SQL Database Course

14 April 2021 19:27

Following :-
SQL Tutorial - Full Database Course for Beginners
https://www.mikedane.com/databases/sql/

Using PopSQL Software

https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

Lesson 1 (SQL)

SQL is a language which is used to interact with rational database management systems.

Rational database management systems is a software application which we can use to create and manage different databases.

What is a Database : Any collection of related information - Phone Book, Shopping List, To do list, Facebook user base, etc.
It can be stored in different ways - on paper, in your mind, on a computer, etc.

Databases can be stored : On paper, In your mind, Computer, PowerPoint, comment section

Database Management Systems (DBMS) : A special software program that helps users to create and maintain a database

CRUD : Create, Read (Retrieve), Update, Delete

Two Types of Databases :

1) Rational Databases (SQL)
- Organize data into one or more tables
  - Each table has columns and rows
  - A unique key identifies each row

2) Non-Relational (noSQL / not just SQL)
- Organize data is anything but a traditional table
  - Key-value stores/hash
  - Documents (JSON, XML, etc)
  - Graphs
  - Flexible Tables

- **Relational Database Management Systems (RDBMS) :** Helps users create and maintain a relational database
  - mySQL, Oracle, postgreSQL, mariaDB, etc
- **Structured Query Language (SQL)**

Database Queries : Queries are requests made to the database management system for specific information (A google search is query)

Wrap Up

- Database is any collection of related information
- Computer are great for storing databases
- Database Management Systems (DBMS) make it easy to create, maintain and secure a database.
- DBMS allow you to perform the C.R.U.D operations and other administrative tasks
- Two types of Databases, Relational & Non-Relational
- Relational databases use SQL and store data in tables with rows and columns
- Non-Relational data store data using other data structures

Lesson 2 (Tables and Key)

Keys :
- Primary Key
- Surrogate Key
- Natural Key
- Foreign Key
- Composite Key

Lesson 3 (SQL Basic)

SQL is actually a hybrid language, it's basically **4 types of languages** in on

- Data Query Language (DQL)
- Data Definition Language (DDL)
- Data Control Language (DCL)
- Data Manipulation Language (DML)

Queries : A query is a set of instructions given to the RDBMS (written in SQL) that tell the RDBMS what information you want it to retr ieve for you.

Lesson 4 (MySQL Windows Installation)

https://dev.mysql.com/downloads/mysql/
Install MySQL and Shell only.

For Visualizing SQL queries, download popsql :
https://popsql.com/

Lesson 5 (Mysql Mac Installation)

Download mySQL Community Server - https://dev.mysql.com/downloads/mysql/
Follow : https://www.mikedane.com/databases/sql/mysql-mac-installation/

Lesson 6 (Creating Tables)

```
CREATE TABLE student (
   student_id INT PRIMARY KEY,
   name VARCHAR(20),
   major VARCHAR(20)
);

DESCRIBE student;

DROP TABLE student;

ALTER TABLE student ADD gpa DECIMAL(4, 2);

ALTER TABLE student DROP COLUMN gpa;
```

- Whenever we are creating a database, the first thing we do is define a schema, create all the different tables and then we st art inserting data.

Lesson 7 (Inserting Data)

```
CREATE TABLE student (
   student_id INT,
   name VARCHAR(20),
   major VARCHAR(20),
   PRIMARY KEY(student_id)
);

SELECT * FROM student;

INSERT INTO student VALUES(1, 'Jack', 'Biology');
INSERT INTO student VALUES(2, 'Kate', 'Sociology');

INSERT INTO student(student_id, name) VALUES(3, 'Claire');
INSERT INTO student(student_id, name) VALUES(4, 'Claire');
```

Lesson 8 (Constraints)

For Unique element in every coloumn : (Use this while creating a table)

```
   --major VARCHAR(20) UNIQUE,

CREATE TABLE student (
   student_id INT AUTO_INCREMENT,
   name VARCHAR(20) NOT NULL,
   major VARCHAR(20) DEFAULT 'undefined',
   PRIMARY KEY(student_id)
);

SELECT * FROM student;
```

```
INSERT INTO student(name) VALUES('Jack');
INSERT INTO student(name, major) VALUES('Kate', 'Sociology');
INSERT INTO student(major, name) VALUES('English', 'Claire');
INSERT INTO student(name, major) VALUES('Jack', 'Biology');
INSERT INTO student(name, major) VALUES('Mike', 'Computer Science');

DROP TABLE student;
```

Lesson 9 (Update & Delete)

```
CREATE TABLE student (
   student_id INT AUTO_INCREMENT,
   name VARCHAR(20) NOT NULL,
   major VARCHAR(20) DEFAULT 'undefined',
   PRIMARY KEY(student_id)
);

SELECT * FROM student;

INSERT INTO student(name) VALUES('Jack');
INSERT INTO student(name, major) VALUES('Kate', 'Sociology');
INSERT INTO student(major, name) VALUES('English', 'Claire');
INSERT INTO student(name, major) VALUES('Jack', 'Biology');
INSERT INTO student(name, major) VALUES('Mike', 'Computer Science');

DROP TABLE student;

UPDATE student
SET major = 'Bio'
WHERE major = 'Biology';

UPDATE student
SET major = 'Bio'
WHERE name = 'Jack';

UPDATE student
SET major = 'Biochemistry'
WHERE major = 'Bio' OR major = 'Chemistry';

UPDATE student
SET name = 'Tom', major = 'undefined'
WHERE student_id = 1;

DELETE from student
WHERE student_id = 3;

DELETE from student;    // to delete everything from the table.
```

Lesson 10 (Basic Queries)

```
SELECT name
FROM student;

SELECT student.name, stude
nt.major
FROM student
ORDER BY name DESC;

SELECT *
FROM student
ORDER BY major, student_id DESC;

SELECT *
FROM student
ORDER BY student_id DESC
LIMIT 2;

SELECT name, major
FROM student
WHERE major = 'Biology' OR name = 'Kate';

-- <, >, <=, >=, =, <>, AND, OR
-- <> is not equal
```

```
SELECT *
FROM student
WHERE student_id > 3;

SELECT *
FROM student
WHERE name IN ('Clair', 'Kate', 'Mike');
```

Lesson 11 (Company Database Intro)

Lesson 12 (Creating Company Database)

```
CREATE TABLE employee (
  emp_id INT PRIMARY KEY,
  first_name VARCHAR(40),
  last_name VARCHAR(40),
  birth_day DATE,
  sex VARCHAR(1),
  salary INT,
  super_id INT,
  branch_id INT
);

CREATE TABLE branch (
  branch_id INT PRIMARY KEY,
  branch_name VARCHAR(40),
  mgr_id INT,
  mgr_start_date DATE,
  FOREIGN KEY(mgr_id) REFERENCES employee(emp_id) ON DELETE SET NULL
);

ALTER TABLE employee
ADD FOREIGN KEY(branch_id)
REFERENCES branch(branch_id)
ON DELETE SET NULL;

ALTER TABLE employee
ADD FOREIGN KEY(super_id)
REFERENCES employee(emp_id)
ON DELETE SET NULL;

CREATE TABLE client (
  client_id INT PRIMARY KEY,
  client_name VARCHAR(40),
  branch_id INT,
  FOREIGN KEY(branch_id) REFERENCES branch(branch_id) ON DELETE SET NULL
);

CREATE TABLE works_with (
  emp_id INT,
  client_id INT,
  total_sales INT,
  PRIMARY KEY(emp_id, client_id),
  FOREIGN KEY(emp_id) REFERENCES employee(emp_id) ON DELETE CASCADE,
  FOREIGN KEY(client_id) REFERENCES client(client_id) ON DELETE CASCADE
);

CREATE TABLE branch_supplier (
  branch_id INT,
  supplier_name VARCHAR(40),
  supply_type VARCHAR(40),
  PRIMARY KEY(branch_id, supplier_name),
  FOREIGN KEY(branch_id) REFERENCES branch(branch_id) ON DELETE CASCADE
);


-- ---------------------------------------------------------------------------

-- Corporate
```

```
INSERT INTO employee VALUES(100, 'David', 'Wallace', '1967-11-17', 'M', 250000, NULL, NULL);

INSERT INTO branch VALUES(1, 'Corporate', 100, '2006-02-09');

UPDATE employee
SET branch_id = 1
WHERE emp_id = 100;

INSERT INTO employee VALUES(101, 'Jan', 'Levinson', '1961-05-11', 'F', 110000, 100, 1);

-- Scranton
INSERT INTO employee VALUES(102, 'Michael', 'Scott', '1964-03-15', 'M', 75000, 100, NULL);

INSERT INTO branch VALUES(2, 'Scranton', 102, '1992-04-06');

UPDATE employee
SET branch_id = 2
WHERE emp_id = 102;

INSERT INTO employee VALUES(103, 'Angela', 'Martin', '1971-06-25', 'F', 63000, 102, 2);
INSERT INTO employee VALUES(104, 'Kelly', 'Kapoor', '1980-02-05', 'F', 55000, 102, 2);
INSERT INTO employee VALUES(105, 'Stanley', 'Hudson', '1958-02-19', 'M', 69000, 102, 2);

-- Stamford
INSERT INTO employee VALUES(106, 'Josh', 'Porter', '1969-09-05', 'M', 78000, 100, NULL);

INSERT INTO branch VALUES(3, 'Stamford', 106, '1998-02-13');

UPDATE employee
SET branch_id = 3
WHERE emp_id = 106;

INSERT INTO employee VALUES(107, 'Andy', 'Bernard', '1973-07-22', 'M', 65000, 106, 3);
INSERT INTO employee VALUES(108, 'Jim', 'Halpert', '1978-10-01', 'M', 71000, 106, 3);


-- BRANCH SUPPLIER
INSERT INTO branch_supplier VALUES(2, 'Hammer Mill', 'Paper');
INSERT INTO branch_supplier VALUES(2, 'Uni-ball', 'Writing Utensils');
INSERT INTO branch_supplier VALUES(3, 'Patriot Paper', 'Paper');
INSERT INTO branch_supplier VALUES(2, 'J.T. Forms & Labels', 'Custom Forms');
INSERT INTO branch_supplier VALUES(3, 'Uni-ball', 'Writing Utensils');
INSERT INTO branch_supplier VALUES(3, 'Hammer Mill', 'Paper');
INSERT INTO branch_supplier VALUES(3, 'Stamford Lables', 'Custom Forms');

-- CLIENT
INSERT INTO client VALUES(400, 'Dunmore Highschool', 2);
INSERT INTO client VALUES(401, 'Lackawana Country', 2);
INSERT INTO client VALUES(402, 'FedEx', 3);
INSERT INTO client VALUES(403, 'John Daly Law, LLC', 3);
INSERT INTO client VALUES(404, 'Scranton Whitepages', 2);
INSERT INTO client VALUES(405, 'Times Newspaper', 3);
INSERT INTO client VALUES(406, 'FedEx', 2);

-- WORKS_WITH
INSERT INTO works_with VALUES(105, 400, 55000);
INSERT INTO works_with VALUES(102, 401, 267000);
INSERT INTO works_with VALUES(108, 402, 22500);
INSERT INTO works_with VALUES(107, 403, 5000);
INSERT INTO works_with VALUES(108, 403, 12000);
INSERT INTO works_with VALUES(105, 404, 33000);
INSERT INTO works_with VALUES(107, 405, 26000);
INSERT INTO works_with VALUES(102, 406, 15000);
INSERT INTO works_with VALUES(105, 406, 130000);
```

