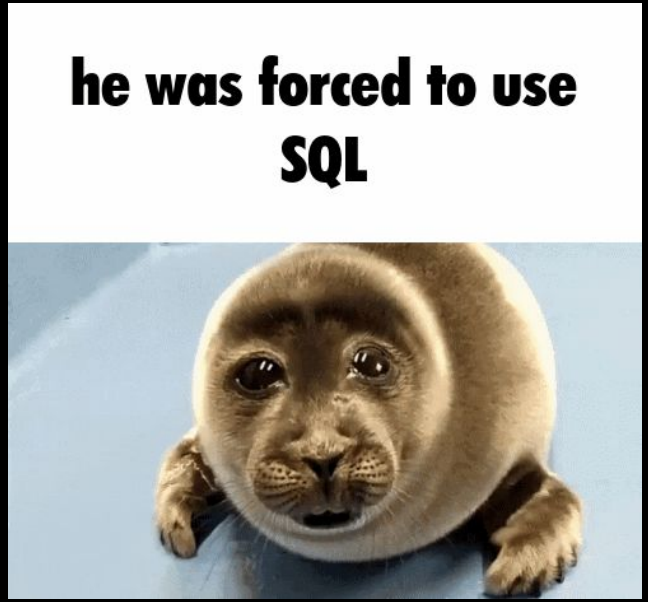
React

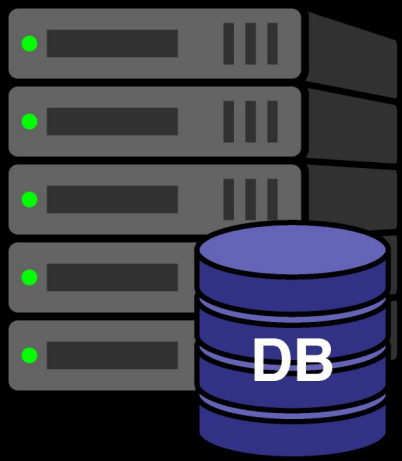
15. Introduction to SQL

1. What is a DB (Database)
2. Introduction to SQL DB
3. Introduction to NoSQL DB
4. SQL vs NoSQL
5. Installing MySQL
6. Connecting App to DB
7. Creating homes Table
8. Querying homes in App
9. Adding DB in Models
10. Adding Home in Model
11. Implementing Model using Where

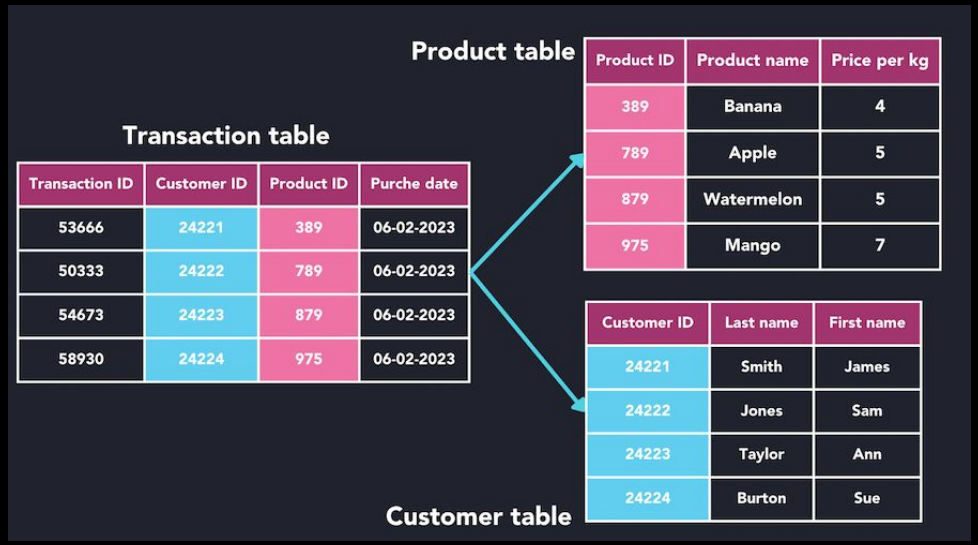


15.1 What is a DB (Database)

1. Store Data: Keep large amounts of data in a structured format.
2. Enable Data Management: Allow for adding, updating, and deleting data easily.
3. Facilitate Quick Access: Provide fast retrieval of data through queries.
4. Ensure Data Integrity: Maintain accuracy and consistency of data over time.
5. Support Multiple Users: Handle concurrent access by many users simultaneously.
6. Secure Data: Protect information through access controls and authentication.



15.2 Introduction to SQL DB



- Vertical Scalability: Typically scaled by increasing the resources of a single server (scaling up).
- Relationships: Tables can have multiple types of relationships.

- Relational Model: Organize data into tables with rows and columns.
- Fixed Schema: Require a predefined schema; the structure of the data must be known in advance.

