.com/watch?v=JBCykXLfvv0>

事前準備

- MongoDBAtlasへのアクセスを許可

“Add IP address”をクリック、“0.0.0.0/0”を入力、外部からアクセス可能にする

×

Add IP Access List Entry

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more.](#)

Access List Entry: 0.0.0.0/0

Comment: Optional comment describing this entry

☐ This entry is temporary and will be deleted in 6 hours

Cancel Confirm

+ ADD IP ADDRESS

Actions

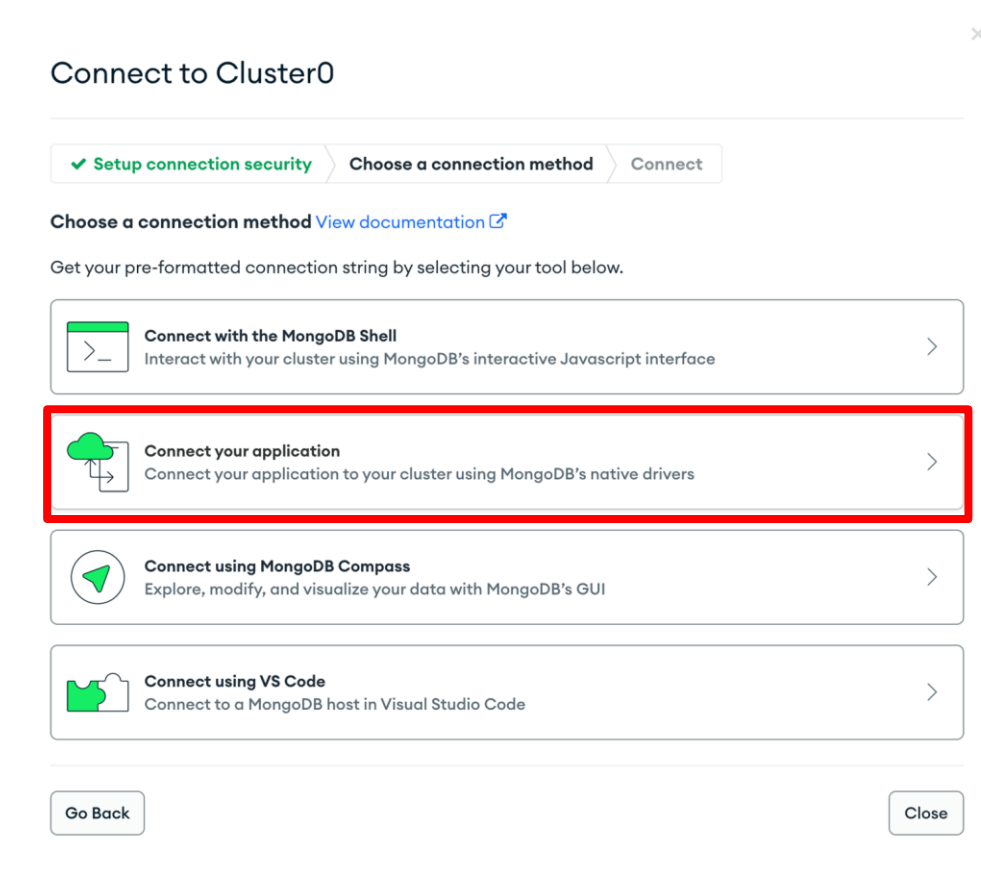
| | | |
|----------|--------|----------|
| ● Active | ⚙ EDIT | 🗑 DELETE |
| ● Active | ⚙ EDIT | 🗑 DELETE |
| ● Active | ⚙ EDIT | 🗑 DELETE |
| ● Active | ⚙ EDIT | 🗑 DELETE |

事前準備

- MongoDBAtlasで接続情報を確認

Connect your applications をクリック。

※今回はJavaScriptから接続するのでその接続方法をこちらで確認する



<https://www.youtube.com/watch?v=JBCykXLfvv0>

事前準備

- MongoDBAtlasで接続情報を確認

const url = の値を選択してコピーする。

```
const uri =  
"mongodb+srv://koh:<password>@cluster0.y5uxhta.mongodb.net/?retryWrites=true&w=majority";
```

Connect to Cluster0

✓ Setup connection security

✓ Choose a connection method

Connect

1 Select your driver and version

DRIVER

Node.js

VERSION

4.1 or later

2 Add your connection string into your application code

☒ Include full driver code example

```
const { MongoClient, ServerApiVersion } = require('mongodb');  
const uri = "mongodb+srv://koh:<password>@cluster0.y5uxhta.mongodb.net/?  
retryWrites=true&w=majority";  
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology:  
true, serverApi: ServerApiVersion.v1 });  
client.connect(err => {  
  const collection = client.db("test").collection("devices");  
  // perform actions on the collection object  
  client.close();  
});
```

Replace **<password>** with the password for the **koh** user. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close

事前準備

- MongoDBAtlasで接続情報を確認

const url = の値を選択してコピーする。

```
const uri =  
"mongodb+srv://koh:<password>@cluster0.y5uxhta.mongodb.net/?retryWrites=true&w=majority";
```

<password>データベース作成時のパスワードに置き換える
例: <password> → password

Connect to Cluster0

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER VERSION
Node.js 4.1 or later

