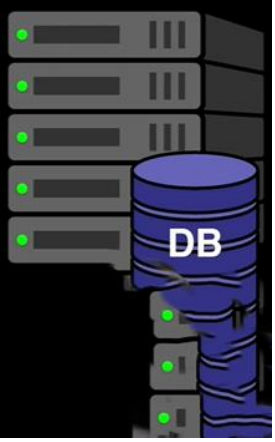
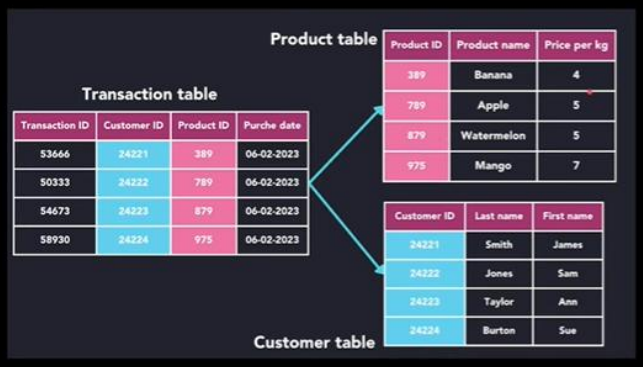
DataBase :

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

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

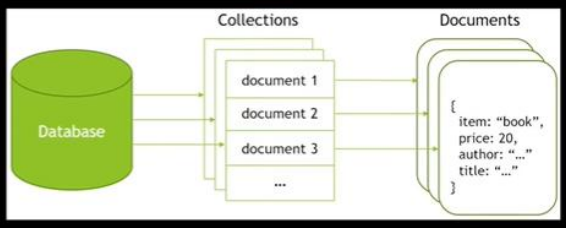
```
SELECT age, country
FROM Customers
WHERE country = 'USA';
```

age	country
31	USA
22	USA

- 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.

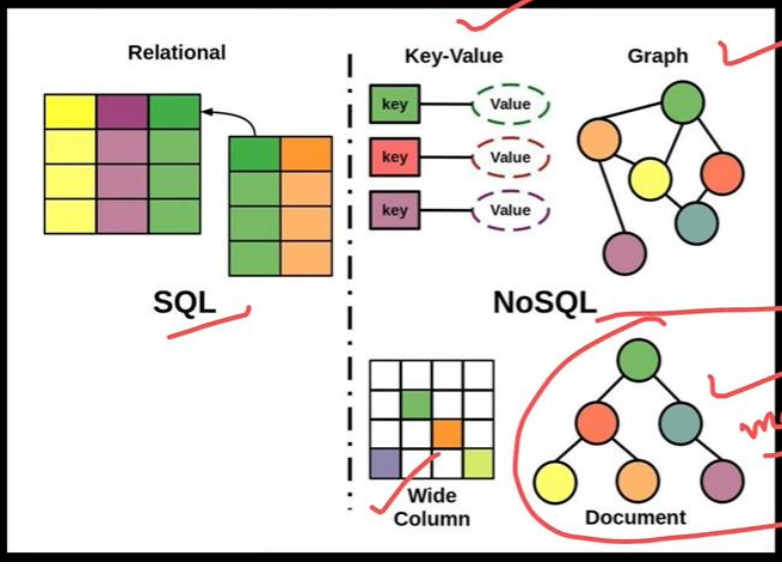
15.3 Introduction to NoSQL DB

- 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.

