SQL- Company Database

1. **Create & alter table**

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

);

1. **Insert data into table**

-- 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);

1. **To access the table**

select \* from employee;

select \* from client;

1. **Examples**

**--find all employees**

SELECT \* FROM employee;

**-- find all clients**

SELECT \* from client;

**--find all employees ordered by salary**

SELECT \* from employee

ORDER BY salary;

SELECT \* from employee

ORDER BY salary DESC; **-- descending order**

**--Find all employees ordered by sex and then name**

SELECT\* FROM employee

ORDER BY sex,first\_name,last\_name;

**--Find the first five employees in the table**

SELECT \* FROM employee

LIMIT 5;

**-- Find first and last name of all employees**

SELECT first\_name, last\_name

FROM employee;

**--Find the forename and surname of all employees**

SELECT first\_name AS forename, last\_name AS surname

FROM employee;

**--Find out all the different genders**

SELECT DISTINCT sex

FROM employee;

**--Find the number of employees**

SELECT COUNT(emp\_id)

FROM employee;

**--Find the number of supervisors**

SELECT COUNT(super\_id)

FROM employee;

**--find the female employees born after 1970**

SELECT COUNT(emp\_id)

FROM employee

WHERE sex="F" AND birth\_day>'1970-01-01';

**--find the avg salary of all the employees**

SELECT AVG(salary)

FROM employee;

**--find the avg salary of all the employees who are male**

SELECT AVG(salary)

FROM employee

WHERE sex='M';

**--find the sum of salary of all employees**

SELECT SUM(salary)

FROM employee;

**--find out how many males or females are there**

**--this is what aggregation is**

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 emp\_id;

**--for client**

SELECT SUM(total\_sales), client\_id

FROM works\_with

GROUP BY client\_id;

SELECT \* FROM employee;

**-- find any clients who are in LLC**

SELECT \*

FROM client

WHERE client\_name LIKE '%LLC';

**--find any branch supplier who are in the label business**

SELECT\*

FROM branch\_supplier

WHERE supplier\_name LIKE '%label%';

**--find any employee borned in october**

SELECT\*

FROM employee

WHERE birth\_day LIKE '\_\_\_\_-10%';

**--find any clients who are schools**

SELECT \*

FROM client

WHERE client\_name LIKE '%school%';

**--find a list of employee and branch name**

SELECT first\_name

FROM employee

UNION

SELECT branch\_name

FROM branch;

**--find a list of all clients & branch suppliers names**

SELECT client\_name, branch\_id

FROM client

UNION

SELECT supplier\_name, branch\_id

FROM branch\_supplier;

**--find a list of all money spent or earned by the company**

SELECT salary

From employee

UNION

SELECT total\_sales

FROM works\_with;

**--joins**

INSERT INTO branch VALUES(4,'Buffalo',NULL,NULL)

SELECT\* FROM branch;

**--find all branches and the name of their managers**

SELECT employee.emp\_id, employee.first\_name, branch.branch\_name

FROM employee

JOIN branch

ON employee.emp\_id=branch.mgr\_id;

**--left join**

SELECT employee.emp\_id, employee.first\_name, branch.branch\_name

FROM employee

LEFT JOIN branch

ON employee.emp\_id=branch.mgr\_id;

**--right join**

SELECT employee.emp\_id, employee.first\_name, branch.branch\_name

FROM employee

RIGHT JOIN branch

ON employee.emp\_id=branch.mgr\_id;

**--Find names of all employees**

**--who have sold over 30,000 to a single client**

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>30000);

**-- Find all clients who are handled by the branch**

**--that michael scott manages**

**--assume you know michaels id**

SELECT client\_name

FROM client

WHERE branch\_id IN(

    SELECT branch\_id

    FROM employee

    WHERE first\_name="Michael");

SELECT \* FROM employee;

**--\*\*\*\*\*\*\* DELETE  \*\*\*\*\*\*\*\*\***

**--on delete set null**

DELETE FROM employee

WHERE emp\_id=102;

SELECT \* FROM employee;

**#lets see what happens to branch**

SELECT \* FROM branch;

**#we get mgr\_id as null bcoz we deleted**

**#the 102 emp\_id entry from the table which**

**#affected the branch table as well**

**#so manager id which is a foreign key is set**

**#to null.**

**--on delete set cascade**

**-- in this we delete the entire row**

**--based on the condition**

DELETE FROM branch

WHERE branch\_id=2;

SELECT \* FROM branch\_supplier;

**#here we dont get any data having branch id=2**

**#as we deleted the same from branch\**

**#primary key should not have null values, in**

**#that case use cascade while using null**

**--triggers**

**#triggers are nothing but block of sql code**

**#which will define a certain action**

**#that should happen when a certain operation gets**

**#performed on the database**

**#if anything gets deleted from the table then**

**#insert the mentioned information**

CREATE TABLE trigger\_test(

    message VARCHAR(100)

);

DELIMETER $$

CREATE

    TRIGGER my\_trigger BEFORE INSERT

    ON employee

    FOR EACH ROW BEGIN

        INSERT INTO trigger\_test VALUES('added new employee');

    END$$

DELIMETER;

**--to define a trigger give it a name before something gets inserted**

**--on the employee table**

**--we want to insert into trigger test**

**-- before anything gets inserted into employee table**

**-- we perform the operation which is in trigger**

INSERT INTO employee

VALUES(109,'Oscar','Martinez','1968-02-19','M',69000,106,3);

SELECT \* FROM employee

SELECT \* FROM trigger\_test;

**--we can also create triggers for update and delete**

**--we can also drop the triggers**

**5. ER Diagram introduction**

**Entity Relationship**

**5.1 Entity**

-an object we want to model & store information about

-while designing a database we need to design a database schema

- we can use ER as a middleware between database and database schemas

- it is used for relationship model

**5.2 Attribute**

-specific pieces of information about an entity

**5.3 primary key**

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

**5.4 composite attribute**

- an attribute that can be broken up into sub-attributes

**5.5 multivalued attribute**

-An attribute that can have more than one value

**5.6 Derived attribute**

-An attribute that can be derived from other attributes

-we can have multiple entities

-relationships

-defines a relationship between two entities

**5.7 total participation**

-all members must participate in the relationship

**5.8 relationship attribute**

-an attribute about the relationship

**5.9 relationship cardinality**

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

**5.10 weak entity**

-an entity that cannot be uniquely identified by its attributes alone

**5.11 identifying relationship**

-a relationship that serves to uniquely identify the weak entity