2 Add your connection string into your application code

☒ Include full driver code example

```
const { MongoClient, ServerApiVersion } = require('mongodb');  
const uri = "mongodb+srv://koh:<password>@cluster0.y5uxhta.mongodb.net/?  
retryWrites=true&w=majority";  
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology:  
true, serverApi: ServerApiVersion.v1 });  
client.connect(err => {  
  const collection = client.db("test").collection("devices");  
  // perform actions on the collection object  
  client.close();  
});
```

Replace **<password>** with the password for the **koh** user. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close

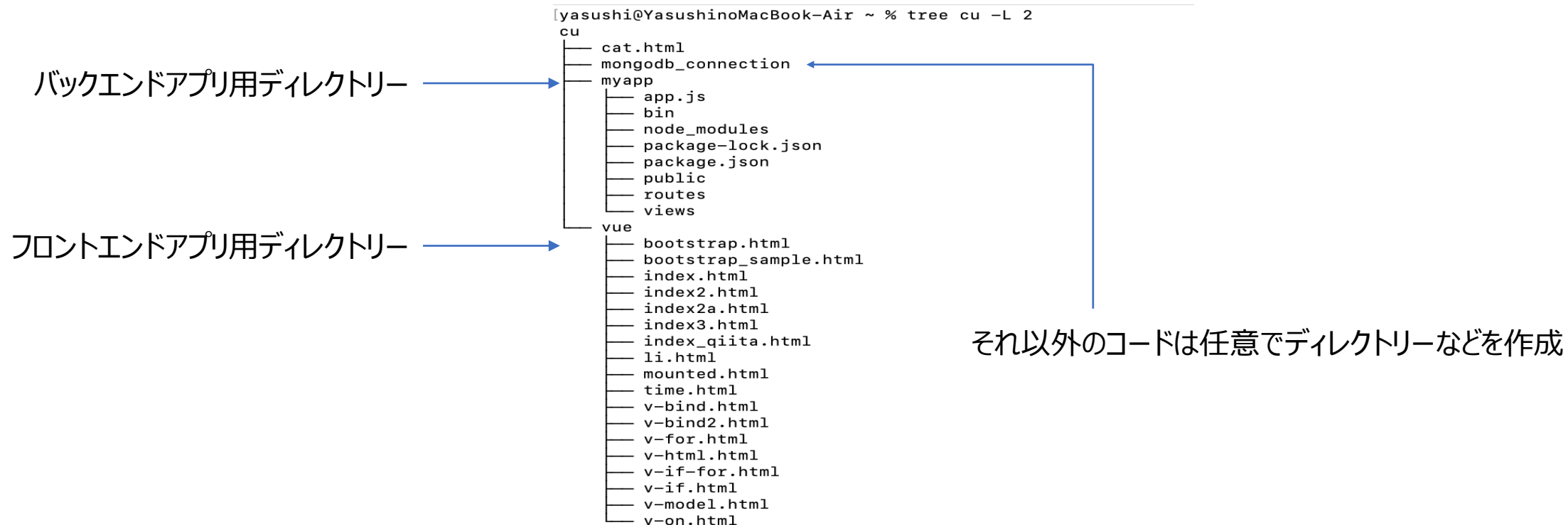
実機演習：事前準備

動画を全画面で視聴してください

実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリー構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。



Node.jsからMongoDB Atlasに接続

ターミナルから以下を実行。

内容：

作業用ディレクトリを作成

*第6回までのプログラムとは違うところに

作成したディレクトリに移動

npmの環境初期化

mongodbモジュールのインストール

ターミナル

```
% mkdir connect_mongodb
```

```
% cd connect_mongodb
```

```
% npm init -y
```

```
% npm install mongodb
```

```
[kokoji@MacBook-Pro-2 mongo % mkdir connect_mongodb  
[kokoji@MacBook-Pro-2 mongo % cd connect_mongodb  
[kokoji@MacBook-Pro-2 connect_mongodb % npm init -y  
Wrote to /Users/kokoji/tempk/mongo/connect_mongodb/package.json:
```

```
{  
  "name": "connect_mongodb",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