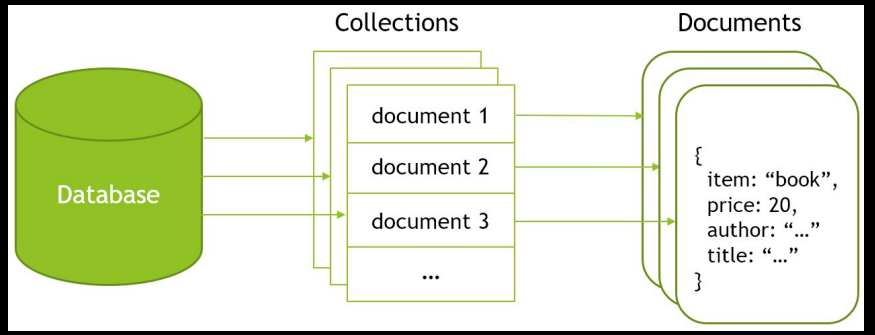
15.2 Introduction to SQL DB

MySQL™

- Relational Model Use of SQL: Utilize SQL for querying and managing data, which is a standardized and widely-used language.
- ACID Compliance: Support transactions that are Atomic, Consistent, Isolated, and Durable.
- Complex Queries: Excel at handling complex queries and relationships between data.

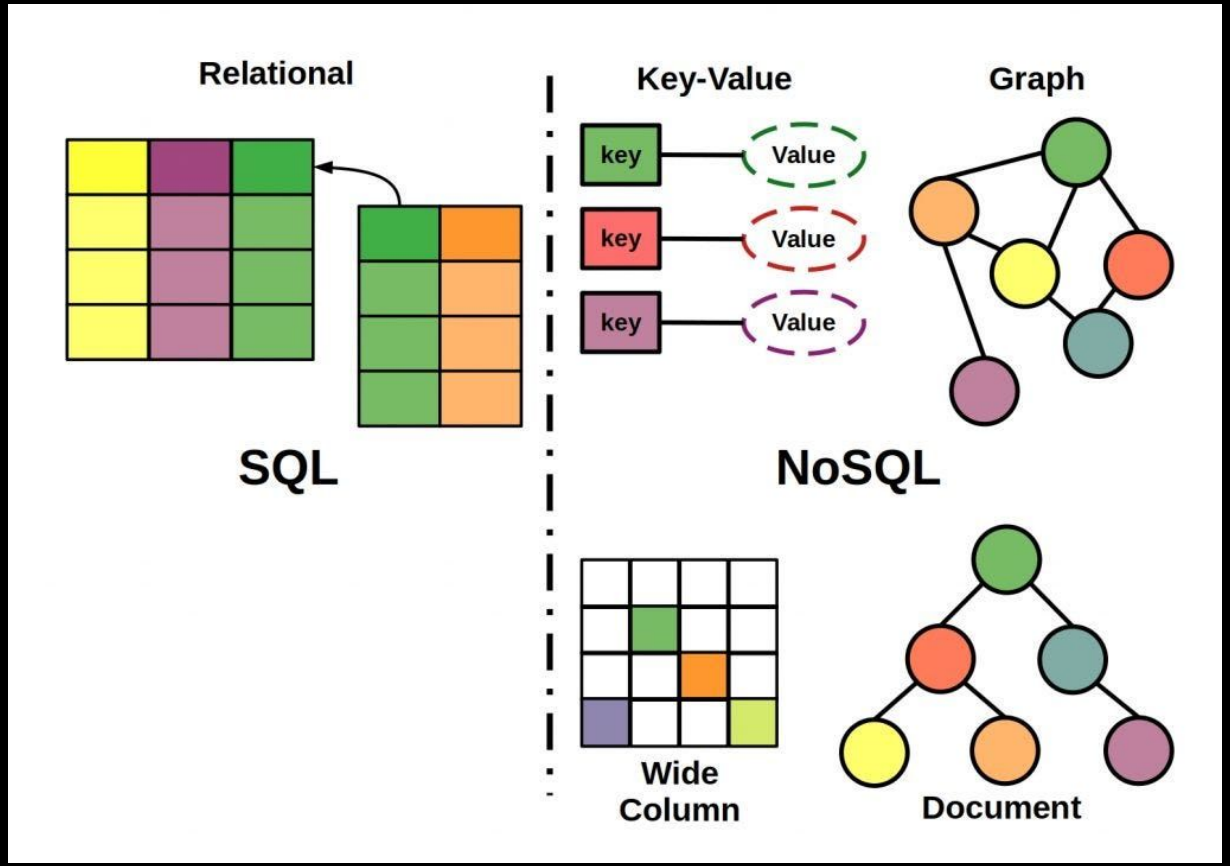
15.3 Introduction to NoSQL DB

- Flexible Schema: Allow for dynamic schemas, accommodating unstructured or semi-structured data without predefined structures.
- Duplicacy over Relations: Duplicates data across records (denormalization) to enhance performance and scalability, rather than relying on complex relationships and joins as in relational databases.
- Horizontal Scalability: Designed to scale out by adding more servers, handling large volumes of data efficiently.
- Performance: Optimized for high throughput and low latency, suitable for real-time applications.



mongoDB.