Lesson 13 (More Basic Queries)

```
-- Find all employees
SELECT *
FROM employee;

-- Find all clients
SELECT *
FROM clients;

-- Find all employees ordered by salary
```

```
SELECT *
from employee
ORDER BY salary ASC/DESC;

-- Find all employees ordered by sex then name
SELECT *
from employee
ORDER BY sex, name;

-- Find the first 5 employees in the table
SELECT *
from employee
LIMIT 5;

-- Find the first and last names of all employees
SELECT first_name, employee.last_name
FROM employee;

-- Find the forename and surnames names of all employees
SELECT first_name AS forename, employee.last_name AS surname
FROM employee;

-- Find out all the different genders
SELECT DISCINCT sex
FROM employee;

-- Find all male employees
SELECT *
FROM employee
WHERE sex = 'M';

-- Find all employees at branch 2
SELECT *
FROM employee
WHERE branch_id = 2;

-- Find all employee's id's and names who were born after 1969
SELECT emp_id, first_name, last_name
FROM employee
WHERE birth_day >= 1970-01-01;

-- Find all female employees at branch 2
SELECT *
FROM employee
WHERE branch_id = 2 AND sex = 'F';

-- Find all employees who are female & born after 1969 or who make over 80000
SELECT *
FROM employee
WHERE (birth_day >= '1970-01-01' AND sex = 'F') OR salary > 80000;

-- Find all employees born between 1970 and 1975
SELECT *
FROM employee
WHERE birth_day BETWEEN '1970-01-01' AND '1975-01-01';

-- Find all employees named Jim, Michael, Johnny or David
SELECT *
FROM employee
WHERE first_name IN ('Jim', 'Michael', 'Johnny', 'David');
```

Lesson 14 (Functions)

```
-- Find the number of employees
SELECT COUNT(super_id)
FROM employee;

-- Find the average of all employee's salaries
SELECT AVG(salary)
FROM employee;

-- Find the sum of all employee's salaries
SELECT SUM(salary)
FROM employee;

-- Find out how many males and females there are
```

```
SELECT COUNT(sex), sex
FROM employee
GROUP BY sex

-- Find the total sales of each salesman
SELECT SUM(total_sales), emp_id
FROM works_with
GROUP BY client_id;

-- Find the total amount of money spent by each client
SELECT SUM(total_sales), client_id
FROM works_with
GROUP BY client_id;
```

Lesson 15 (Wildcards)

```
-- % = any # characters, _ = one character

-- Find any client's who are an LLC
SELECT *
FROM client
WHERE client_name LIKE '%LLC';

-- Find any branch suppliers who are in the label business
SELECT *
FROM branch_supplier
WHERE supplier_name LIKE '% Label%';

-- Find any employee born on the 10th day of the month
SELECT *
FROM employee
WHERE birth_day LIKE '_____10%';

-- Find any clients who are schools
SELECT *
FROM client
WHERE client_name LIKE '%Highschool%';
```

Lesson 16 (Union)

```
-- Find a list of employee and branch names
SELECT employee.first_name AS Employee_Branch_Names
FROM employee
UNION
SELECT branch.branch_name
FROM branch;

-- Find a list of all clients & branch suppliers' names
SELECT client.client_name AS Non-Employee_Entities, client.branch_id AS Branch_ID
FROM client
UNION
SELECT branch_supplier.supplier_name, branch_supplier.branch_id
FROM branch_supplier;
```

Lesson 17 (Joins)

```
-- Add the extra branch
INSERT INTO branch VALUES(4, "Buffalo", NULL, NULL);

SELECT employee.emp_id, employee.first_name, branch.branch_name
FROM employee
JOIN branch    -- LEFT JOIN, RIGHT JOIN
ON employee.emp_id = branch.mgr_id;
```

Lesson 18 (Nested Queries)

```
-- Find names of all employees who have sold over 50,000
```

```
SELECT employee.first_name, employee.last_name
FROM employee
WHERE employee.emp_id IN (SELECT works_with.emp_id
                    FROM works_with
                    WHERE works_with.total_sales > 50000);

-- Find all clients who are handles by the branch that Michael Scott manages
-- Assume you know Michael's ID
SELECT client.client_id, client.client_name
FROM client
WHERE client.branch_id = (SELECT branch.branch_id
                    FROM branch
                    WHERE branch.mgr_id = 102);

 -- Find all clients who are handles by the branch that Michael Scott manages
 -- Assume you DONT'T know Michael's ID
 SELECT client.client_id, client.client_name
 FROM client
 WHERE client.branch_id = (SELECT branch.branch_id
                     FROM branch
                     WHERE branch.mgr_id = (SELECT employee.emp_id
                                         FROM employee
                                         WHERE employee.first_name = 'Michael' AND employee.last_name ='Scott'
                                         LIMIT 1));


-- Find the names of employees who work with clients handled by the scranton branch
SELECT employee.first_name, employee.last_name
FROM employee
WHERE employee.emp_id IN (
                    SELECT works_with.emp_id
                    FROM works_with
                    )
AND employee.branch_id = 2;

-- Find the names of all clients who have spent more than 100,000 dollars
SELECT client.client_name
FROM client
WHERE client.client_id IN (
                    SELECT client_id
                    FROM (
                        SELECT SUM(works_with.total_sales) AS totals, client_id
                        FROM works_with
                        GROUP BY client_id) AS total_client_sales
                    WHERE totals > 100000
);
```

Lesson 19 (On Delete)

```
CREATE TABLE branch (
  branch_id INT PRIMARY KEY,
  branch_name VARCHAR(40),
  mgr_id INT,
  mgr_start_date DATE,
  FOREIGN KEY(mgr_id) REFERENCES employee(emp_id) ON DELETE SET NULL
);

DELETE
FROM employee
WHERE emp_id = 102;

SELECT *
FROM branch;

SELECT *
FROM employee;

CREATE TABLE branch_supplier (
  branch_id INT,
  supplier_name VARCHAR(40),
  supply_type VARCHAR(40),
  PRIMARY KEY(branch_id, supplier_name),
  FOREIGN KEY(branch_id) REFERENCES branch(branch_id) ON DELETE CASCADE
);

DELETE
FROM branch
```

```
WHERE branch_id = 2;

SELECT *
FROM branch_supplier;
```

Lesson 20 (Triggers)

Trigger is basically a block of SQL code which we can write, which will define a certain action that should happen when a cer tain operation gets performed on the database.