```
[kokoji@MacBook-Pro-2 connect_mongodb % npm install mongodb
```

```
added 16 packages, and audited 17 packages in 7s
```

```
found 0 vulnerabilities
```

```
[kokoji@MacBook-Pro-2 connect_mongodb %  
[kokoji@MacBook-Pro-2 connect_mongodb % ls  
node_modules          package-lock.json      package.json  
kokoji@MacBook-Pro-2 connect_mongodb % █
```

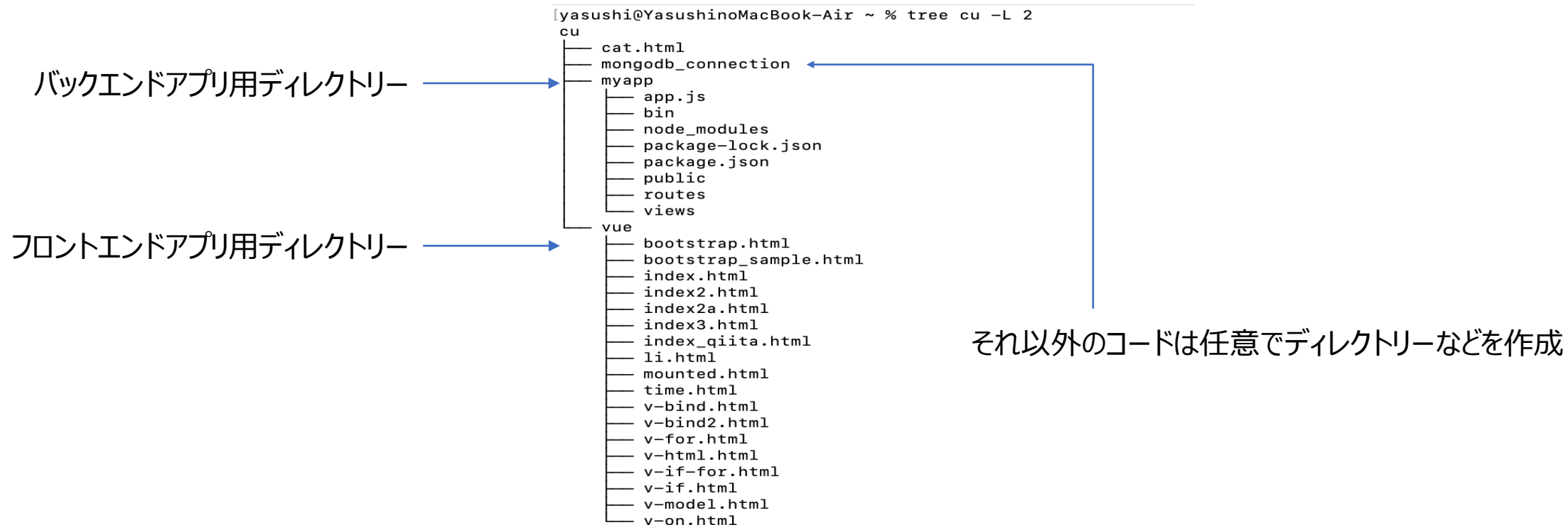
実機演習：Node.jsからMongoDB Atlasに接続

動画を全画面で視聴してください

実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリー構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。



プログラムの場所について

今回のプログラムは前回まで作成した場所（myapp）とは別のところに作成

（ご自分のホームディレクトリなど） - myapp - （前回までのプログラム）

|
-- connect_mongodb - （第三章のプログラム）

Node.jsからmongoDB実習

ここでは第6回で定義したREST APIで使したjsonデータを元に、Node.jsからmongoDBにアクセスしてCRUD操作を実施する（CRUDは、Create、Read、Update、Deleteの略）。

| CRUD操作 | メソッド | 説明 |
|--------|------------|------------------------|
| CREATE | insertMany | 複数のjsonノートデータをDBに登録する。 |
| READ | findOne | 登録したノートデータを1件参照する。 |
| UPDATE | replaceOne | 登録したノートデータを1件更新する。 |
| DELETE | deleteOne | 登録したノートデータを1件削除する。 |

複数のjsonノートデータをDBに登録する

- insertMany.jsを作成する。

ここをあらかじめ作成していた接続情報と入れ替える

```
const { MongoClient } = require("mongodb");

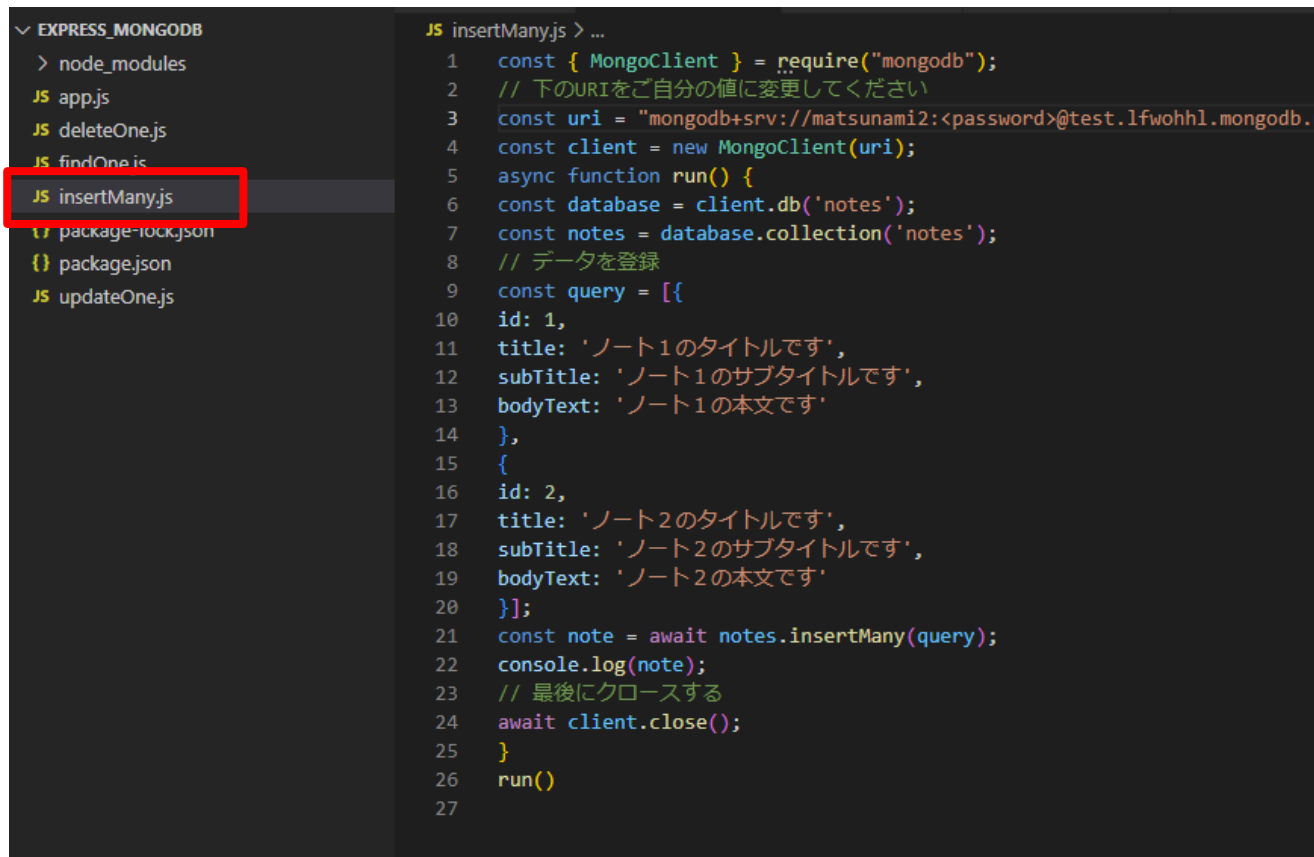
// 下のURIをご自分の値に変更してください
const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?retryWrites=true&w=majority";