15.4 SQL vs NoSQL

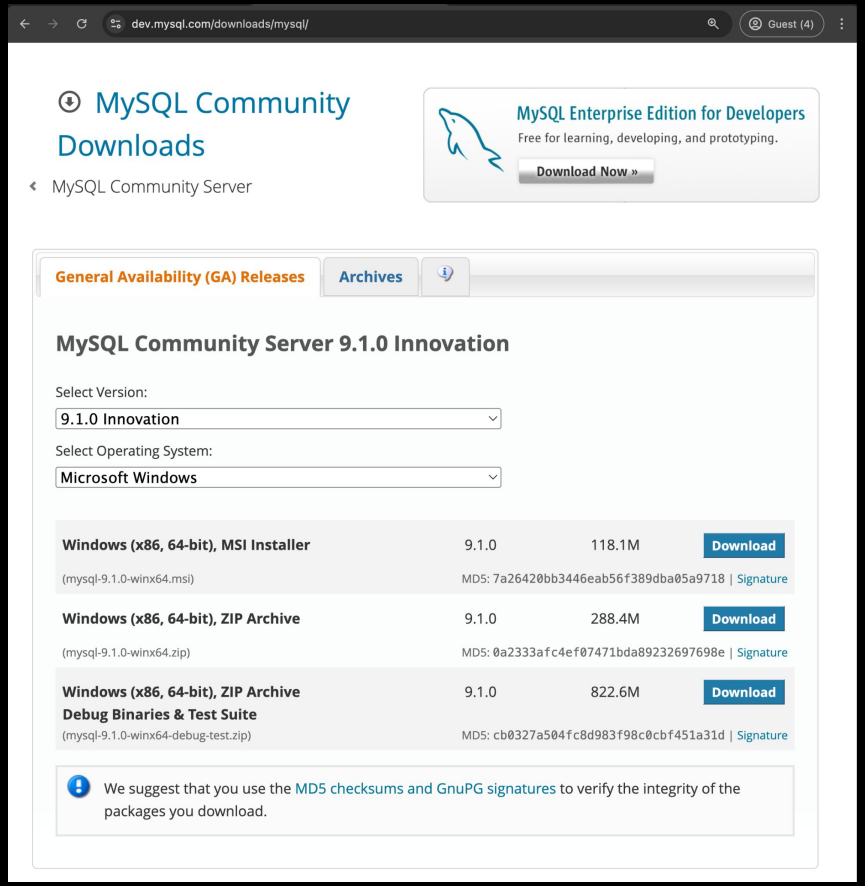


15.4 SQL vs NoSQL

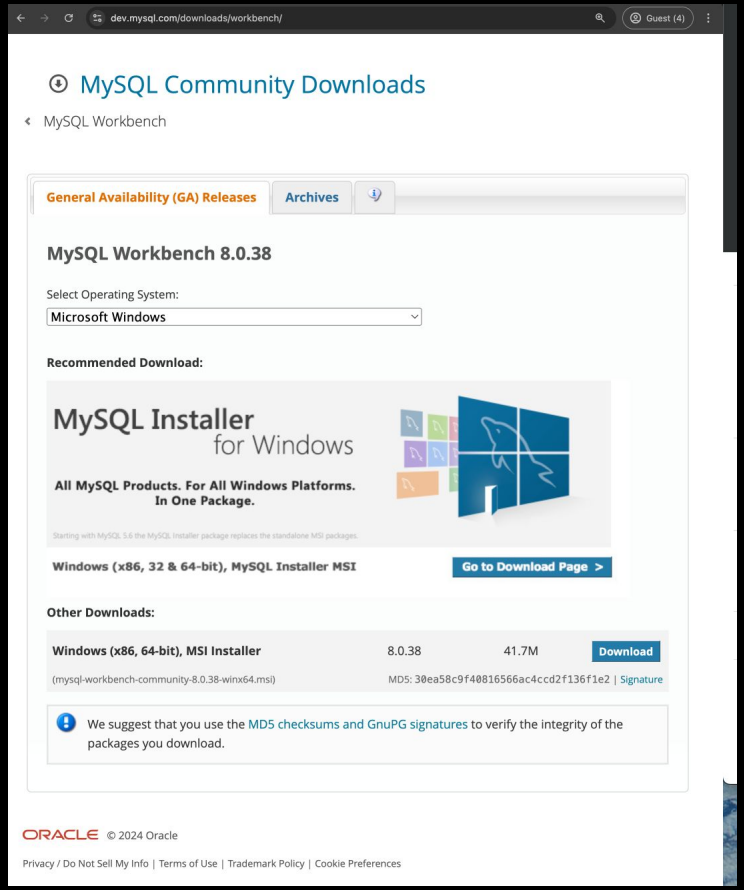
Feature	SQL Databases	NoSQL Databases
Data Model	Relational (tables with rows and columns)	Non-relational (document, key-value, graph, etc.)
Schema	Fixed schema (predefined structure)	Flexible schema (dynamic structure)
Scalability	Vertically scalable (scale up)	Horizontally scalable (scale out)
Query Language	SQL (Structured Query Language)	Various query languages and APIs
ACID Compliance	Strong ACID compliance	Varies; often prioritizes performance over ACID
Use Cases	Structured data and complex queries	Unstructured data and real-time applications

