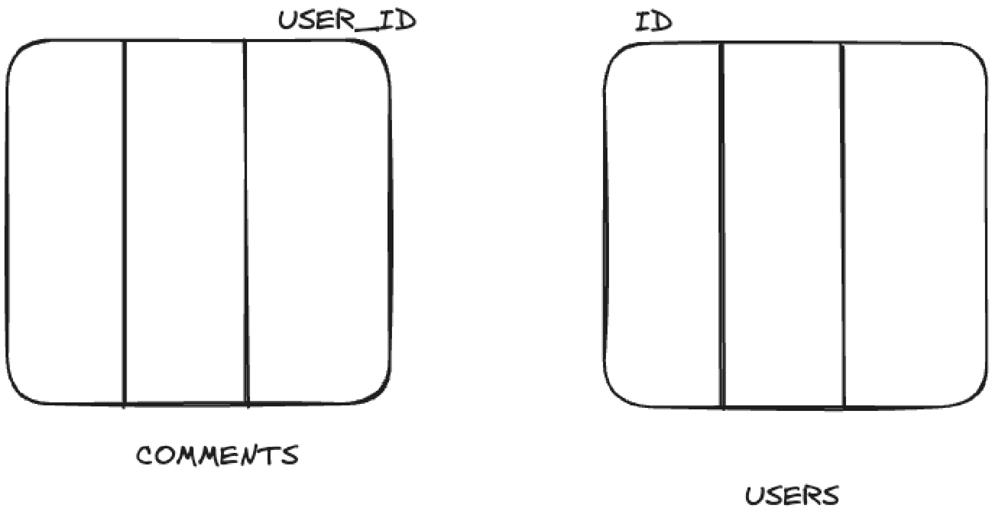
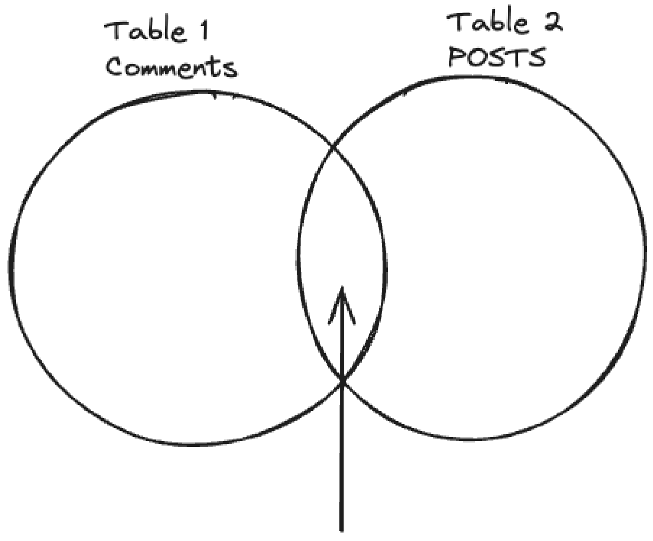
Database designing