const client = new MongoClient(uri);

async function run() {
  const database = client.db('notes');
  const notes = database.collection('notes');

  // データを登録
  const query = [{
    id: 1,
    title: 'ノート1のタイトルです',
    subTitle: 'ノート1のサブタイトルです',
    bodyText: 'ノート1の本文です'
  },
  {
    id: 2,
    title: 'ノート2のタイトルです',
    subTitle: 'ノート2のサブタイトルです',
    bodyText: 'ノート2の本文です'
  }
  ];
  const note = await notes.insertMany(query);
  console.log(note);
  // 最後にクローズする
  await client.close();
}
run()
```

insertMany.js



```
JS insertMany.js > ...
1  const { MongoClient } = require("mongodb");
2  // 下のURIをご自分の値に変更してください
3  const uri = "mongodb+srv://matsunami2:<password>@test.1fwohhl.mongodb.net/";
4  const client = new MongoClient(uri);
5  async function run() {
6    const database = client.db('notes');
7    const notes = database.collection('notes');
8    // データを登録
9    const query = [{
10     id: 1,
11     title: 'ノート1のタイトルです',
12     subTitle: 'ノート1のサブタイトルです',
13     bodyText: 'ノート1の本文です'
14   },
15   {
16     id: 2,
17     title: 'ノート2のタイトルです',
18     subTitle: 'ノート2のサブタイトルです',
19     bodyText: 'ノート2の本文です'
20   }
21   ];
22   const note = await notes.insertMany(query);
23   console.log(note);
24   // 最後にクローズする
25   await client.close();
26 }
27 run()
```


複数のjsonノードデータをDBに登録する

- insertMany.jsを実行する。ターミナルで以下を実行。

node insertMany.js