15.5 Installing MySQL

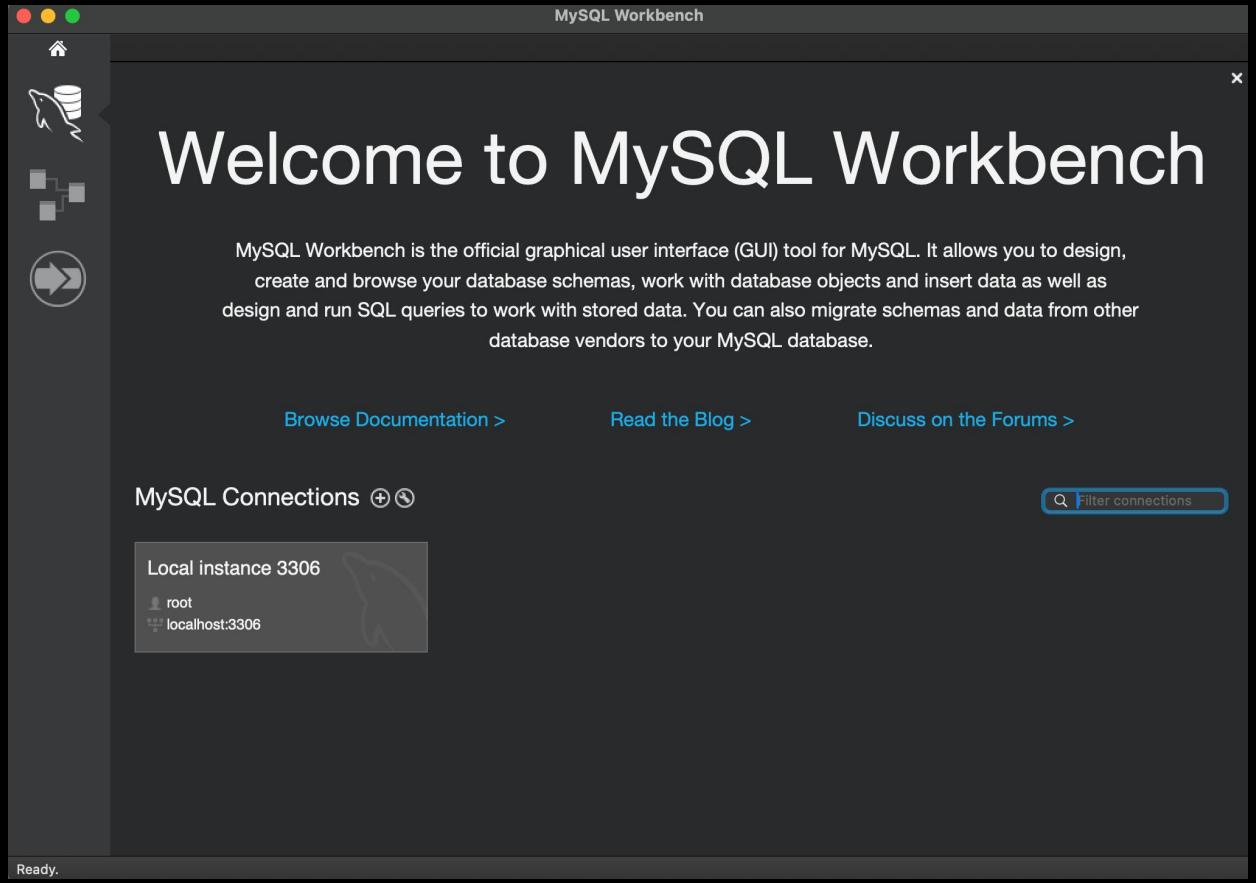
MySQL™



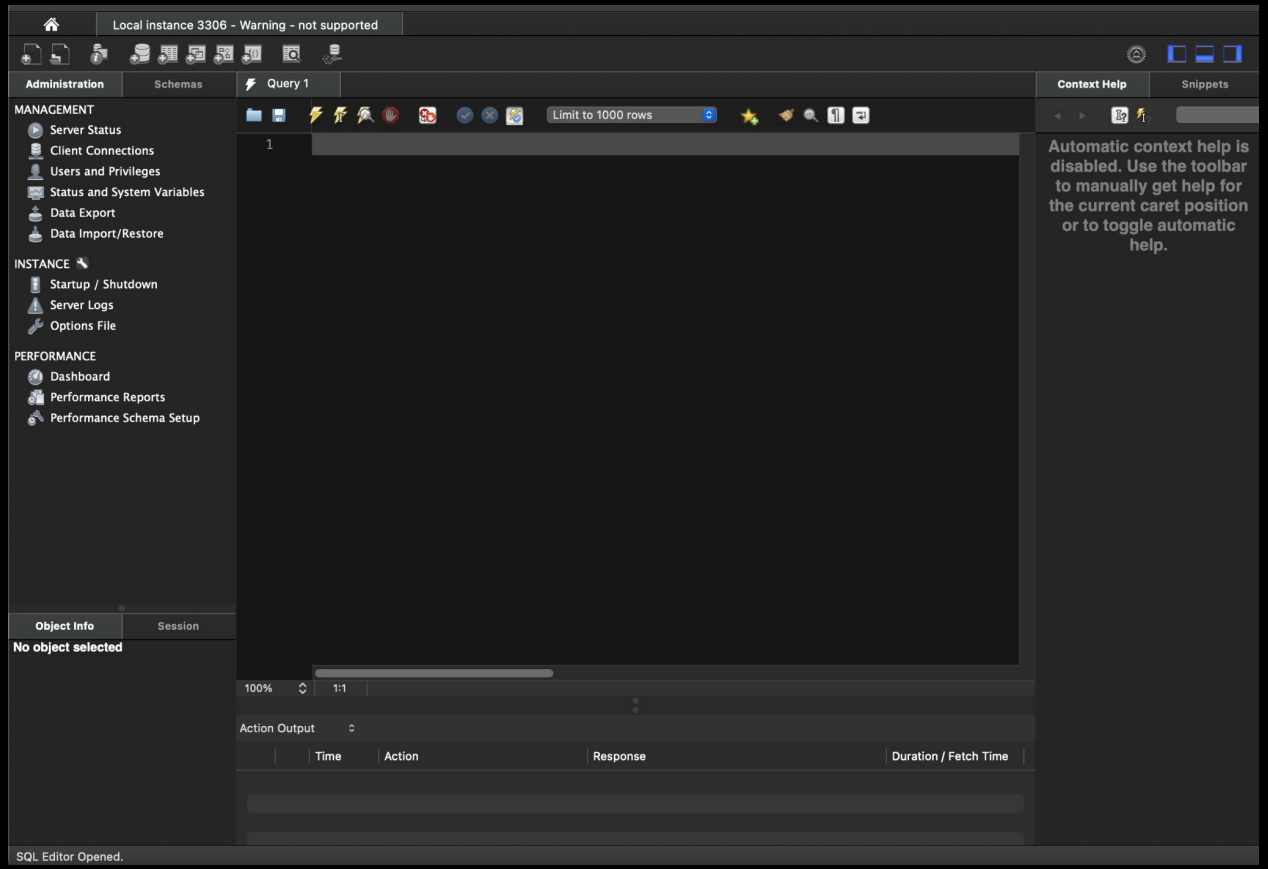
15.5 Installing MySQL



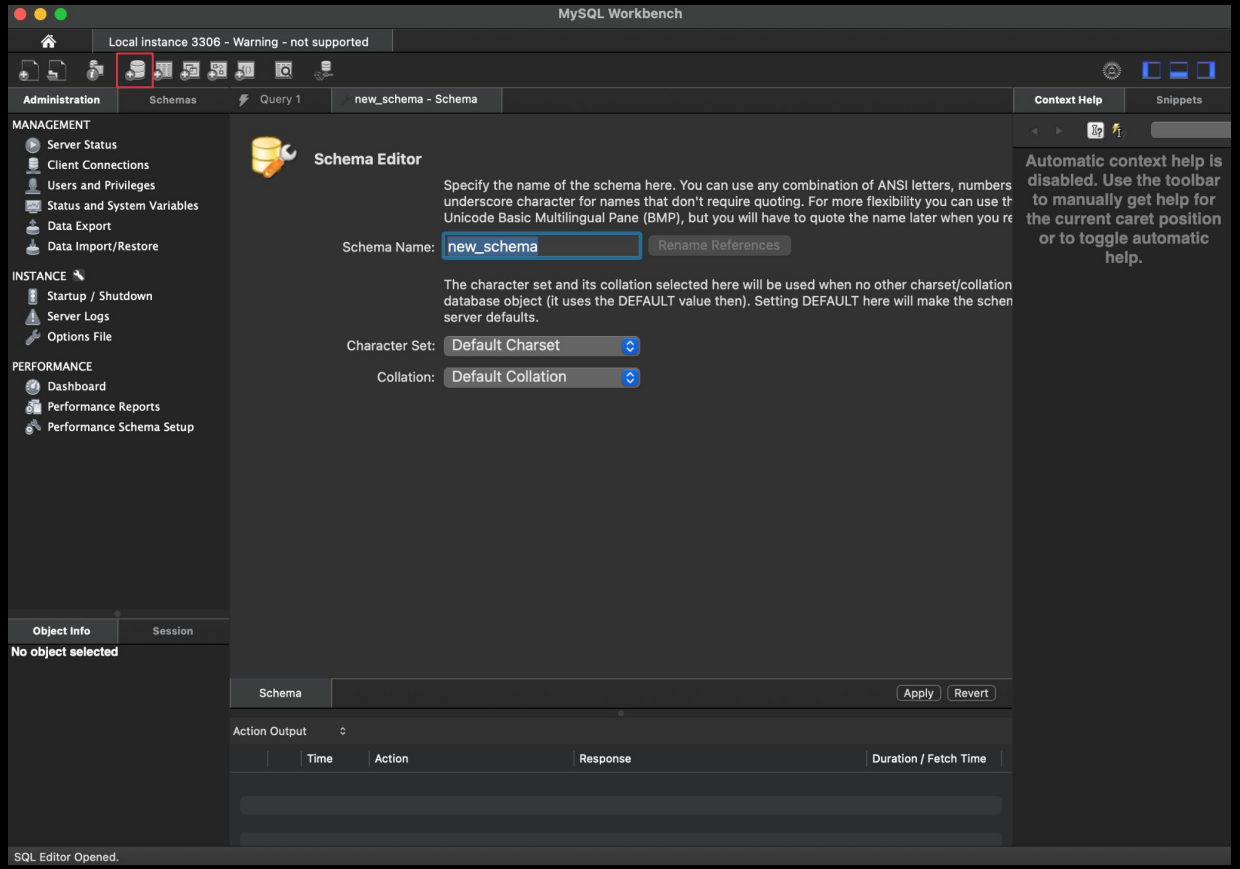
15.5 Installing MySQL



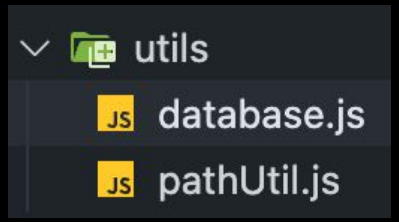
15.5 Installing MySQL