JOINS



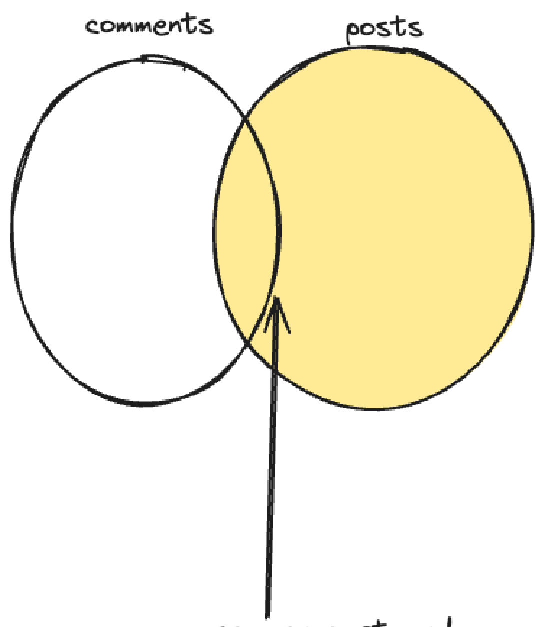


Inner join

id	content	postid
1	a	1
2	b	2
3	c	2

id	content
1	xyz
2	abc
3	def

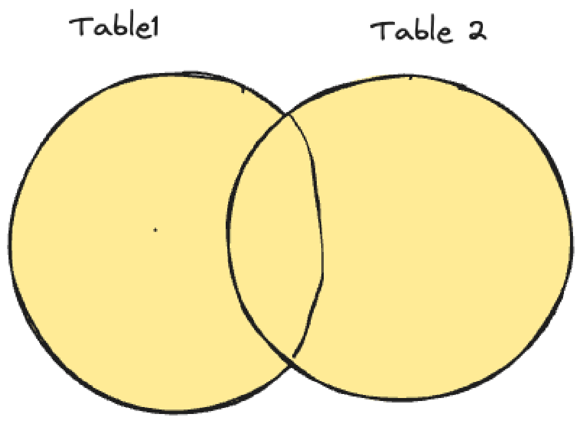
commentId	postid	content	postcontent
1	1	a	xyz
2	2	b	abc
3	2	c	abc



common part and
part only present in posts

↓

RIGHT JOIN -> COMMON RECORDS +
(NULL, REMAINING POSTS)



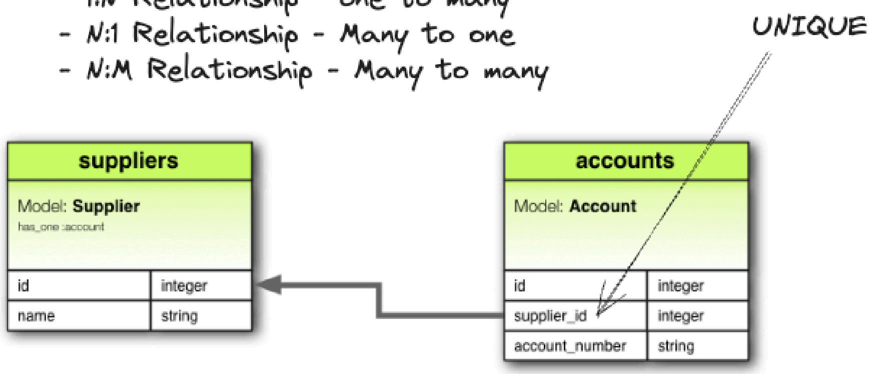
Full outer join

Relational

- 1:1 Relationship - one to one
- 1:N Relationship - one to many
- N:1 Relationship - Many to one
- N:M Relationship - Many to many

one:one relation

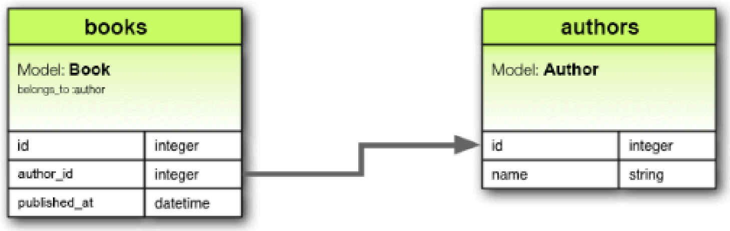
A supplier has one
account and
one account
belongs to a supplier



```
class Supplier < ApplicationRecord
  has_one :account
end
```

Many to one

A book belongs to
an author but
an author can have



```
class Book < ApplicationRecord
```

many books

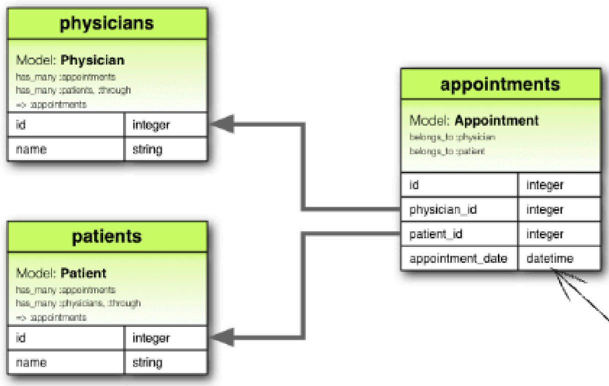
```
  belongs_to :author
end
```

A comment belongs to a post
but a post can get many comments

Quention has many answers but an answer
belongs to a question

Many to many

Physician can treat
many patients and
a patient can be
treated by many
physicians