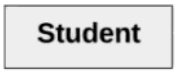
Delimiter is changing ; ending SQL with ; to some other symbol;

```
-- CREATE
--     TRIGGER `event_name` BEFORE/AFTER INSERT/UPDATE/DELETE
--     ON `database`.`table`
--     FOR EACH ROW BEGIN
--                     -- trigger body
--                     -- this code is applied to every
--                     -- inserted/updated/deleted row
--     END;

CREATE TABLE trigger_test (
    message VARCHAR(100)
);



DELIMITER $$
CREATE
    TRIGGER my_trigger BEFORE INSERT
    ON employee
    FOR EACH ROW BEGIN
        INSERT INTO trigger_test VALUES('added new employee');
    END$$
DELIMITER ;
INSERT INTO employee
VALUES(109, 'Oscar', 'Martinez', '1968-02-19', 'M', 69000, 106, 3);


DELIMITER $$
CREATE
    TRIGGER my_trigger BEFORE INSERT
    ON employee
    FOR EACH ROW BEGIN
        INSERT INTO trigger_test VALUES(NEW.first_name);
    END$$
DELIMITER ;
INSERT INTO employee
VALUES(110, 'Kevin', 'Malone', '1978-02-19', 'M', 69000, 106, 3);

DELIMITER $$
CREATE
    TRIGGER my_trigger BEFORE INSERT
    ON employee
    FOR EACH ROW BEGIN
        IF NEW.sex = 'M' THEN
            INSERT INTO trigger_test VALUES('added male employee');
        ELSEIF NEW.sex = 'F' THEN
            INSERT INTO trigger_test VALUES('added female');
        ELSE
            INSERT INTO trigger_test VALUES('added other employee');
        END IF;
    END$$
DELIMITER ;
INSERT INTO employee
VALUES(111, 'Pam', 'Beesly', '1988-02-19', 'F', 69000, 106, 3);


DROP TRIGGER my_trigger;
```

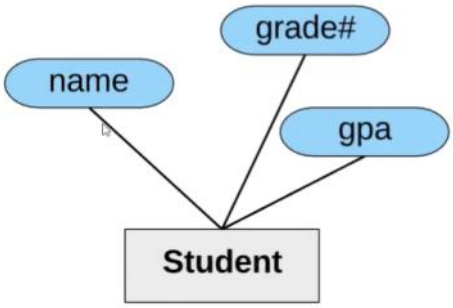
Lesson 21 (Er Diagrams Intro)

ER = Entity Relationship
ER Diagram = A great way to take us know data storage requirements like business requirement and convert them into an actual database sc hema.

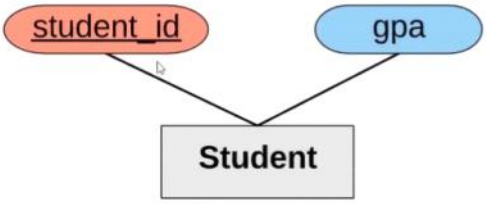
Entity - An object we want to model & store information about

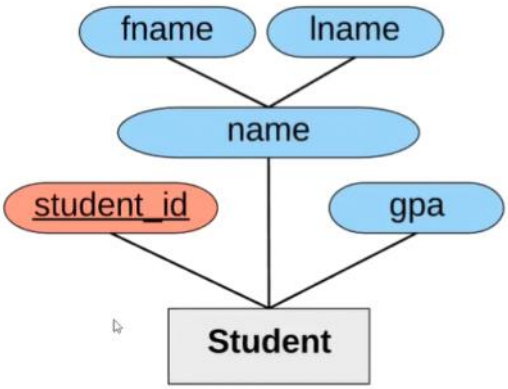


Attributes - Specific pieces of information about an entity



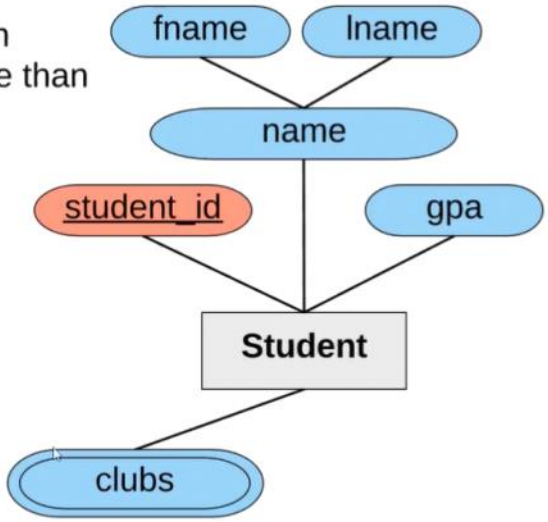
Primary Key - An attribute(s) that uniquely identify an entry in the database table



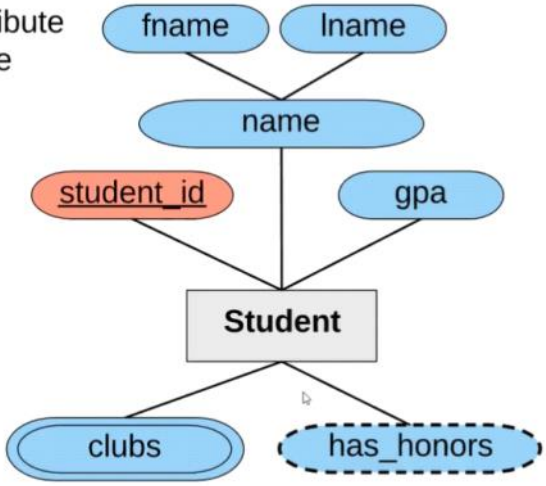


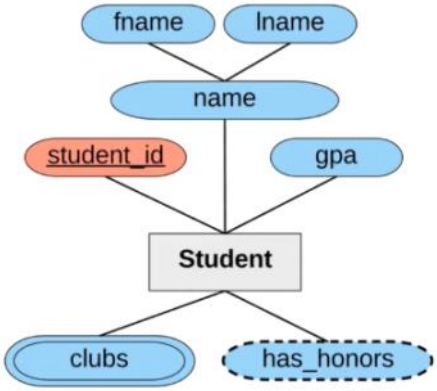
Composite Attribute - An attribute that can be broken up into sub-attributes

Multi-valued Attribute - An attribute that can have more than one value

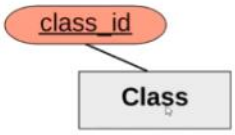


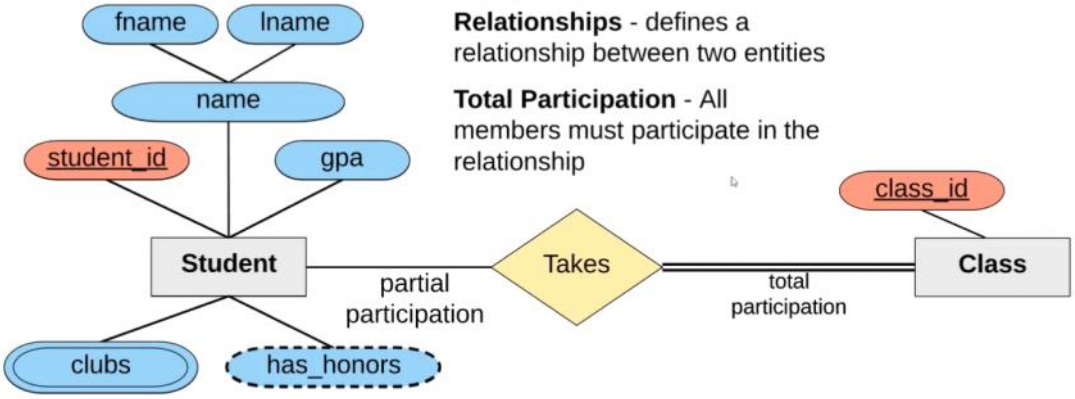
Derived Attribute - An attribute that can be derived from the other attributes