15.5 Installing MySQL



15.6 Connecting App to DB




```
prashantjain@Prashants-Mac-mini Chapter 13 - MVC % npm install --save mysql2

added 12 packages, and audited 222 packages in 586ms

49 packages are looking for funding
  run `npm fund` for details
```

database.js

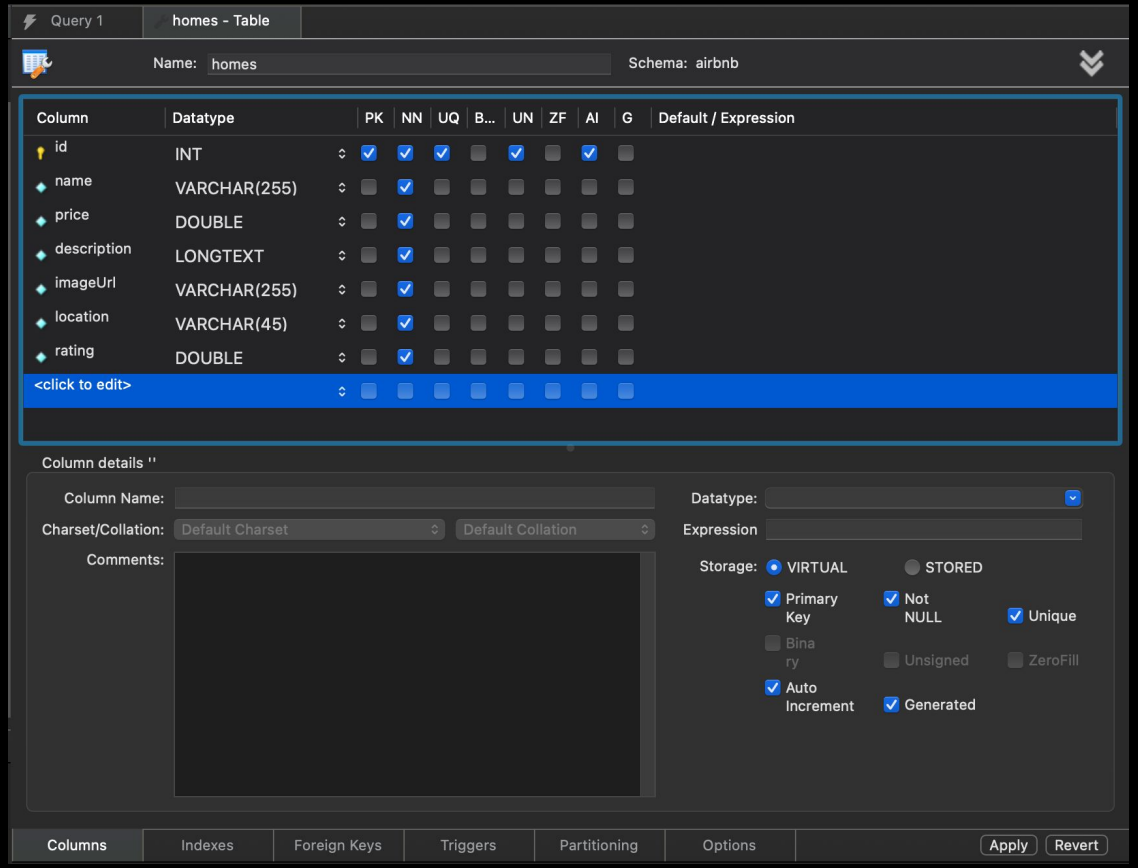
utils > database.js > ...

```js
const mysql = require("mysql2");

const pool = mysql.createPool({
  host: "localhost",
  user: "root",
  password: "CompleteCoding@01",
  database: "airbnb",
});

module.exports = pool.promise();
```