```
% node insertMany.js
```

実行結果

```
{
  acknowledged: true,
  insertedCount: 2,
  insertedIds: {
    '0': new ObjectId("6405b0addc45316dae25b13c"),
    '1': new ObjectId("6405b0addc45316dae25b13d")
  }
}
```

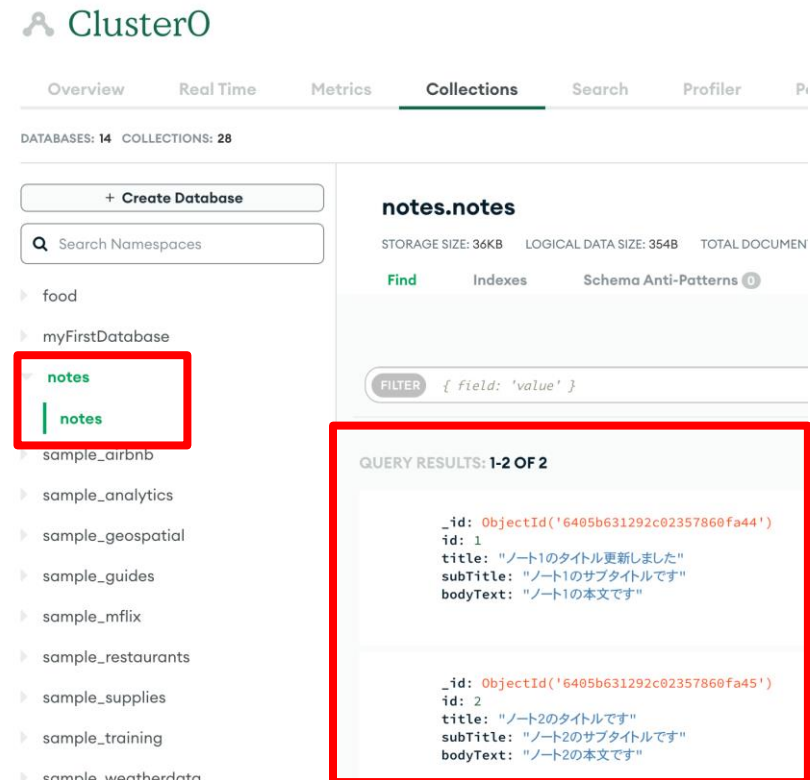
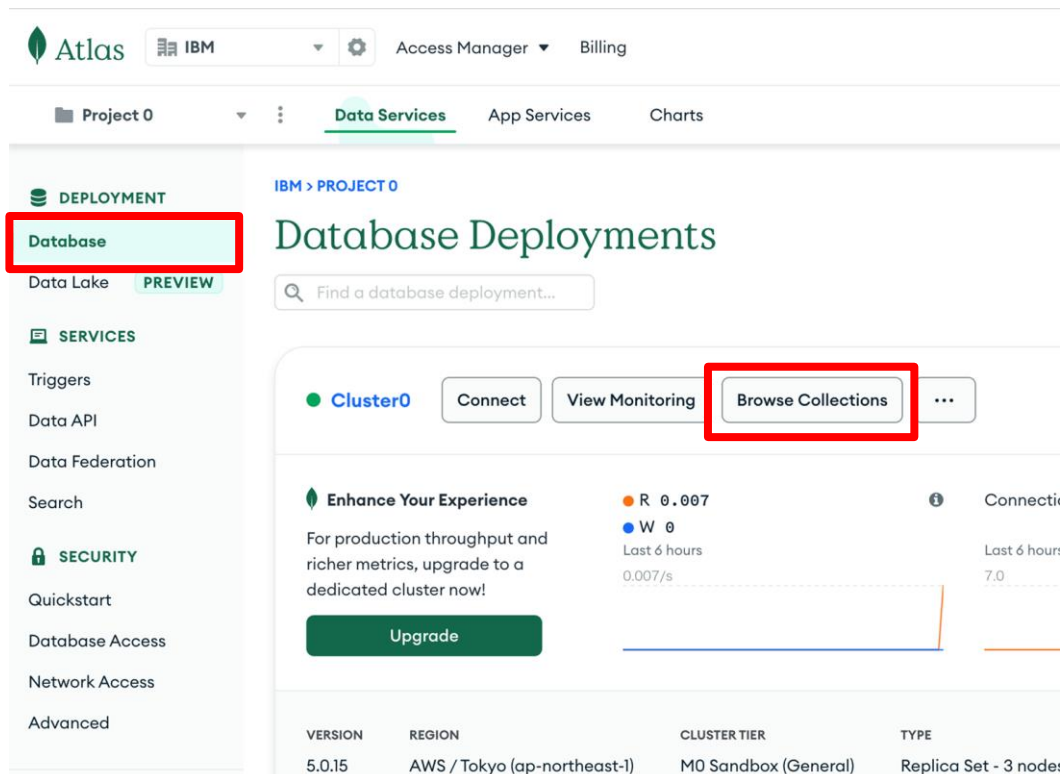
```
[kokoji@MacBook-Pro-2 mongodbaccess % node insertMany.js
{
  acknowledged: true,
  insertedCount: 2,
  insertedIds: {
    '0': new ObjectId("6405b0addc45316dae25b13c"),
    '1': new ObjectId("6405b0addc45316dae25b13d")
  }
}
```

複数のjsonノートデータをDBに登録する

- mongoDB Atlasのサイトで登録結果を確認する。

Atlasのサイトにアクセスし、Databaseを開き、「Browse Collections」をクリック

「notes」 / 「notes」をクリックすると、前ページで登録したjsonデータが表示される。



登録したノートデータを1件参照する

- findOne.jsを作成する。

ここをあらかじめ作成していた接続情報と入れ替える

```
const { MongoClient } = require("mongodb");

// 下のURIをご自分の値に変更してください
const uri =
  "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?retryWrites=true&w=majority";

const client = new MongoClient(uri);

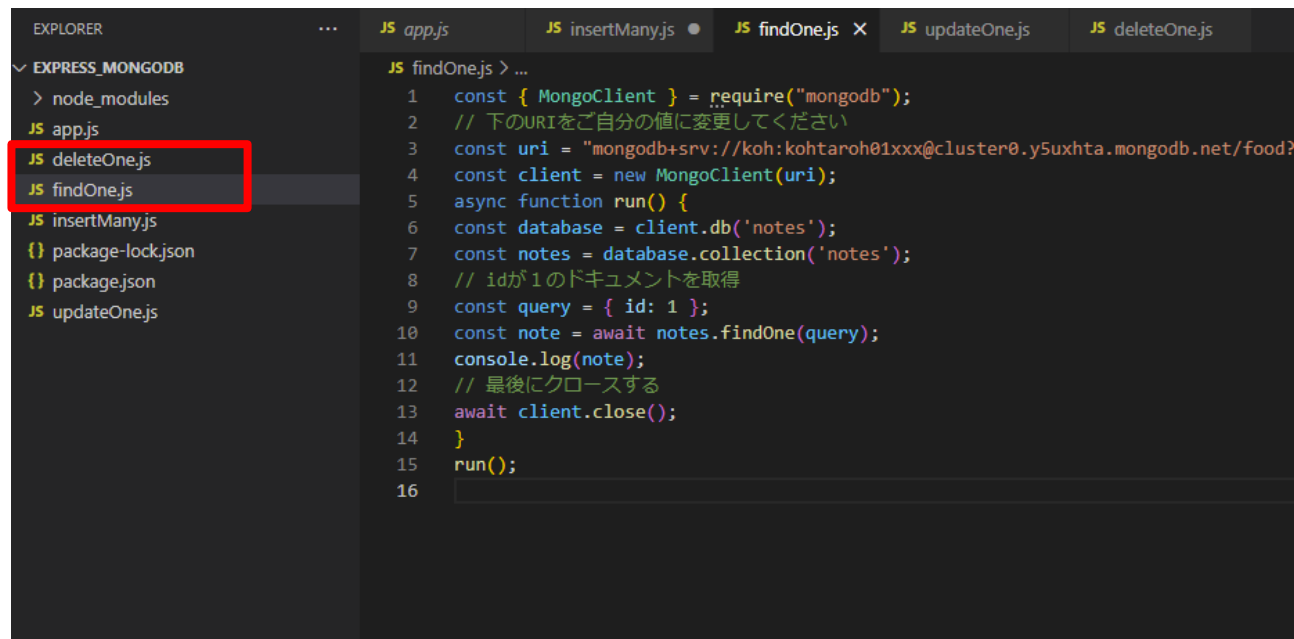
async function run() {
  const database = client.db('notes');
  const notes = database.collection('notes');