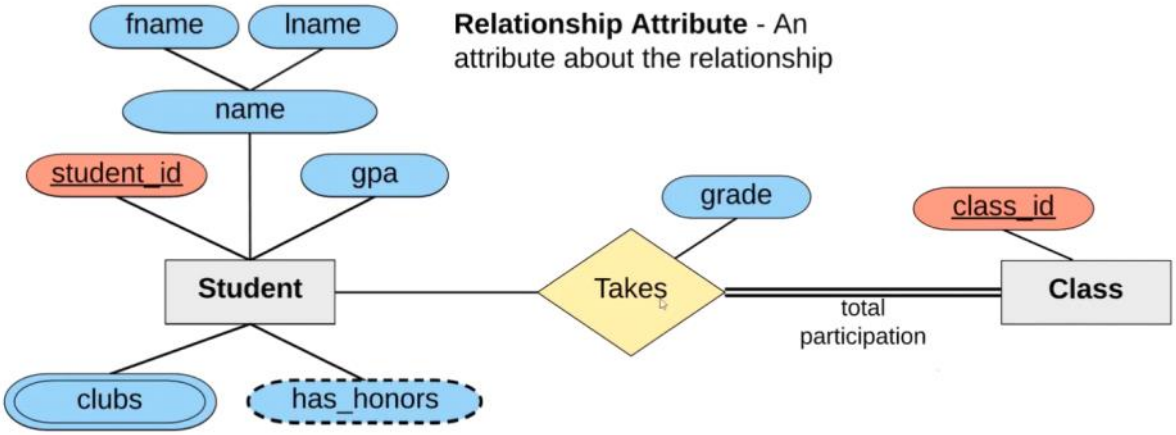
Multiple Entities - You can define more than one entity in the diagram



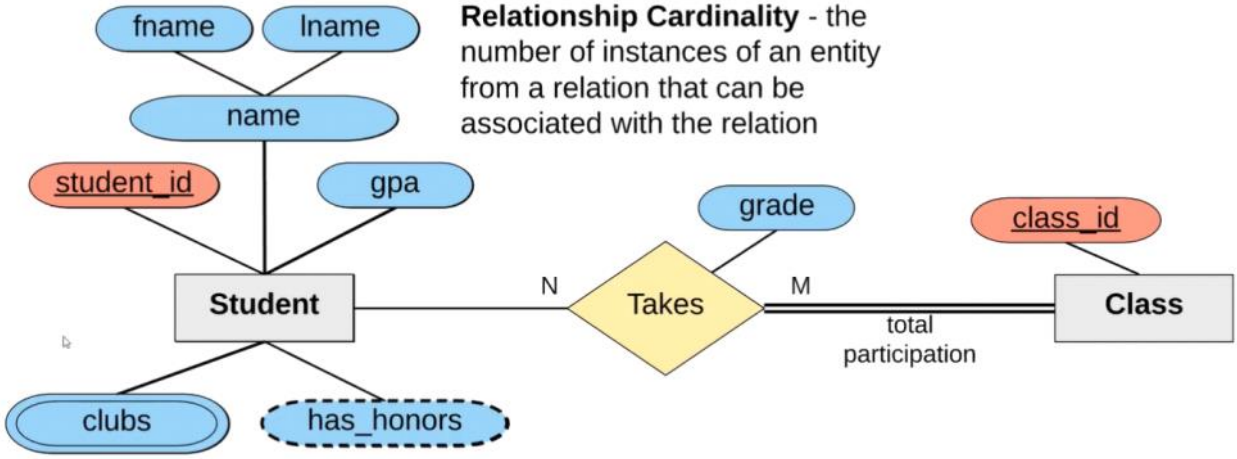


Relationships - defines a relationship between two entities

Total Participation - All members must participate in the relationship



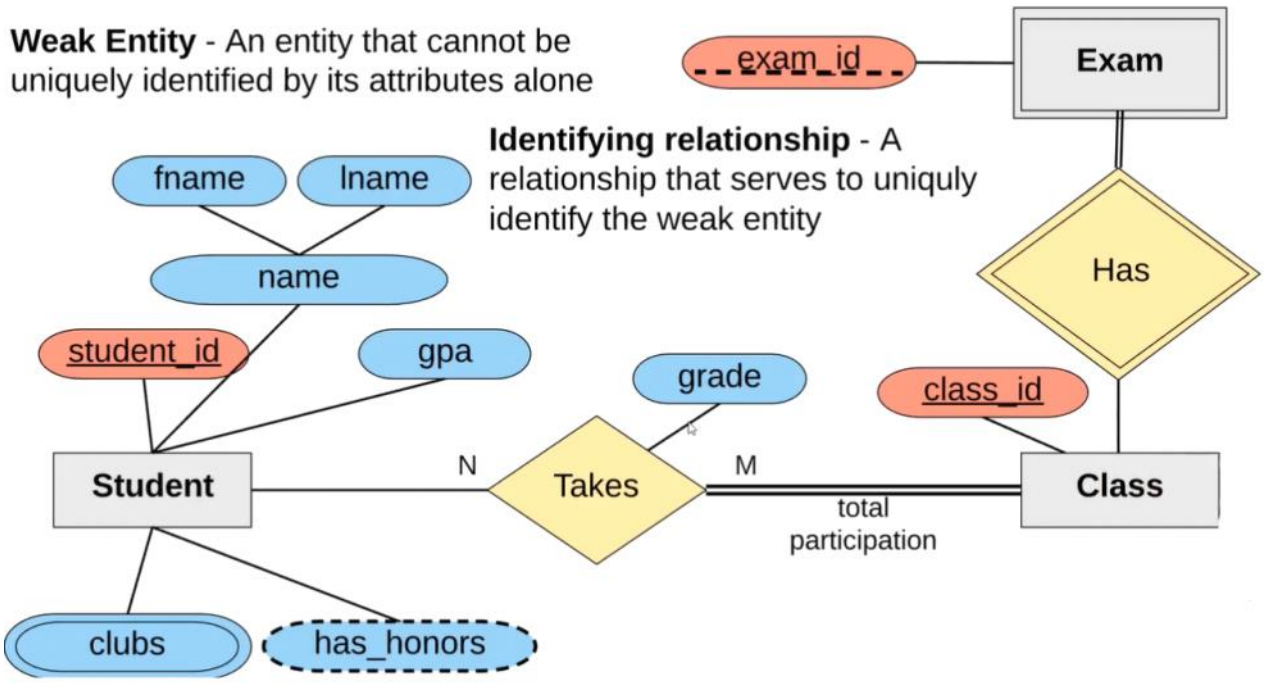
Relationship Attribute - An attribute about the relationship



Relationship Cardinality - the number of instances of an entity from a relation that can be associated with the relation

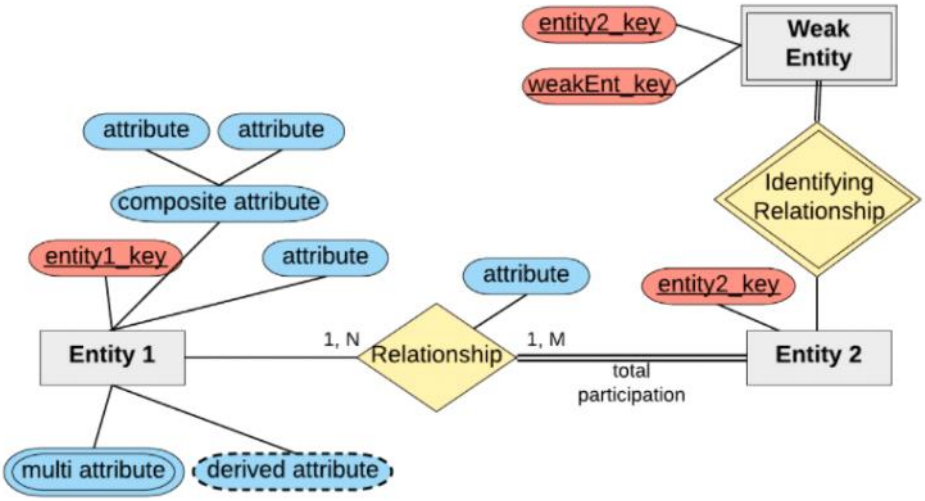
1 : 1
1 : N
N : M

Weak Entity - An entity that cannot be uniquely identified by its attributes alone

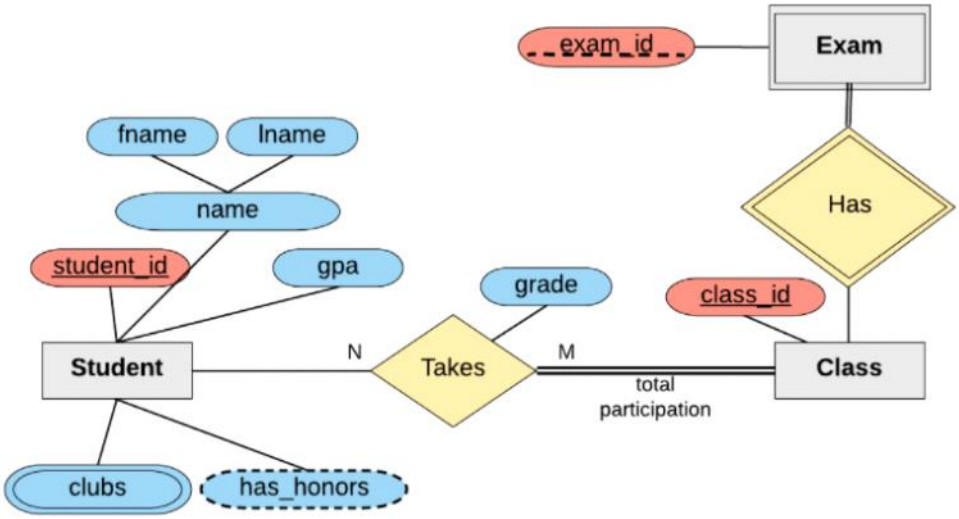


Identifying relationship - A relationship that serves to uniquly identify the weak entity

ER Diagram Template



Student Diagram



Lesson 22 (Designing an Er Diagram)

Company Data Storage Requirements
The company is organized into branches. Each branch has a unique number, a name, and a particular employee who manages it.