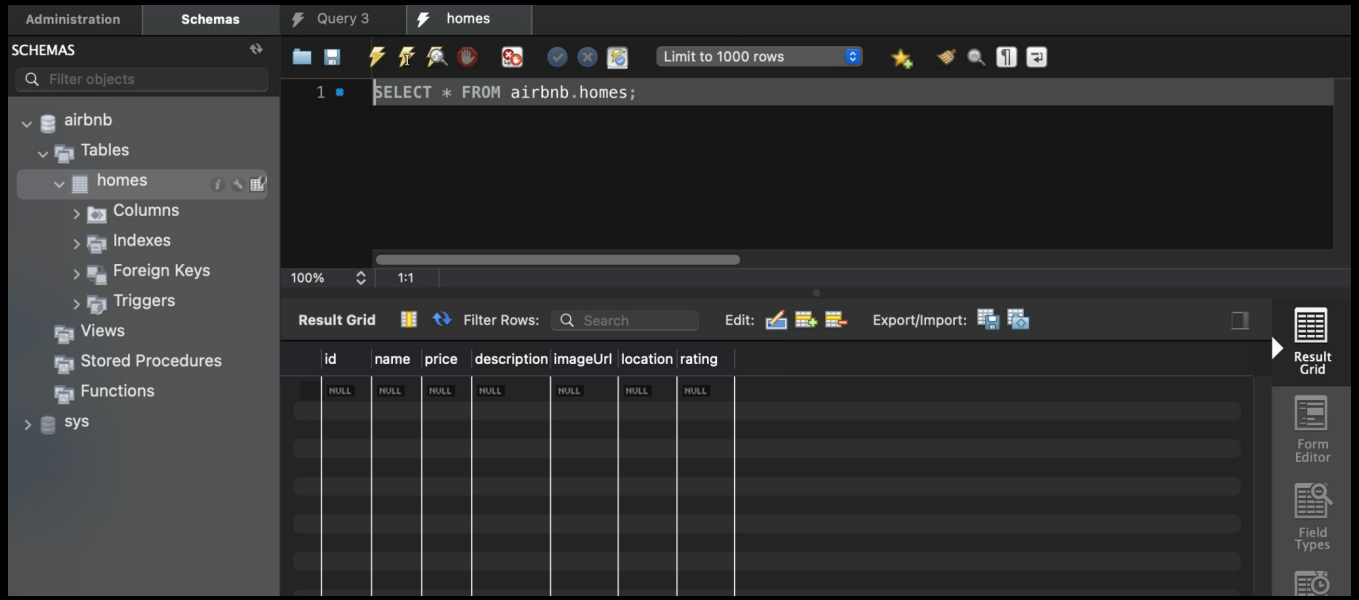
15.7 Creating homes Table

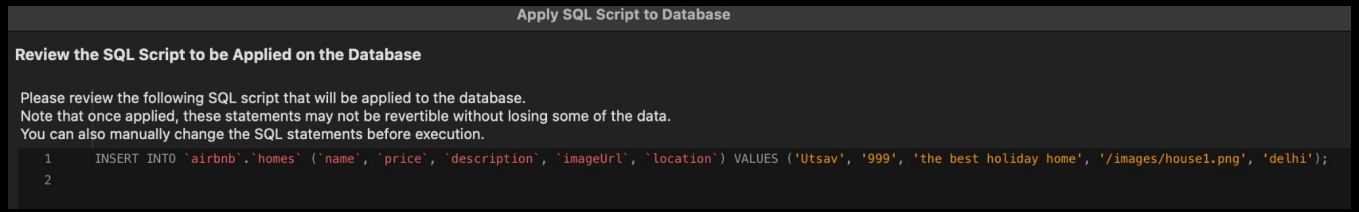


15.7 Creating homes Table

```sql
CREATE TABLE `airbnb`.`homes` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `price` DOUBLE NOT NULL,
  `description` LONGTEXT NOT NULL,
  `imageUrl` VARCHAR(255) NOT NULL,
  `location` VARCHAR(45) NOT NULL,
  `rating` DOUBLE NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE);
```

15.7 Creating homes Table





15.8 Querying homes in App

```
const db = require("./utils/database");

db.execute("SELECT * FROM homes").then(([rows, fields]) => {
  console.log(rows);
  console.log(fields);
}).catch((error) => {
  console.log("Error Fetching Homes", error);
});
```

```
Server running on address http://localhost:3000
[
  {
    id: 1,
    name: 'Utsav',
    price: 999,
    description: 'the best holiday home',
    imageUrl: '/images/house1.png',
    location: 'delhi',
    rating: 4.5
  }
]
[
  `id` INT UNSIGNED NOT NULL PRIMARY KEY UNIQUE_KEY AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `price` DOUBLE NOT NULL,
  `description` LONGTEXT NOT NULL,
  `imageUrl` VARCHAR(255) NOT NULL,
  `location` VARCHAR(45) NOT NULL,
  `rating` DOUBLE NOT NULL
]
```

15.9 Adding DB in Models

1. Remove the test code from app.js
2. Change the Home.js model file to remove all code related to file operations.
3. Import the DB from the utils.
4. Change photoUrl to imageUrl and houseName to name in the entire project.
5. Implement fetchAll:
    a. Using the query we used while testing.
    b. fetchAll will not take a callback but return a promise.
6. Go to StoreController and use the promise to get the data here.
7. Fix all the usages of fetchAll.

