

# RUMAD

Rutgers Mobile App Development

# Overview

For this section, we'll be covering:

- Databases
- Setting up Supabase

# Databases

# What is a database?

- Collection of structured information (data)
- Allows users to store, retrieve, update, and delete data efficiently

Columns (Attributes)

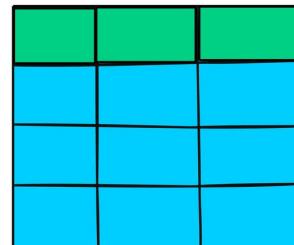
First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itottlaw	28

Record

# Types of Databases?

- Relational Databases
  - Tabular format
  - Uses SQL for querying
- Non-Relational (NoSQL) Databases
  - More flexibility in the way data is stored
  - No standardized querying language

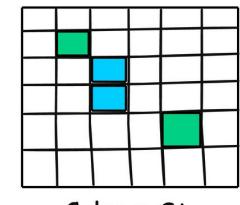
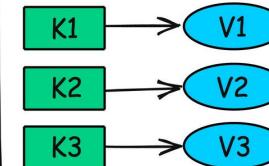
SQL



Relational

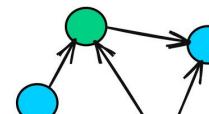
[blog.algomaster.io](http://blog.algomaster.io)

NoSQL

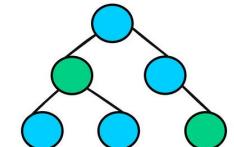


Column Store

Key-Value



Graph



Document

# Why do we need databases?

Recall the homework exercises we did:

- We have a tasks-cases.json with all our tasks.
- We imported these task cases into an **in-memory** list called **tasks**.
- We did operations on this list but what happens when we restart our server?  
What happens when we need to store 100K → 1M tasks?

Databases are essential for storing large amounts of data in one place. With databases, organizations can quickly access, manage, modify, update, organize and retrieve their data.

# What is a DBMS?

- Database Management System
  - Software for creating, managing, and interacting with databases
- Examples of DBMS:
  - mySQL
  - MongoDB
  - Supabase



PostgreSQL

# Relational Databases

# What is a Table?

- A database is a collection of **tables**.
- A table is a structured way of storing data in rows and columns.
- Row = a record in your table
- Column = an attribute/property

**Ex:** You can have a table of students with columns: name, netID, major

The rows of the table is each instance of a student.

## Columns (Attributes)

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaaw	28

## Rows (Records)

# How to interact with a DB?

## C.R.U.D Operations

1. **Create**
  - a. Inserting a record of data into your DB
2. **Read**
  - a. Getting>Selecting data from your DB
3. **Update**
  - a. Updating an existing record
4. **Delete**
  - a. Deleting a record from your DB

We'll go into more detail later!

# A.C.I.D. Properties

A good relational database is **ACID**-compliant

## A - Atomicity

- “All or Nothing” - if one part of the transaction fails, all changes are reverted

## C - Consistency

- All rules/constraints are followed no matter what.

## I - Isolation

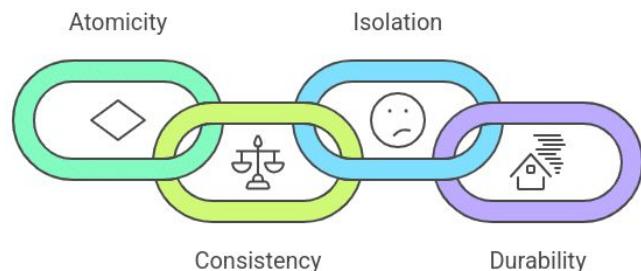
- When concurrent transactions are made, each one behaves as if it's the only one.