The company makes it's money by selling to clients. Each client has a name and a unique number to identify it.

The foundation of the company is it's employees. Each employee has a name, birthday, sex, salary and a unique number.

An employee can work for one branch at a time, and each branch will be managed by one of the employees that work there. We'll also want to keep track of when the current manager started as manager.

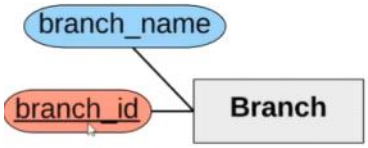
An employee can act as a supervisor for other employees at the branch, an employee may also act as the supervisor for employe es at other branches. An employee can have at most one supervisor.

A branch may handle a number of clients, with each client having a name and a unique number to identify it. A single client m ay only be handled by one branch at a time.

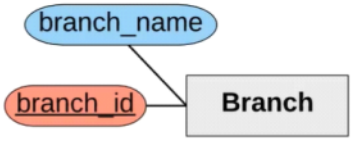
Employees can work with clients controlled by their branch to sell them stuff. If nescessary multiple employees can work with the same client. We'll want to keep track of how many dollars worth of stuff each employee sells to each client they work with.

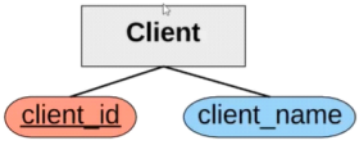
Many branches will need to work with suppliers to buy inventory. For each supplier we'll keep track of their name and the typ e of product they're selling the branch. A single supplier may supply products to multiple branches.

The company is organized into **branches**. Each branch has a unique number, and a name

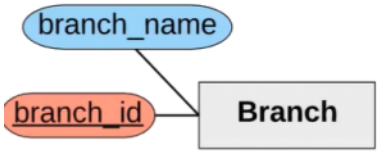


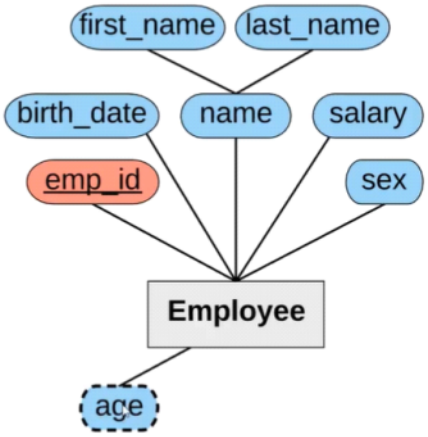
The company makes it's money by selling to **clients**. Each **client** has a name and a unique number to identify it.

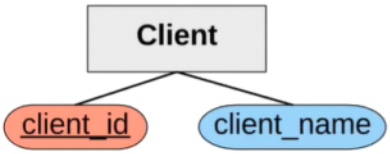




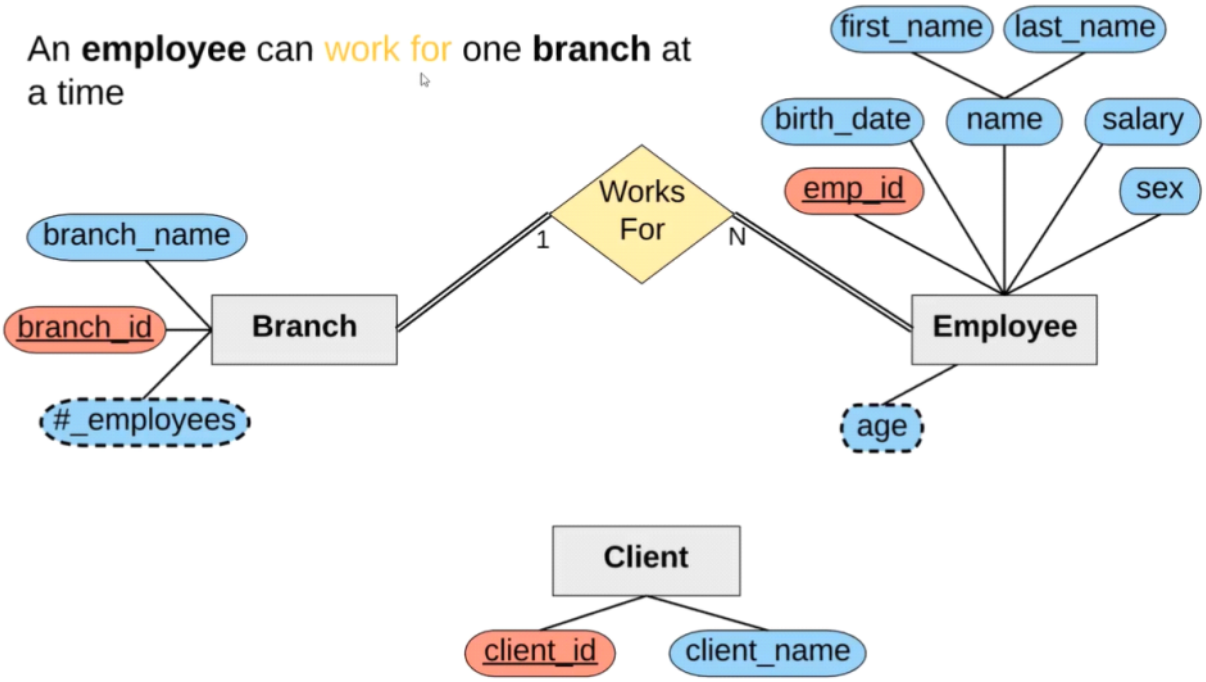
The foundation of the company is it's **employees**. Each **employee** has a name, birthday, sex, salary and a unique number to identify it.