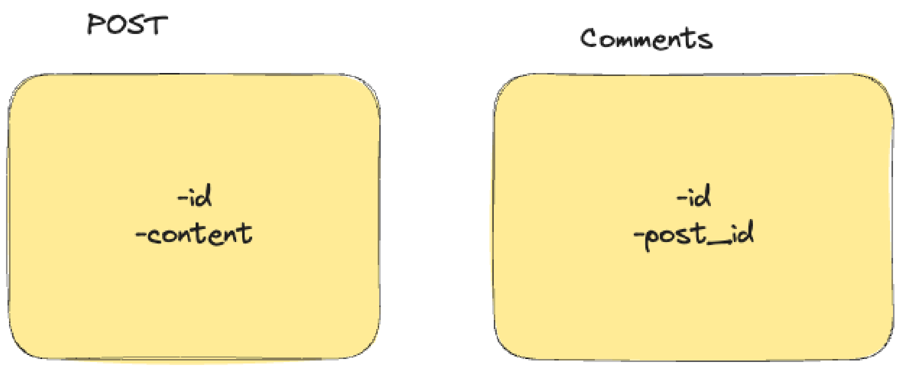


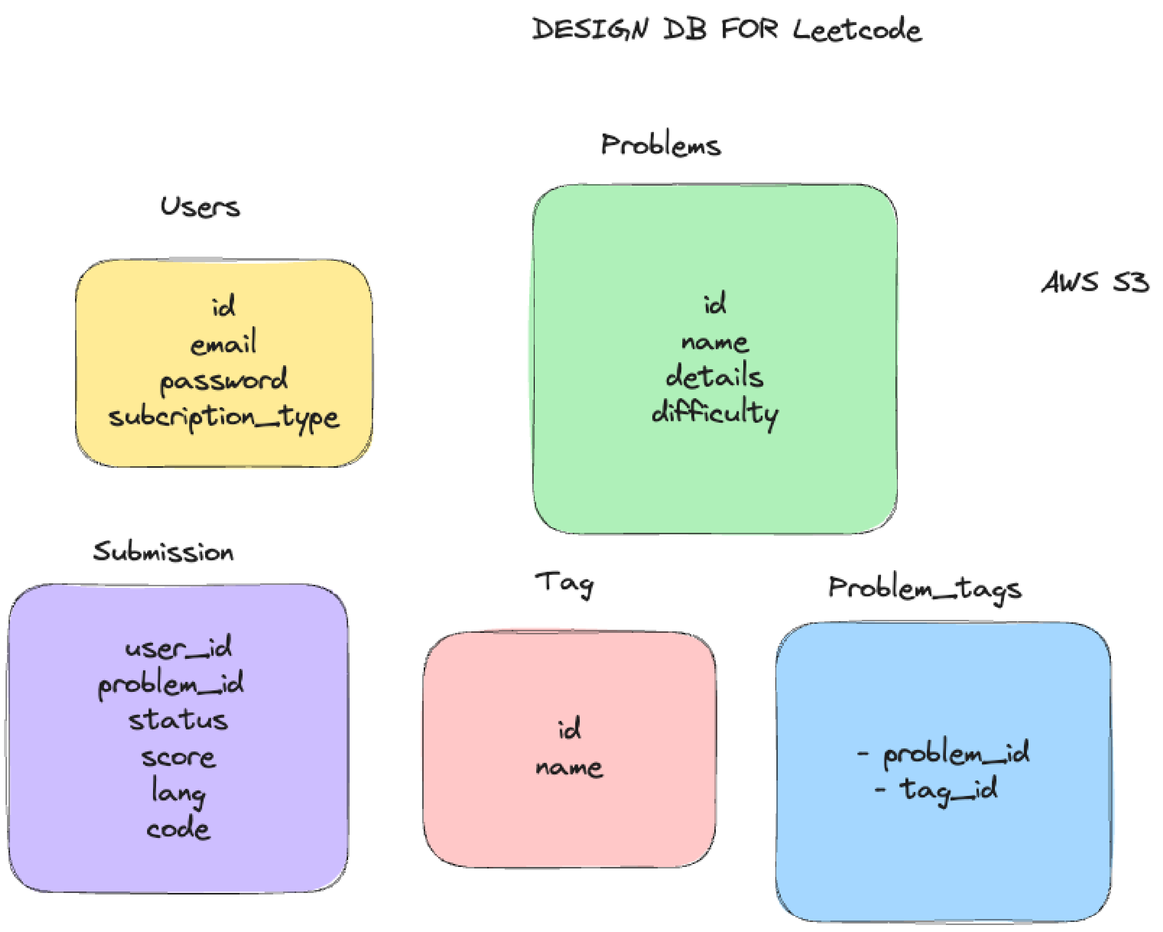
JOIN TABLE
OR
THROUGH TAB

```
class Physician < ApplicationRecord
  has_many :appointments
  has_many :patients, :through => :appointments
end

class Appointment < ApplicationRecord
  belongs_to :physician
  belongs_to :patient
end

class Patient < ApplicationRecord
  has_many :appointments
  has_many :physicians, :through => :appointments
end
```





```sql
create DATABASE CLASSDB; -- Create a database using CREATE DATABASE command

SHOW DATABASES; -- List all the DBs in your MYSQL Server

CREATE DATABASE IF NOT EXISTS CLASSDB; -- it will only create the database
if it doesn't exits

DROP DATABASE CLASSDB; -- Deleting a database

USE CLASSDB; -- Select a DB to work
```

```sql
SHOW TABLES; -- List all the tables in the selected DB


CREATE DATABASE FBDB; -- Create a new database


USE FBDB; -- Select the new database


CREATE TABLE USERS (

EMAIL VARCHAR(50) ,

PASSWORD VARCHAR(50) ,

USERNAME VARCHAR(50) ,

ID INT PRIMARY KEY AUTO_INCREMENT

); -- Create a table


SHOW TABLES; -- List all the tables in the selected DB


DESC USERS; -- Describe the table


INSERT INTO USERS (USERNAME, EMAIL, PASSWORD) VALUES

('SANKET', 'SANKET@GMAIL.COM', '123456'); -- Insert data into the table


INSERT INTO USERS (USERNAME, EMAIL, PASSWORD) VALUES

('SARTHAK', 'SJ@GMAIL.COM', '123456'); -- Insert data into the table
```

```sql
SELECT ID, EMAIL, USERNAME FROM USERS; -- Select data from the table

SELECT * FROM USERS; -- Select all the data from the table

INSERT INTO USERS (USERNAME, EMAIL, PASSWORD) VALUES
('JD', 'JD@GMAIL.COM', '123456'),
('RIYA', 'RY@GMAIL.COM', '123456'),
('ROHIT', 'RR@GMAIL.COM', '123456') ; -- Insert multiple data into the table


-- CREATE A POSTS TABLE WITH ID, CONTENT, USER_ID, CREATED_AT COLUMNS

CREATE TABLE POSTS (