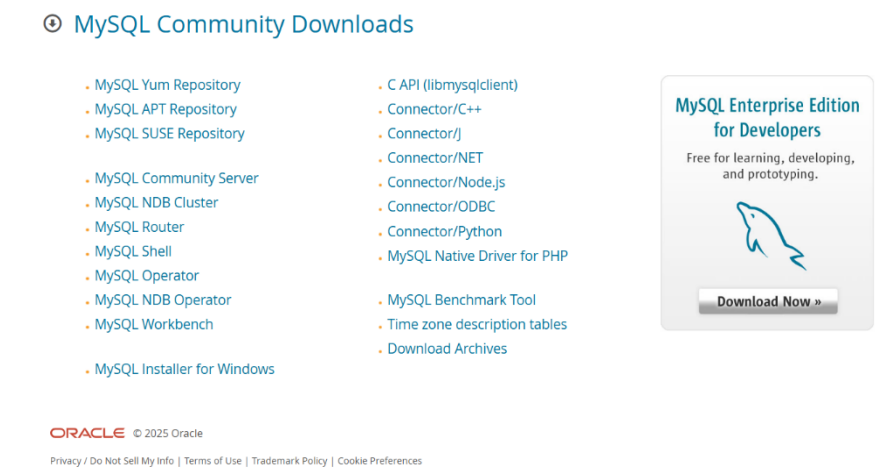
15.4 SQL vs NoSQL

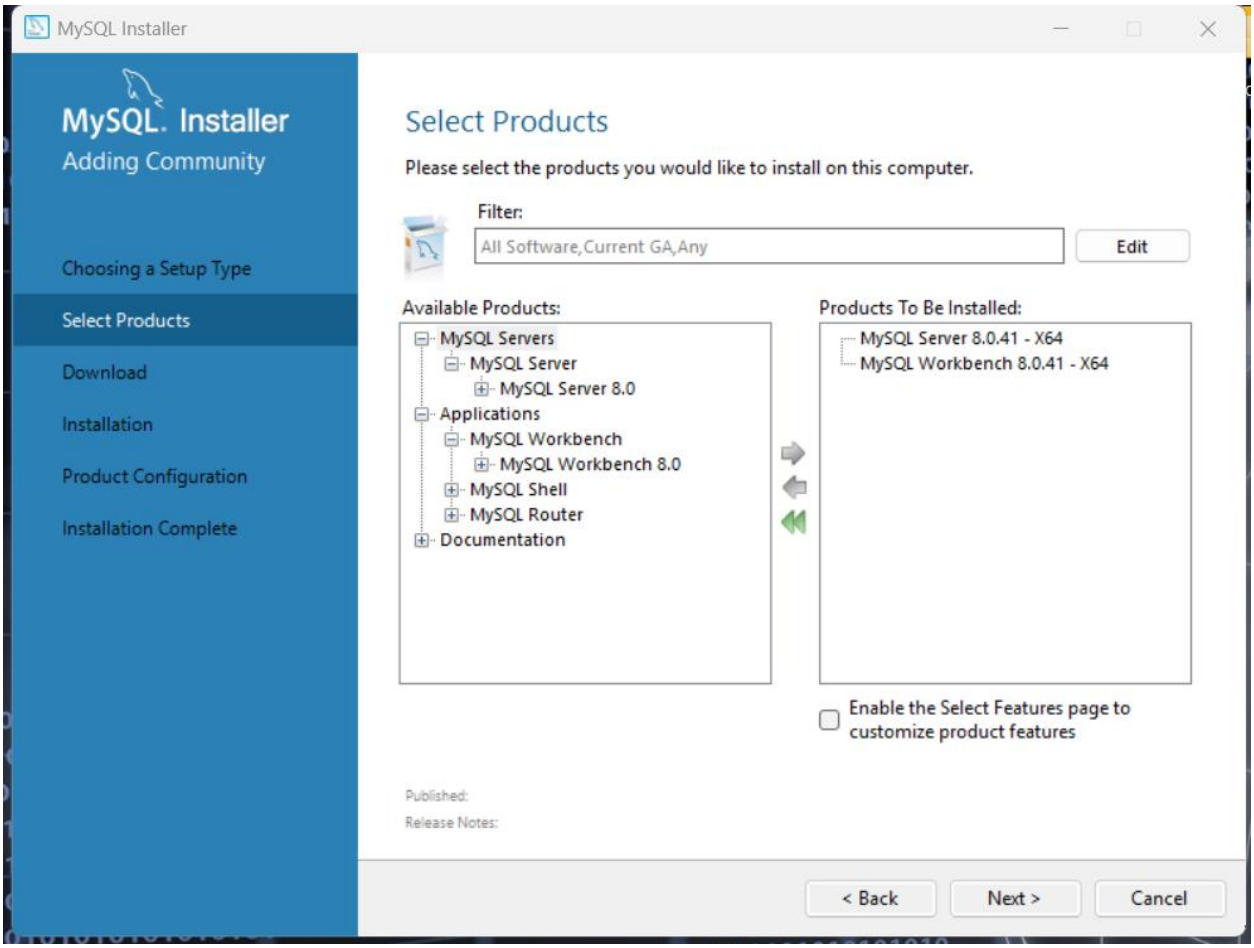


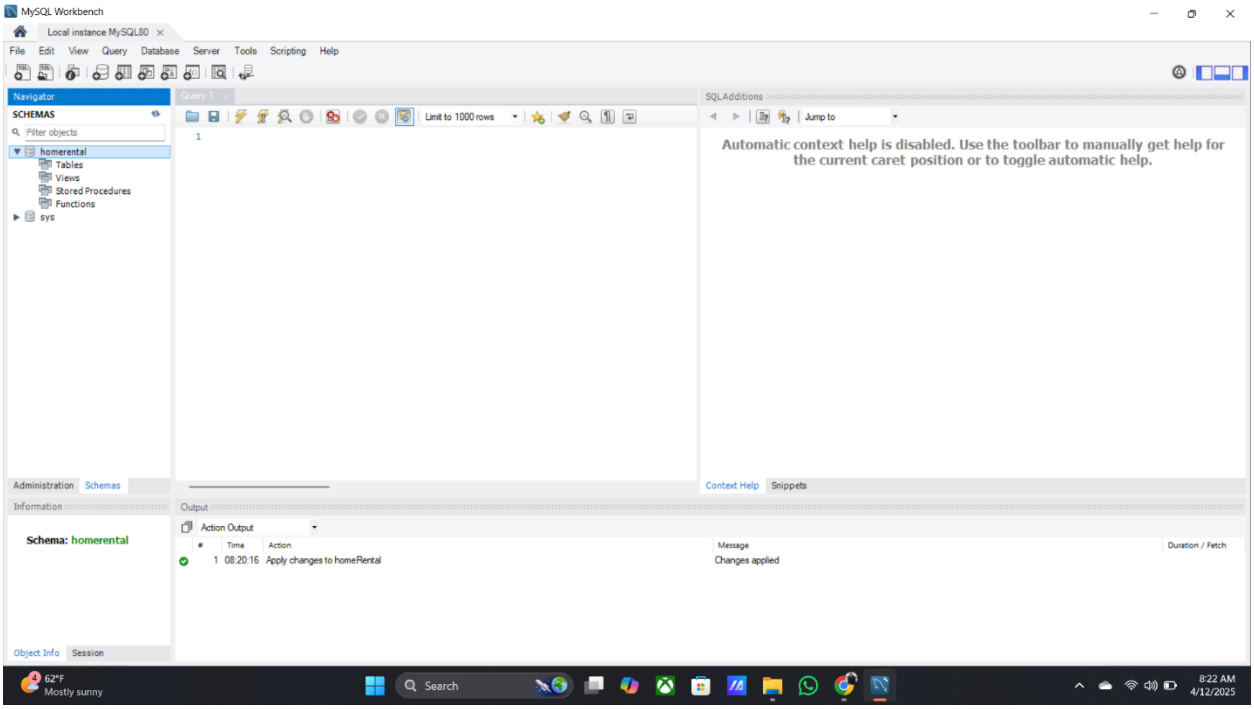
15.4 SQL vs NoSQL

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
Use Cases	Structured data and complex queries	Unstructured data and real-time app