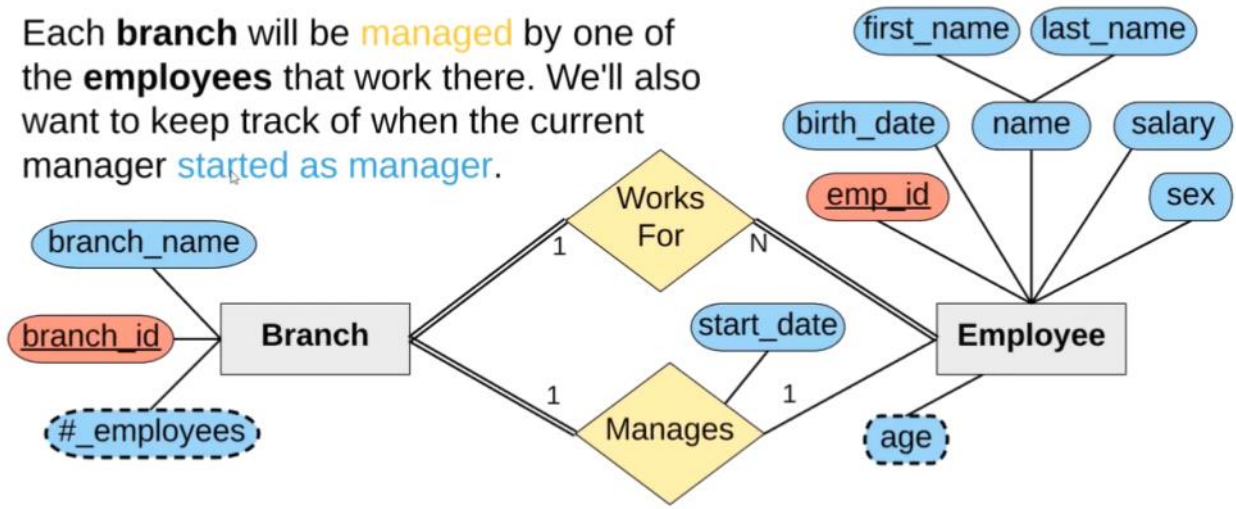


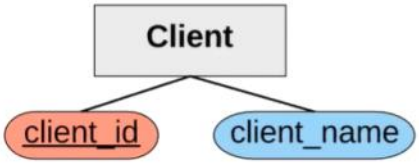


An **employee** can work for one **branch** at a time

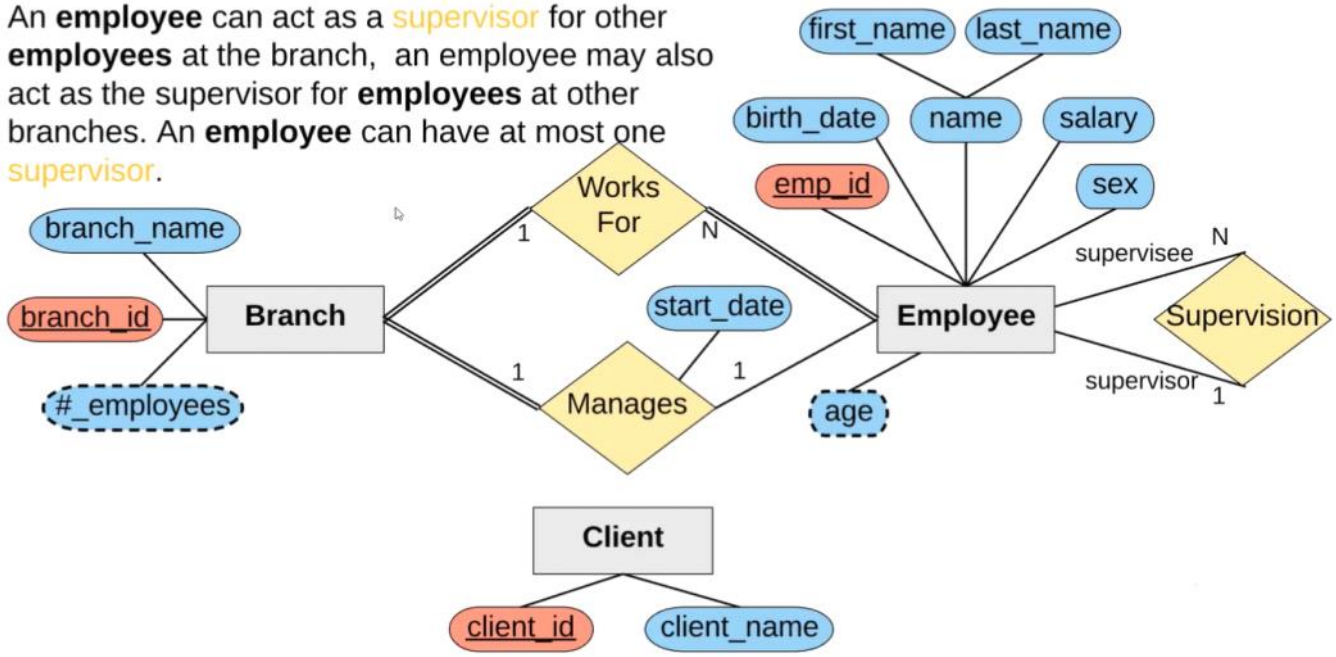


Each **branch** will be managed by one of the **employees** that work there. We'll also want to keep track of when the current manager started as manager.

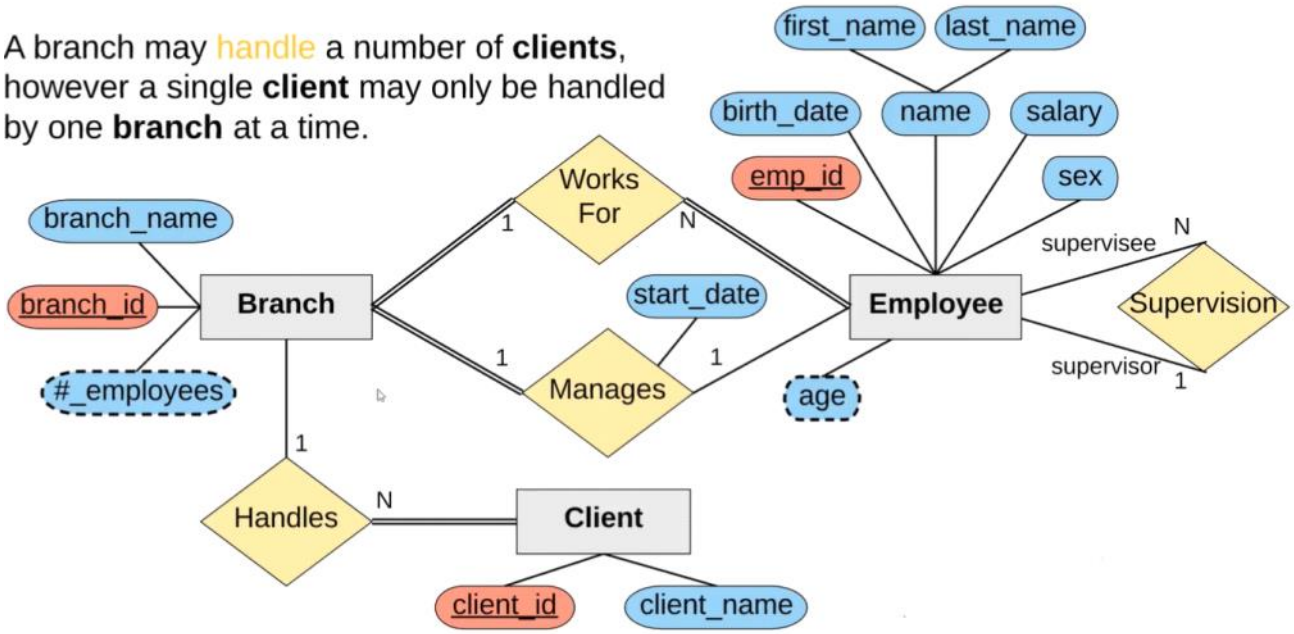




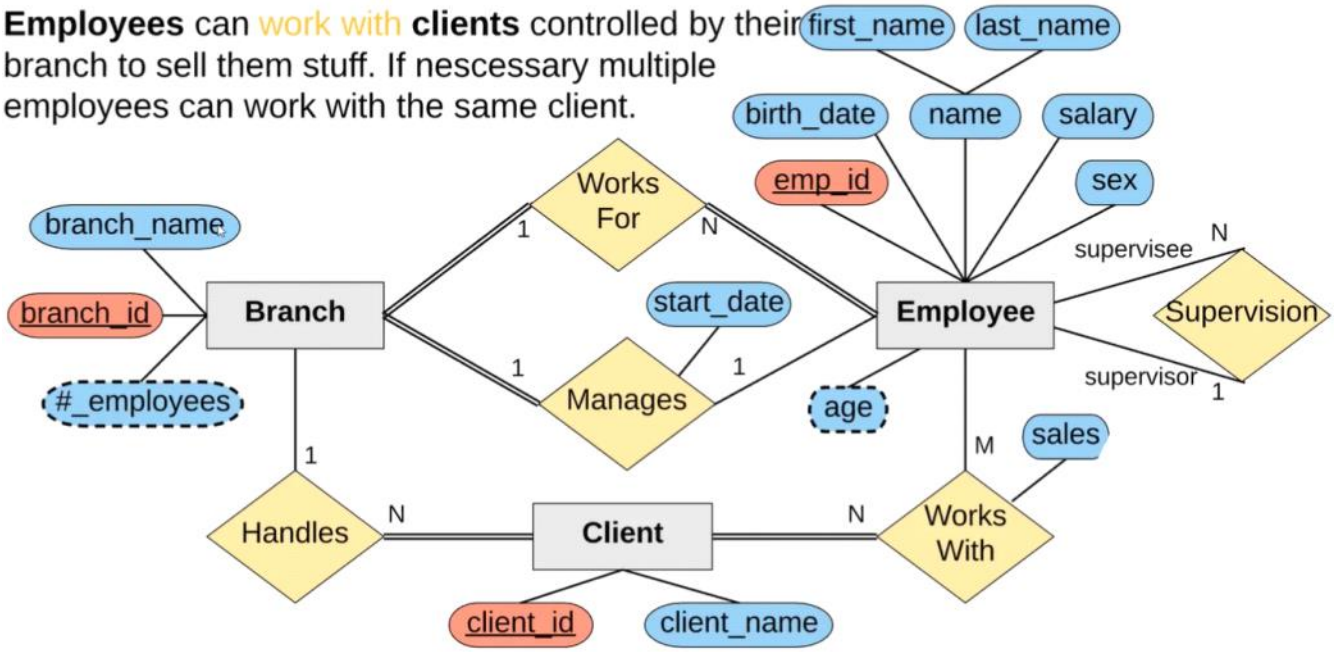
An **employee** can act as a supervisor for other **employees** at the branch, an employee may also act as the supervisor for **employees** at other branches. An **employee** can have at most one supervisor.