15.9 Adding DB in Models

2,3.

```
const db = require("../utils/database");
const Favourites = require("./favourites");

module.exports = class Home {
  constructor(houseName, price, location, rating, photoUrl) {
    this.houseName = houseName;
    this.price = price;
    this.location = location;
    this.rating = rating;
    this.photoUrl = photoUrl;
  }

  save() { ⇥ tab
  }

  static fetchAll() {
  }

  static findById(id) {
  }

  static deleteById(id) {
  }
};
```

15.9 Adding DB in Models

5.
```
static fetchAll() {
  return db.execute("SELECT * FROM homes");
}
```

6.
```
exports.getHomes = (req, res, next) => {
  Home.fetchAll()
    .then(([rows, fields]) => {
      res.render("store/home-list", {
        registeredHomes: rows,
        pageTitle: "Homes List",
        currentPage: "Home",
      });
    })
    .catch((error) => {
      console.log("Error Fetching Homes", error);
    });
};
```

15.9 Adding DB in Models

7.

```
exports.getIndex = (req, res, next) => {
  Home.fetchAll()
  .then(([rows, fields]) => {
    res.render("store/index", {
      registeredHomes: rows,
      pageTitle: "airbnb Home",
      currentPage: "index",
    })
  })
  .catch((error) => {
    console.log("Error Fetching Homes", error);
  });
};
```

```
exports.getHostHomes = (req, res, next) => {
  Home.fetchAll().then(([rows, fields]) => {
    res.render("host/host-home-list", {
      registeredHomes: rows,
      pageTitle: "Host Homes",
      currentPage: "hostHomes",
    });
  }).catch((error) => {
    console.log("Error Fetching Homes", error);
  });
};
```

```
exports.getFavouriteList = (req, res, next) => {
  Favourites.getFavourites((favourites) => {
    Home.fetchAll().then(([registeredHomes, fields]) => {
      const favouritesWithDetails = favourites.map((homeId) =>
        registeredHomes.find((home) => home.id === homeId)
      );
      res.render("store/favourite-list", {
        favourites: favouritesWithDetails,
        pageTitle: "My Favourites",
        currentPage: "favourites",
      });
    }).catch((error) => {
      console.log("Error Fetching Homes", error);
    });
  });
};
```

15.10 Adding Home in Model

1. Add the description field in home. Change constructor and usage.
2. Make changes in UI to input and show it everywhere.
3. Implement the save method using the insert query.
4. Change the usages of save method to use the promise.

15.10 Adding Home in Model

1.

```
module.exports = class Home {
  constructor(name, description, price, location, rating, imageUrl) {
    this.name = name;
    this.description = description;
    this.price = price;
    this.location = location;
    this.rating = rating;
    this.imageUrl = imageUrl;
  }
```

```
exports.postAddHome = (req, res, next) => {
  const { name, description, price, location, rating, imageUrl } = req.body;
  const home = new Home(name, description, price, location, rating, imageUrl);
```

```
exports.postEditHome = (req, res, next) => {
  const { id, name, description, price, location, rating, imageUrl } = req.body;
  const home = new Home(name, description, price, location, rating, imageUrl);
```

15.10 Adding Home in Model

2.

```
<input
  type="text"
  name="description"
  value="<%= home ? home.description : '' %>"
  placeholder="Enter your House Description"
  class="w-full px-4 py-2 mb-4 border rounded-md focus:outline-none focus:ring-2
  focus:ring-red-500"/>

<div class="border-b pb-4">
    <h3 class="text-2xl font-semibold mb-2">Description</h3>
    <p class="text-gray-600"><%= home.description %></p>
</div>

<div class="border-b pb-4">
    <h3 class="text-2xl font-semibold mb-2">Location</h3>
    <p class="text-gray-600"><%= home.location %></p>
</div>
```

15.10 Adding Home in Model

3.

```
save() {
  return db.execute(
    "INSERT INTO homes (name, price, location, rating, imageUrl) VALUES (?, ?, ?, ?, ?)",
    [this.name, this.price, this.location, this.rating, this.imageUrl]
  );
}
```

4.

```
exports.postAddHome = (req, res, next) => {
  const { name, description, price, location, rating, imageUrl } = req.body;
  const home = new Home(name, description, price, location, rating, imageUrl);
  home.save().then(() => {
    res.render("host/home-added", {
      pageTitle: "Home Added Successfully",
      currentPage: "homeAdded",
    });
  }).catch((error) => {
    console.log("Error Adding Home", error);
  });
};

exports.postEditHome = (req, res, next) => {
  const { id, name, description, price, location, rating, imageUrl } = req.body;
  const home = new Home(name, description, price, location, rating, imageUrl);
  home.id = id;
  home.save().then(() => {
    res.redirect("/host/host-home-list");
  }).catch((error) => {
    console.log("Error Editing Home", error);
  });
};
```

15.11 Implementing Model using Where

```
static findById(id) {
  return db.execute("SELECT * FROM homes WHERE id = ?", [id]);
}

static deleteById(id) {
  return db.execute("DELETE FROM homes WHERE id = ?", [id]);
}
```

```
exports.postDeleteHome = (req, res, next) => {
  const homeId = req.params.homeId;
  Home.deleteById(homeId).then(() => {
    res.redirect("/host/host-home-list");
  }).catch((error) => {
    console.log("Error Deleting Home", error);
  });
};
```

```
exports.getHome = (req, res, next) => {
  const homeId = req.params.homeId;
  Home.findById(homeId).then(([rows]) => {
    const home = rows[0];
    if (!home) {
      return res.redirect("/homes");
    }
    res.render("store/home-detail", {
      home: home,
      pageTitle: home.name,
      currentPage: "homes",
    });
  }).catch((error) => {
    console.log("Error Fetching Home", error);
  });
};
```