  // idが1のドキュメントを取得
  const query = { id: 1 };
  const note = await notes.findOne(query);

  console.log(note);
  // 最後にクローズする
  await client.close();
}

run();
```

findOne.js



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the file structure of a project named 'EXPRESS_MONGODB'. The files listed are 'app.js', 'deleteOne.js', 'findOne.js', 'insertMany.js', 'package-lock.json', 'package.json', and 'updateOne.js'. The 'findOne.js' file is selected and highlighted with a red rectangle. On the right, the Editor pane shows the code for 'findOne.js'. The code is as follows:

```
JS findOne.js > ...
1  const { MongoClient } = require("mongodb");
2  // 下のURIをご自分の値に変更してください
3  const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?r
4  const client = new MongoClient(uri);
5  async function run() {
6  const database = client.db('notes');
7  const notes = database.collection('notes');
8  // idが1のドキュメントを取得
9  const query = { id: 1 };
10 const note = await notes.findOne(query);
11 console.log(note);
12 // 最後にクローズする
13 await client.close();
14 }
15 run();
16
```

登録したノートデータを1件参照する

- findOne.jsを実行する。ターミナルで以下を実行。

node findOne.js

```
% node findOne.js
```

実行結果

```
{
  _id: new ObjectId("6405b0addc45316dae25b13c"),
  id: 1,
  title: 'ノート1のタイトルです',
  subTitle: 'ノート1のサブタイトルです',
  bodyText: 'ノート1の本文です'
}
```

```
kokoji@MacBook-Pro-2 mongodbaccess % node findOne.js
{
  _id: new ObjectId("6405b0addc45316dae25b13c"),
  id: 1,
  title: 'ノート1のタイトルです',
  subTitle: 'ノート1のサブタイトルです',
  bodyText: 'ノート1の本文です'
}
```

登録したノートデータを1件更新する

- updateOne.jsを作成する。

ここをあらかじめ作成していた接続情報と入れ替える

```
const { MongoClient } = require("mongodb");

// 下のURIをご自分の値に変更してください
const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?retryWrites=true&w=majority";

const client = new MongoClient(uri);

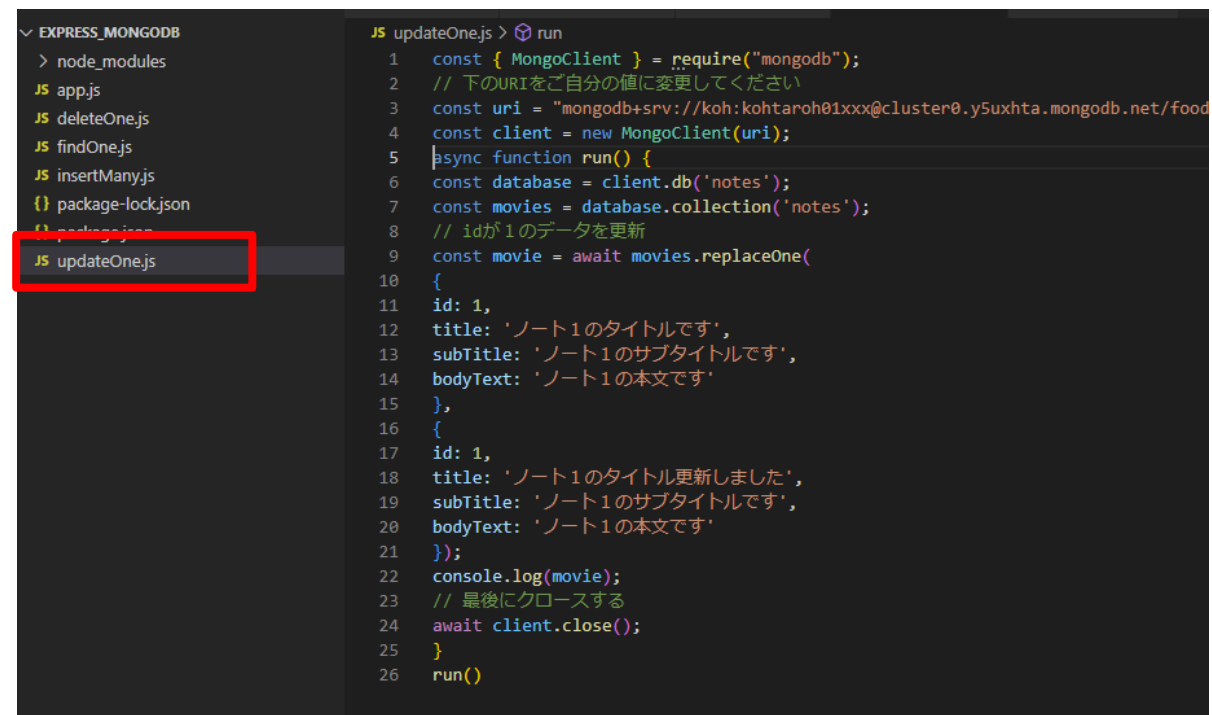
async function run() {
  const database = client.db('notes');
  const movies = database.collection('notes');