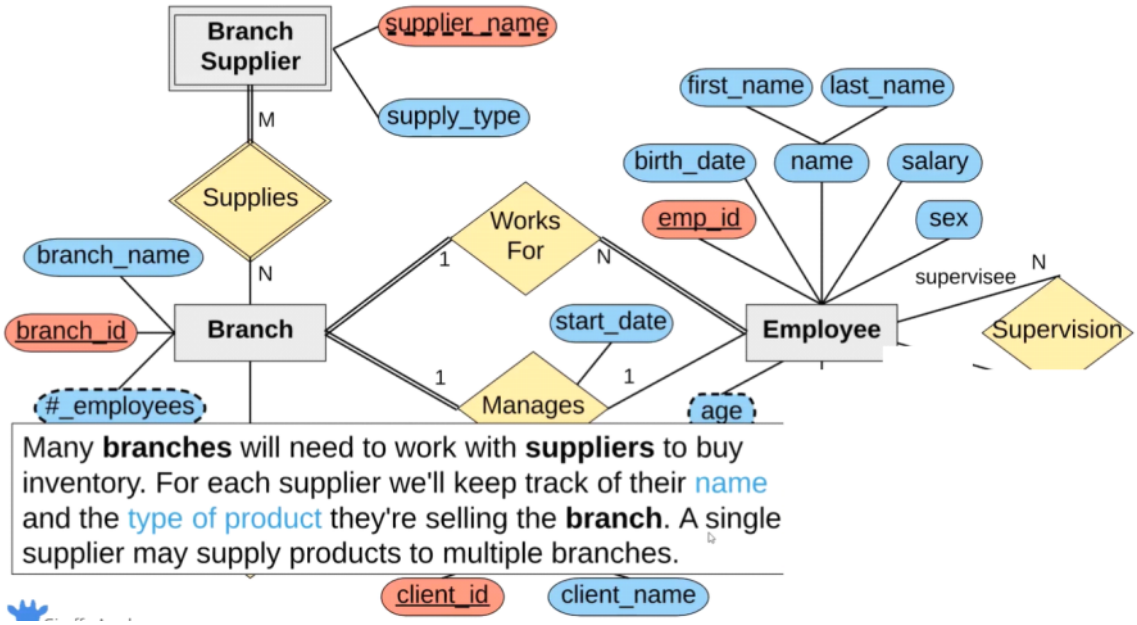
A branch may handle a number of **clients**, however a single **client** may only be handled by one **branch** at a time.



Employees can work with **clients** controlled by their branch to sell them stuff. If nescessary multiple employees can work with the same client.

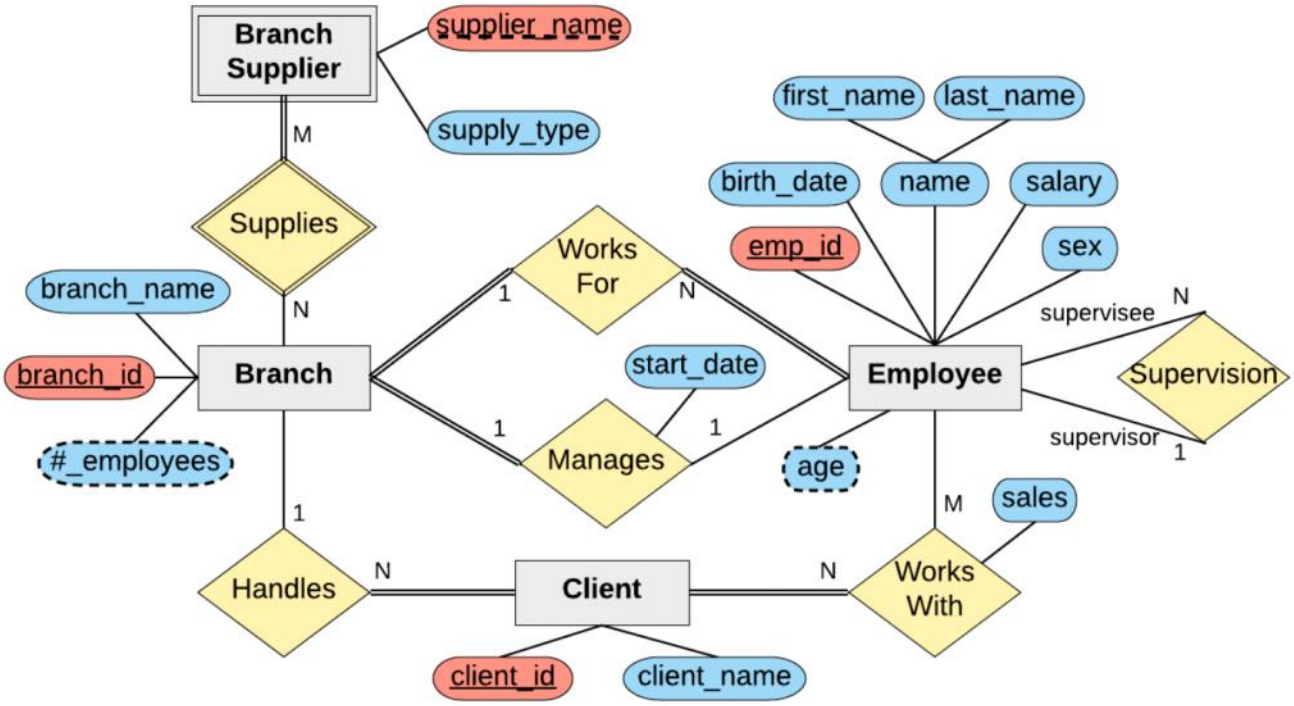


We'll want to keep track of how many dollars worth of stuff each employee sells to each client they work with.

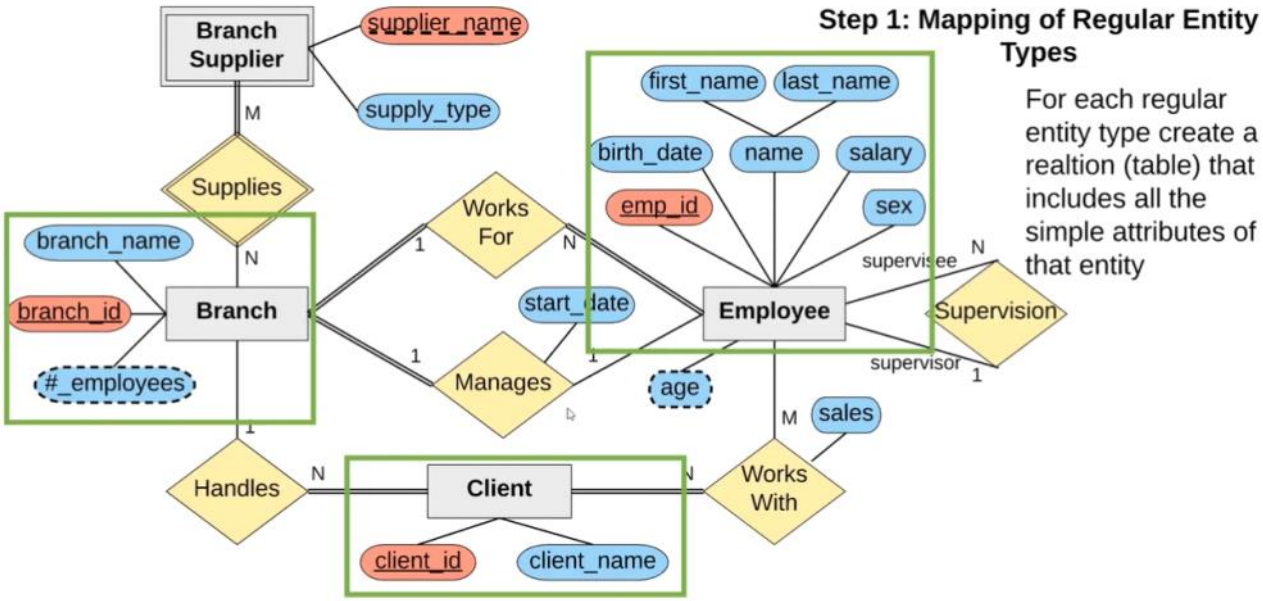


Many **branches** will need to work with **suppliers** to buy inventory. For each supplier we'll keep track of their name and the type of product they're selling the **branch**. A single supplier may supply products to multiple branches.

client_id client_name



Lesson 23 (Er Diagram Mapping)



Employee

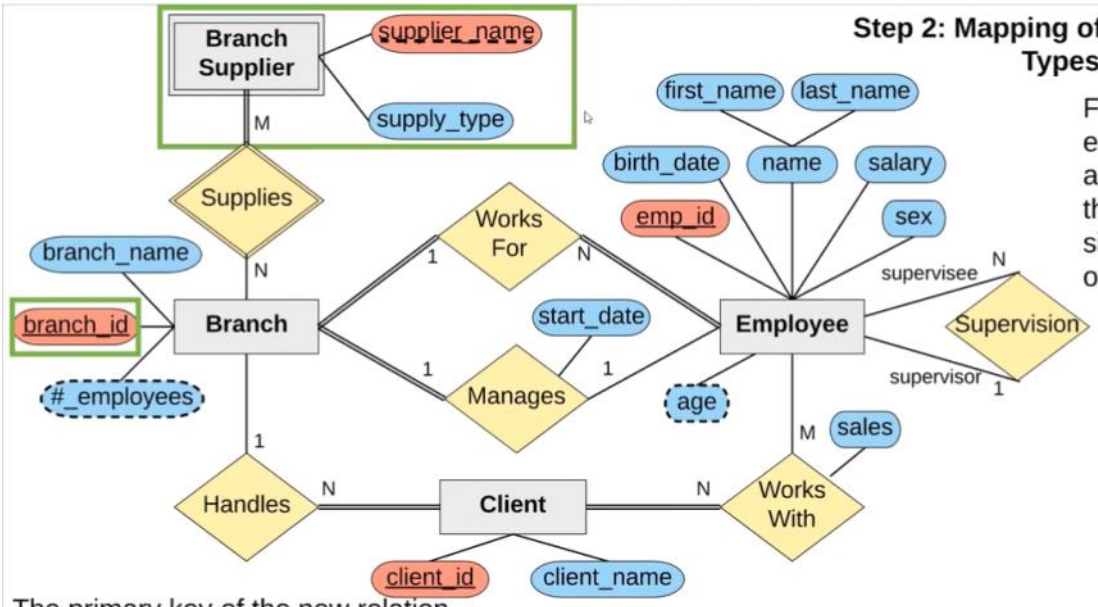
emp_id	first_name	last_name	birth_date	sex	salary

Branch

branch_id	branch_name

Client

client_id	client_name



Step 2: Mapping of Weak Entity Types

For each weak entity type create a relation (table) that includes all simple attributes of the weak entity.