## D - Durability

- Once a transaction is successfully completed, the result is permanent. (Crashes & restarts don't wipe/revert the database)

**Transaction** = sequence of one or more operations on the DB

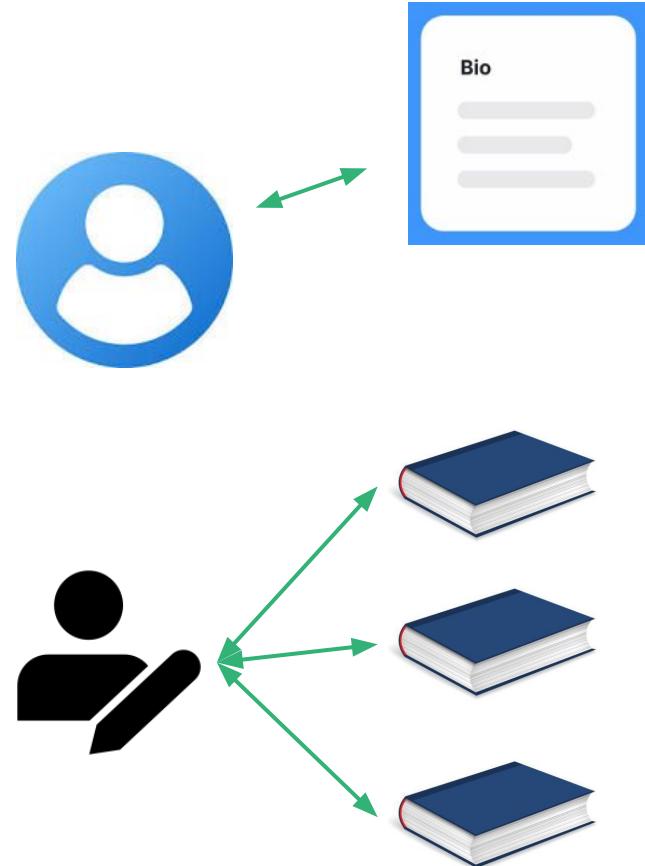
ACID Properties in Database



# SQL Relationships

There are 4 main kinds of relationships between tables in Relational Databases:

1. One to One
  - a. **One** Record in Table A → **One** Record in Table B
  - b. Ex: A user has only one user profile and vice-versa
2. One to Many/Many to One
  - a. **One** Record in Table A → **Many** Records in Table B
  - b. Ex: An author can write many books; many books can be associated with one author



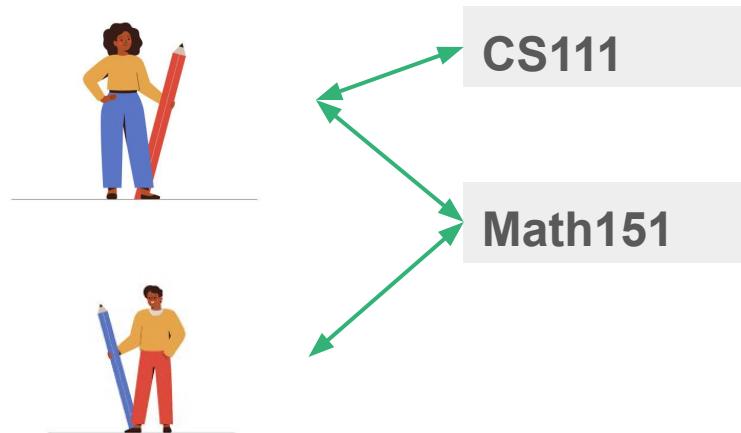
# SQL Relationships Cont.

## 1. Many to Many

- a. Many Records in Table A → Many Records in Table B
- b. Ex: A student can enroll in many courses and courses can have many students

## 2. Self-Referencing

- a. Record(s) in Table A → Record(s) in Table A



employees		
employee_id	employee_name	manager_id
1	Alice	NULL
2	Bob	1
3	Charlie	1

foreign key  
primary key

# Primary Keys

- **Uniquely** identifies a row through some sort of id.
- Examples:
  - customer\_id
  - student\_id
- What was the primary key for tasks in our homework exercises?

Customer ID	Forename	Surname
1	Simon	Jones
2	Emma	Price
3	Laura	Jones
4	Jonathan	Hale
5	Emma	Smith

Simple primary key

# Foreign Keys

- This is how we implement our SQL relationships
- A foreign key in one column **references** a primary key in another column
- These columns could be in different tables or the in the same table

Primary Key			Foreign Key	
ID	Name	Course	ID	Marks
1041	Sara	Java	1041	65
1204	Aryan	C++	1204	55
1043	Sameer	Python	1043	73
1032	Abhijeet	Oracle	1032	62

**Students Details**                    **Students Marks**

What kind of relationship might exist between students and their grades?

# Foreign Keys

Example:

We have an Authors table and Books table

How do we establish this relation?:

1 author → the books that the author wrote

We create a column in our **books** table called **author\_id**. This will be our foreign key.