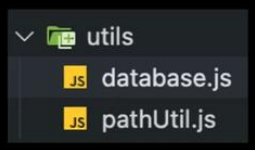
My SQL download process::




MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Download
Installation
Product Configuration
Installation Complete
Select Products
Please select the products you would like to install on this computer.
Filter:
All Software,Current GA,Any
Edit
Available Products:
MySQL Servers
MySQL Server
MySQL Server 8.0
Applications
MySQL Workbench
MySQL Workbench 8.0
MySQL Shell
MySQL Router
Documentation
Products To Be Installed:
MySQL Server 8.0.41 - X64
MySQL Workbench 8.0.41 - X64
Enable the Select Features page to customize product features
Published:
Release Notes:
< Back
Next >
Cancel


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
homerental
Tables
Views
Stored Procedures
Functions
sys
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Schema: homerental
Action Output
1 08:20:16 Apply changes to homeRental
Changes applied
62°F
Mostly sunny
Search
8:22 AM
4/12/2025

15.6 Connecting App to DB

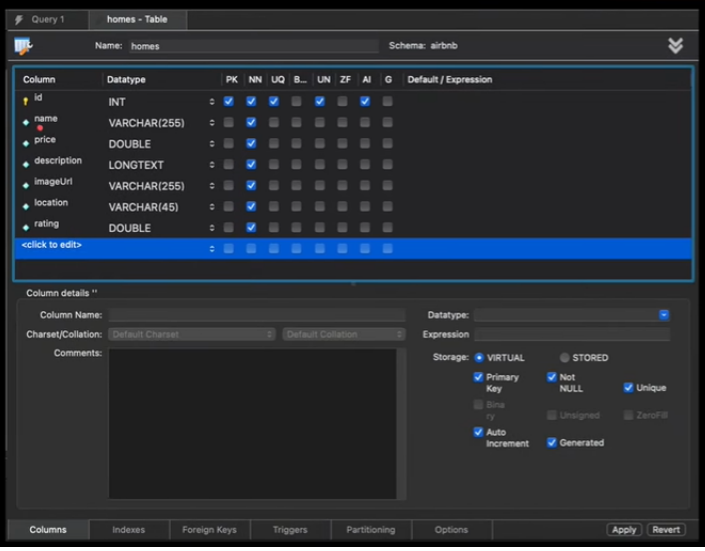



```
prashantjain@Prashants-Mac-mini Chapter 13 - MVC % npm install --save mysql2

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 > ...
1  const mysql = require("mysql2");
2
3  const pool = mysql.createPool({
4    host: "localhost",
5    user: "root",
6    password: "CompleteCoding@01",
7    database: "airbnb",
8  });
9
10 module.exports = pool.promise();
```

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

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);
    });
};
```

15.10 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.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          error);
  });
};
```

```
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);
  });
};
```

Practise Milestone (Solution)

```
save() {
  if (this.id) {
    // Update existing home
    return airbnbDb.execute(
      `UPDATE homes SET houseName=?, price=?, location=?, rating=?, photoUrl=?, description=? WHERE id=?`,
      [this.houseName, this.price, this.location, this.rating, this.photoUrl, this.description, this.id]
    );
  } else {
    //return airbnbDb.execute(`INSERT INTO homes (houseName, price, location, rating, photoUrl,
    description) VALUES ('${this.houseName}', ${this.price}, '${this.location}', ${this.rating}, '${this.
    photoUrl}', '${this.description}')`);
    // Insert new home
    return airbnbDb.execute(
      `INSERT INTO homes (houseName, price, location, rating, photoUrl, description) VALUES
      (?, ?, ?, ?, ?, ?)`,
      [this.houseName, this.price, this.location, this.rating, this.photoUrl, this.description]
    );
  }
}
```