The primary key of the new relation should be the partial key of the weak entity plus the primary key of its owner

Employee

emp_id	first_name	last_name	birth_date	sex	salary

Branch

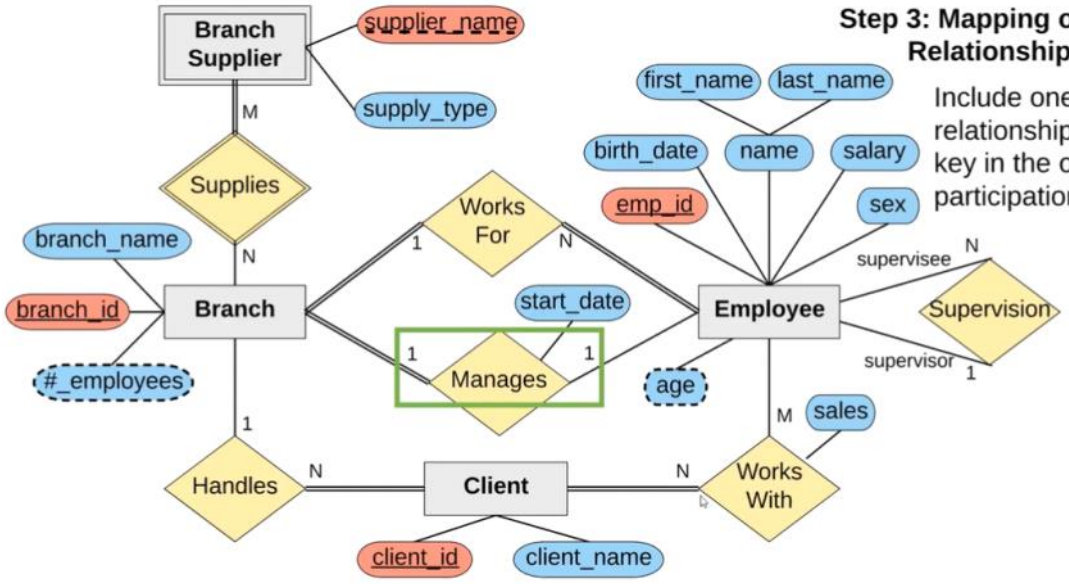
branch_id	branch_name

Client

client_id	client_name

Branch Supplier

branch_id	supplier_name	supply_type



Step 3: Mapping of Binary 1:1 Relationship Types

Include one side of the relationship as a foreign key in the other Favor total participation

Employee

emp_id	first_name	last_name	birth_date	sex	salary

Branch

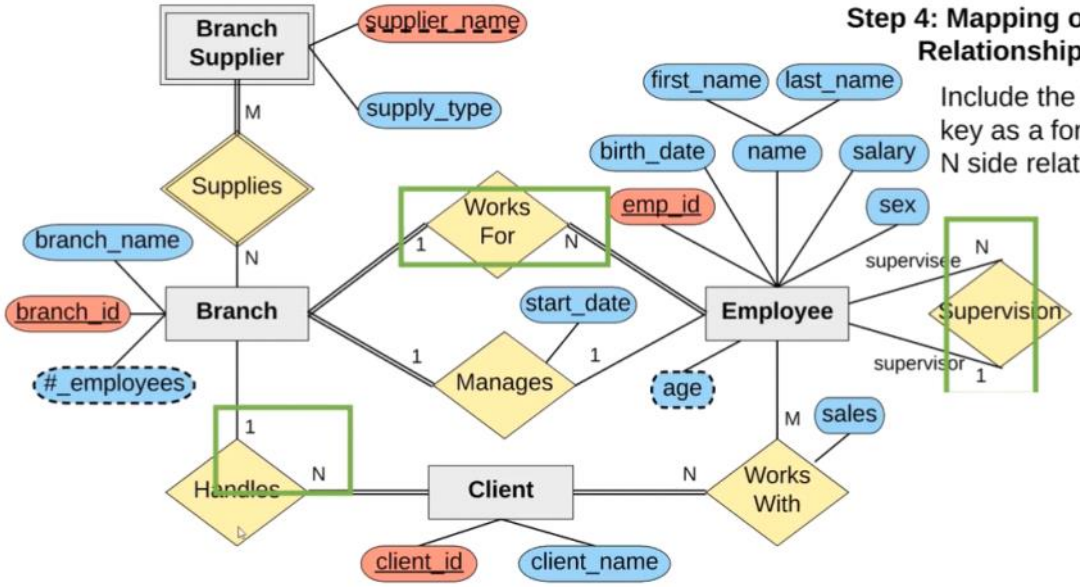
branch_id	branch_name	mgr_id	mgr_start_date

Client

client_id	client_name

Branch Supplier

branch_id	supplier_name	supply_type



Step 4: Mapping of Binary 1:N Relationship Types

Include the 1 side's primary key as a foreign key on the N side relation (table)

Employee

emp_id	first_name	last_name	birth_date	sex	salary	super_id	branch_id

Branch

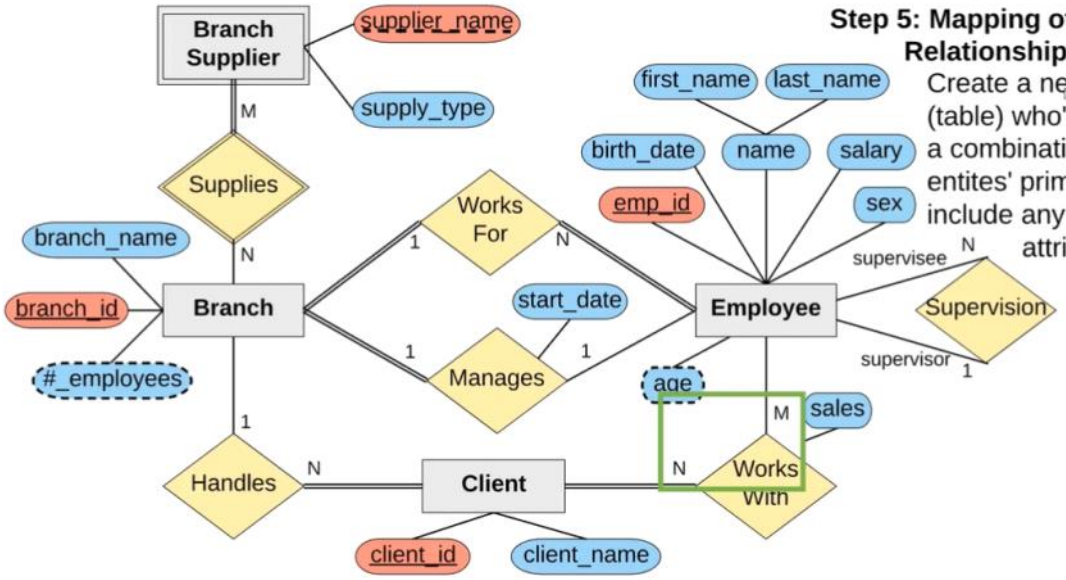
branch_id	branch_name	mgr_id	mgr_start_date

Client

client_id	client_name	branch_id

Branch Supplier

branch_id	supplier_name	supply_type



Step 5: Mapping of Binary M:N Relationship Types

Create a new relation (table) who's primary key is a combination of both entites' primary key's. Also include any relationship attribtes

Employee

emp_id	first_name	last_name	birth_date	sex	salary	super_id	branch_id

Branch

branch_id	branch_name	mgr_id	mgr_start_date

Client

client_id	client_name	branch_id

Branch Supplier

branch_id	supplier_name	supply_type

Works On

emp_id	client_id	total_sales

Company Database Schema