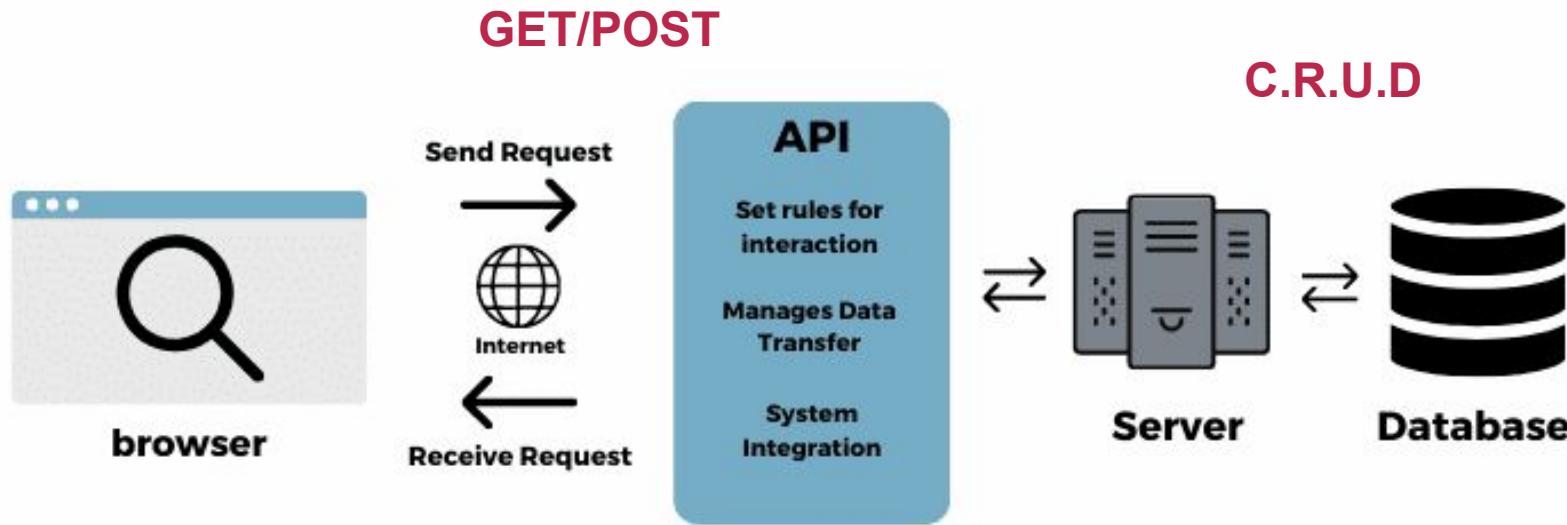
We make author\_id reference **id** in our authors table.

AUTHORS

<b>id</b>	<b>name</b>
1	Jane Austen
2	Stephen King

BOOKS

<b>id</b>	<b>title</b>	<u><b>author_id</b></u>
1	Pet Cemetery	2
2	IT	2



# Data Flow within an Application

The background of the image is a solid red color with a subtle, abstract pattern. It features several organic, rounded shapes in a lighter shade of red, resembling stylized leaves or petals. These shapes are scattered across the frame, with some overlapping and others positioned more centrally. The overall effect is clean and modern.

# Supabase

# What is Supabase?

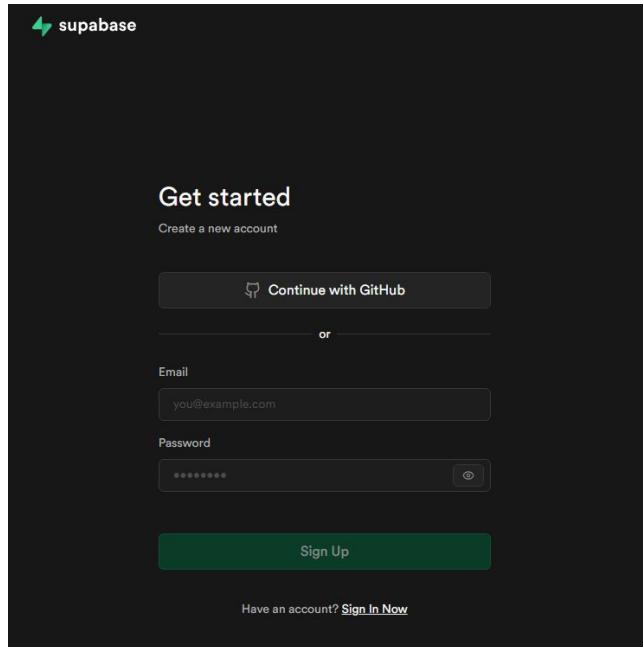
- Supabase is an open-source cloud database
  - Relational Database
  - The database exists on a cloud server
  - You don't need to host on your PC
  - BaaS – backend as a service
- Supabase is PostgreSQL\*-based
  - Real-time functionality
  - Storage & authentication
  - SQL Templates
    - If you don't know SQL!
  - Policies