  // idが1のデータを更新
  const movie = await movies.replaceOne(
    {
      id: 1,
      title: 'ノート1のタイトルです',
      subTitle: 'ノート1のサブタイトルです',
      bodyText: 'ノート1の本文です'
    },
    {
      id: 1,
      title: 'ノート1のタイトル更新しました',
      subTitle: 'ノート1のサブタイトルです',
      bodyText: 'ノート1の本文です'
    }
  );

  console.log(movie);
  // 最後にクローズする
  await client.close();
}

run();
```

updateOne.js



```
JS updateOne.js > run
1  const { MongoClient } = require("mongodb");
2  // 下のURIをご自分の値に変更してください
3  const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food";
4  const client = new MongoClient(uri);
5  async function run() {
6    const database = client.db('notes');
7    const movies = database.collection('notes');
8    // idが1のデータを更新
9    const movie = await movies.replaceOne(
10     {
11       id: 1,
12       title: 'ノート1のタイトルです',
13       subTitle: 'ノート1のサブタイトルです',
14       bodyText: 'ノート1の本文です'
15     },
16     {
17       id: 1,
18       title: 'ノート1のタイトル更新しました',
19       subTitle: 'ノート1のサブタイトルです',
20       bodyText: 'ノート1の本文です'
21     }
22   );
23   console.log(movie);
24   // 最後にクローズする
25   await client.close();
26 }
27 run();
```

登録したノートデータを1件更新する

- updateOne.jsを実行する。ターミナルで以下を実行。

node updateOne.js

```
% node updateOne.js
```

実行結果

```
{
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
```

```
[kokoji@MacBook-Pro-2 mongodbaccess % node updateOne.js
{
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
[kokoji@MacBook-Pro-2 mongodbaccess % node findOne.js
{
  _id: new ObjectId("6405b631292c02357860fa44"),
  id: 1,
  title: 'ノート1のタイトル更新しました',
  subTitle: 'ノート1のサブタイトルです',
  bodyText: 'ノート1の本文です'
}
```

登録したノートデータを1件削除する

- deleteOne.jsを作成する。

ここをあらかじめ作成していた接続情報と入れ替える

```
const { MongoClient } = require("mongodb");

// 下のURIをご自分の値に変更してください
const uri =
  "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?retryWrites=true&w=majority";

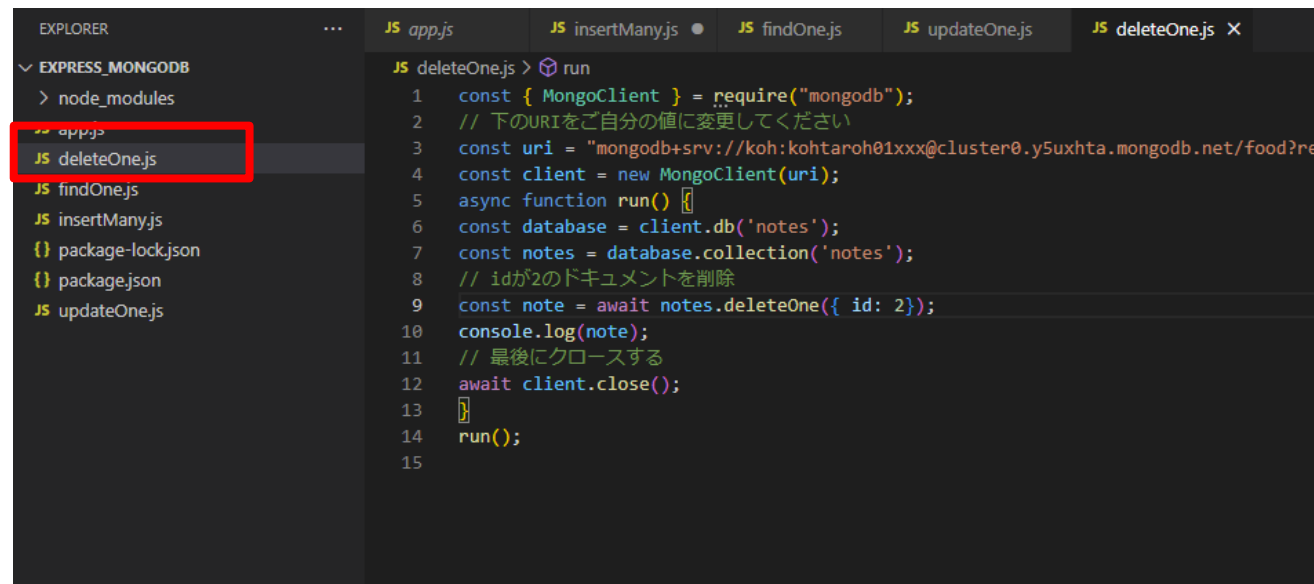
const client = new MongoClient(uri);

async function run() {
  const database = client.db('notes');
  const notes = database.collection('notes');