ID INT PRIMARY KEY AUTO_INCREMENT,
CONTENT VARCHAR(255),
USER_ID INT, -- TO WHOM THE POST BELONGS
CREATED_AT TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO POSTS (CONTENT, USER_ID) VALUES
('HELLO WORLD', 1); -- Insert data into the table

INSERT INTO POSTS (CONTENT, USER_ID) VALUES
```

```sql
('HELLO WORLD', 1); -- Insert data into the table



INSERT INTO POSTS (CONTENT, CREATED_AT, USER_ID) VALUES

('HELLO WORLD AGAIN', '2021-01-01 12:00:00', 1); -- Insert data into the
table



SELECT * FROM POSTS; -- Select all the data from the table



SELECT * FROM USERS WHERE ID = 3; -- Select all the data from the table



SELECT * FROM POSTS WHERE USER_ID = 1 AND CONTENT = 'HELLO WORLD'; -- Select
all the data from the table



-- OPERATOR IN MYSQL: =, !=, <, >, <=, >=, AND, OR, NOT, IN, BETWEEN, LIKE,
IS NULL, IS NOT NULL



SELECT * FROM POSTS WHERE CONTENT LIKE '%AGAIN'; -- Select all the data from
the table

-- %AGAIN% SUBSTRING MATCH

-- %AGAIN STARTS WITH ANYTHING BUT ENDS WITH AGAIN

-- AGAIN% STARTS WITH AGAIN BUT CAN HAVE ANYTHING AFTER THAT



SELECT * FROM POSTS WHERE CONTENT LIKE '%WORLD' ORDER BY CREATED_AT ASC;
```

```sql
DELETE FROM POSTS WHERE ID = 1; -- Delete a row from the table

DROP TABLE POSTS; -- Delete a table

UPDATE POSTS SET CONTENT = 'MY WORLD' WHERE ID = 2; -- Update a row in the
table


-- Pagination

-- If we want to fetch only x number of rows from the table

SELECT * FROM USERS LIMIT 2; -- Fetch only 2 rows

SELECT * FROM USERS LIMIT 2 OFFSET 4; -- Fetch only 2 rows starting from the
3rd row

SELECT * FROM USERS LIMIT 1 OFFSET 2;


CREATE TABLE COMMENTS (

ID INT PRIMARY KEY AUTO_INCREMENT,

CONTENT VARCHAR(255),

USER_ID INT, -- THE USER WHO MADE THE COMMENT

POST_ID INT, -- THE POST ON WHICH THE COMMENT IS MADE

CREATED_AT TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