\*PostgreSQL is another commonly used database

- Supabase is built upon Postgres (abstraction of Postgres)



# Set Up on Supabase Website



Sign in to Supabase  
- Github or Email

The image shows the 'Create a new organization' page. The title 'Create a new organization' is at the top. Below it, a descriptive text states: 'This is your organization within Supabase. For example, you can use the name of your company or department.' There are several input fields: 'Name' (with placeholder 'Organization name'), 'Type of organization' (set to 'Personal'), 'Plan' (set to 'Free - \$0/month'), and 'Pricing' (with a dropdown arrow). A note at the bottom left says 'You can rename your organization later'. At the bottom right is a large green 'Create organization' button.

Create an organization

# Project Creation

- Create a project
- You can have multiple projects inside an organization
- Set a strong password!

Create a new project  
Your project will have its own dedicated instance and full Postgres database.  
An API will be set up so you can easily interact with your new database.

Organization

Project name

Database Password   Copy  
This password is strong. [Generate a password](#)

Region

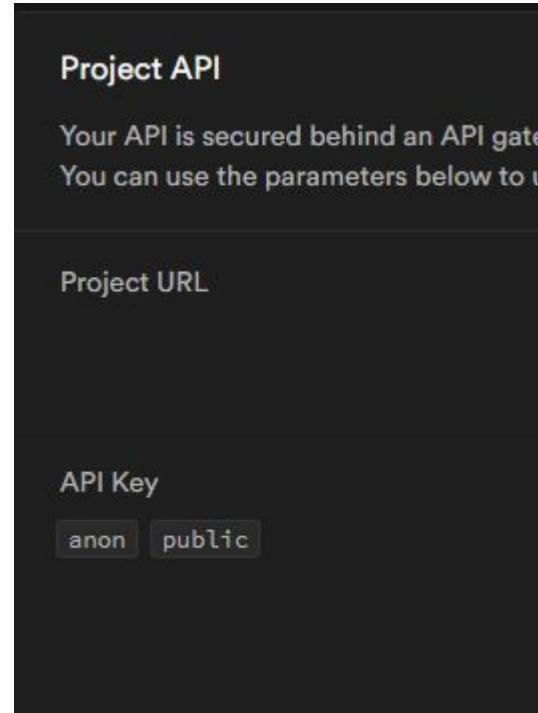
Select the region closest to your users for the best performance.

SECURITY OPTIONS >

You can rename your project later

# Supabase Config

- After basic setup of your Supabase project, you will get a list of information:
  - Project URL
  - API Key
- Used to connect your app to supabase
- This information is specific to **your database**
- Your API key should be maintained securely
  - Don't want other people accessing your database!



# Installing Supabase for Nodejs

- Use node package manager to install supabase
- Initialize supabase client
  - Supabase Url (domain)
  - Supabase Key
- **createClient** takes in your personalized url and key to connect to your database



```
Terminal 1 npm install @supabase/supabase-js
```

```
require('dotenv').config()  
const { createClient } = require('@supabase/supabase-js')  
  
//your Supabase url + public key from the supabase website!  
const domain = process.env.SUPABASE_DOMAIN  
const supabase_public_key = process.env.SUPABASE_PUBLIC_KEY  
  
const supabase = createClient(domain, supabase_public_key)
```

# Questions?

Please fill out the feedback form when you have a chance!

**Feedback Form**



# Next week...

- More on Supabase
  - CRUD Operations
  - Supabase w/ NodeJS
  - Supabase Authentication

# Did you complete Week 0?



Scan here for a guide to setting up your  
development environment!