  // idが2のドキュメントを削除
  const note = await notes.deleteOne({ id: 2 });
  console.log(note);
  // 最後にクローズする
  await client.close();
}

run();
```

deleteOne.js



The screenshot shows the VS Code interface. On the left, the 'EXPLORER' sidebar displays the file structure of a project named 'EXPRESS_MONGODB'. The files listed are 'node_modules', 'app.js', 'deleteOne.js', 'findOne.js', 'insertMany.js', 'package-lock.json', 'package.json', and 'updateOne.js'. The 'deleteOne.js' file is highlighted with a red rectangle. On the right, the editor window shows the code for 'deleteOne.js'. The code is as follows:

```
1 const { MongoClient } = require("mongodb");
2 // 下のURIをご自分の値に変更してください
3 const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?re
4 const client = new MongoClient(uri);
5 async function run() {
6   const database = client.db('notes');
7   const notes = database.collection('notes');
8   // idが2のドキュメントを削除
9   const note = await notes.deleteOne({ id: 2 });
10  console.log(note);
11  // 最後にクローズする
12  await client.close();
13 }
14 run();
15
```

登録したノートデータを 1 件削除する

- deleteOne.jsを実行する。ターミナルで以下を実行。

node deleteOne.js

% node deleteOne.js

実行結果

{ acknowledged: true, deletedCount: 1 }

ターミナルの表示例

```
kokoji@MacBook-Pro-2 mongodbaccess % node deleteOne.js  
{ acknowledged: true, deletedCount: 1 }
```

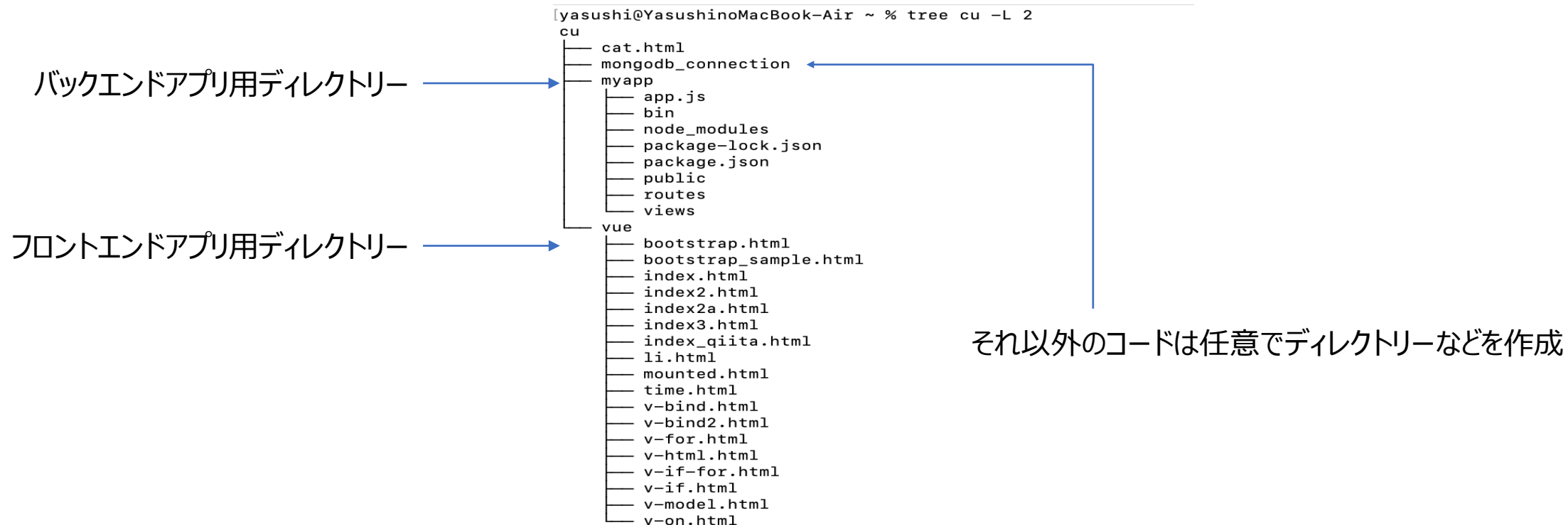

実機演習：mongoDB データベースの編集

動画を全画面で視聴してください

実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリー構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。



第3章 まとめ

- Node.jsからmongodbにアクセスするプログラムを作成し、実行した。
- Node.jsからmongodbにアクセスし、データの登録、参照、更新、削除を実施した。

第7回 まとめ

- データベースの概念を学習した
- ドキュメントデータベース Mongo DBについて学習した
- Node.jsからMongoDBへの接続プログラムの作成、実行を行った。
- Node.jsからMongoDBへのアクセスプログラムを作成、データの登録、参照、更新、削除を実施した。

JavaScriptフレームワークによるWebプログラミング
第7回 データベース実習

第3章

Node.jsからmongodbへの接続

終わり