```sql
INSERT INTO COMMENTS (CONTENT, USER_ID, POST_ID) VALUES

('NICE POST', 1, 1); -- Insert data into the table


INSERT INTO COMMENTS (CONTENT, USER_ID, POST_ID) VALUES

('NICE POST', 1, 2); -- Insert data into the table


SELECT * FROM COMMENTS; -- Select all the data from the table


DELETE FROM COMMENTS; -- NOT THE PREFFERED WAY


TRUNCATE TABLE COMMENTS; -- DELETE ALL THE ROWS FROM THE TABLE, FASTER THAN
DELETE


-- CREATE A TABLE FOR MANAGING LIKES

-- LIKES CAN BE DONE ON POSTS AND COMMENTS

-- ID, USER_ID, CREATED_AT, LIKEABLE_ID, LIKEABLE_TYPE (ENUM)

-- 1, 1, 2021-01-01 12:00:00, 1, POST


CREATE TABLE LIKES (

ID INT PRIMARY KEY AUTO_INCREMENT,

USER_ID INT NOT NULL,

CREATED_AT TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
LIKEABLE_ID INT,

LIKEABLE_TYPE ENUM('POST', 'COMMENT')

);



INSERT INTO LIKES (USER_ID, LIKEABLE_ID, LIKEABLE_TYPE) VALUES

(1, 1, 'POST'); -- Insert data into the table



INSERT INTO LIKES (USER_ID, LIKEABLE_ID, LIKEABLE_TYPE) VALUES

(1, 1, 'POST'); -- Insert data into the table



SELECT * FROM LIKES; -- Select all the data from the table



-- MODIFY THE DEFINITION OF THE TABLE

ALTER TABLE LIKES MODIFY LIKEABLE_TYPE ENUM('POST', 'COMMENT', 'REEL');



DESC LIKES; -- Describe the table



INSERT INTO LIKES (USER_ID, LIKEABLE_ID, LIKEABLE_TYPE) VALUES

(1, 1, 'REEL'); -- Insert data into the table



DROP TABLE LIKES; -- Delete a table



-- If we create a comment then it should have some check to identify whether
the post exists or not and the user exists or not
```

```sql
-- We can use foregin key here: A foreign key is a column or a group of columns in a table that reference the primary key of another table.



DROP TABLE COMMENTS;



-- Now make the comments using foreign key constraints

CREATE TABLE COMMENTS (

ID INT PRIMARY KEY AUTO_INCREMENT,

CONTENT VARCHAR(255),

USER_ID INT, -- THE USER WHO MADE THE COMMENT

POST_ID INT, -- THE POST ON WHICH THE COMMENT IS MADE

CREATED_AT TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

FOREIGN KEY (USER_ID) REFERENCES USERS(ID),

FOREIGN KEY (POST_ID) REFERENCES POSTS(ID)

);



desc comments;



SELECT * FROM COMMENTS;



-- TRY TO FETCH USER DETAILS AND POST DETAILS ALSO WHEN GETTING THE COMMENTS



SELECT * FROM COMMENTS INNER JOIN USERS ON COMMENTS.USER_ID = USERS.ID JOIN POSTS ON COMMENTS.POST_ID = POSTS.ID;
SELECT * FROM COMMENTS RIGHT JOIN POSTS ON COMMENTS.POST_ID = POSTS.ID;


SELECT * FROM POSTS LEFT JOIN COMMENTS ON POSTS.ID = COMMENTS.POST_ID;
```