Daiki Itoh

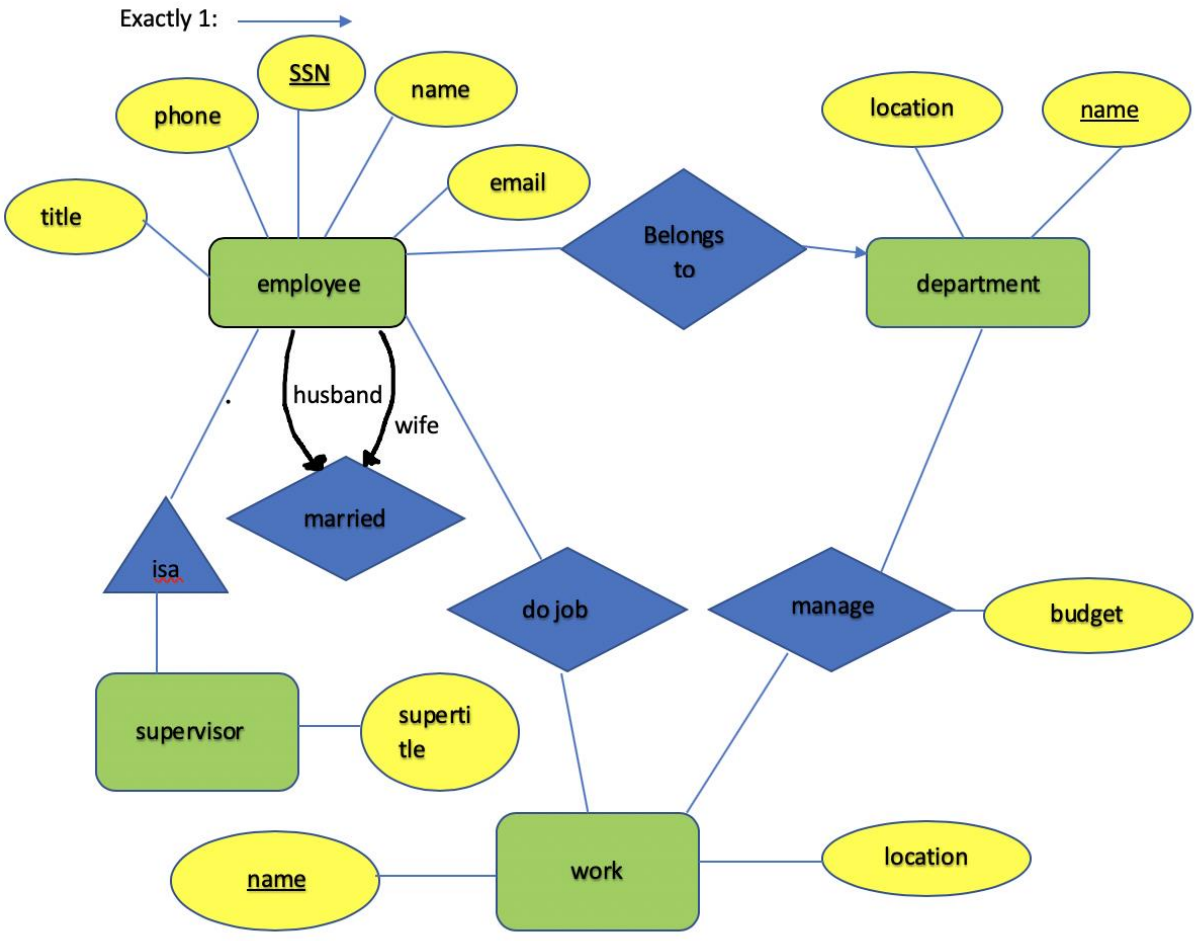
1. **Description**

   It is a database to store company's internal information about employees, departments, and its work. The example of work is projects, presentations, and so on.

   Functionalities:

   1. The user can add/delete an employee to employee table.
   2. The user can add/delete a department to department table.
   3. The user can add/delete a work to work table.
   4. The user can manage information about the company.
   5. The user can have relation of each employee to its department and work (vice versa).

2. **ER**



3. **Relations**

   Department (<u>name</u>, location)

Employee (ssn, name, title, phone, email, supertitle) <- using nulls to include supervisor
Work (name, location)
Married (husband, wife)
BelongTo (employee, dpt)
Manage (dpt, work, budget)
DoJob (employee, work)

4. **SQL**

```
-- ============ Convert your E/R to relations

-- ============ CREATE TABLE's
  -- Re-do your CREATE TABLE to have the constraints
  -- (at least one for each kind of constraints below)
    -- primary key
    -- foreign key
    -- attribute constraint
    -- tuple constraint

-- Create a table Department setting name as primary key
DROP TABLE Department;
CREATE TABLE Department(
   name VARCHAR(20) PRIMARY KEY,
   location VARCHAR(50) NOT NULL
);

-- Create a table Employee setting ssn as primary key
DROP TABLE Employee;
CREATE TABLE Employee(
   ssn VARCHAR(20) PRIMARY KEY,
   name VARCHAR(30) NOT NULL,
   title VARCHAR(20) NOT NULL,
   email VARCHAR(40) NOT NULL,
   phone VARCHAR(16) NOT NULL,
   supertitle VARCHAR(20),
   CHECK (title NOT LIKE '%Manager%' OR (supertitle IS NOT NULL AND supertitle =
'Supervisor'))
);

-- Create a table Work setting name as primary key
DROP TABLE Work;
CREATE TABLE Work(
   name VARCHAR(40) PRIMARY KEY,
   location VARCHAR(50) NOT NULL
```

```
);

-- Create a table Married setting husband & 2 as primary key
DROP TABLE Married;
CREATE TABLE Married(
   husband VARCHAR(20) REFERENCES Employee(ssn)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
   wife VARCHAR(20) REFERENCES Employee(ssn)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
   PRIMARY KEY(husband, wife)
);

-- Create a table BelongTo setting ssn as primary key
DROP TABLE BelongTo;
CREATE TABLE BelongTo(
   employee VARCHAR(20) PRIMARY KEY,
   dpt VARCHAR(20) NOT NULL,
   FOREIGN KEY (employee)
    REFERENCES Employee(ssn)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
   FOREIGN KEY (dpt)
    REFERENCES Department(name)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- Create a table Manage
DROP TABLE Manage;
CREATE TABLE Manage(
   dpt VARCHAR(20),
   work VARCHAR(40),
   budget DECIMAL(7,2) NOT NULL CHECK (budget > 0.0),
   FOREIGN KEY (dpt)
    REFERENCES Department(name)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
   FOREIGN KEY (work)
    REFERENCES Work(name)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    PRIMARY KEY(dpt, work)
```

```
);

-- Create a table DoJob
DROP TABLE DoJob;
CREATE TABLE DoJob(
   employee VARCHAR(20),
   work VARCHAR(40),
   FOREIGN KEY (employee)
     REFERENCES Employee(ssn)
     ON DELETE CASCADE
     ON UPDATE CASCADE,
   FOREIGN KEY (work)
     REFERENCES Work(name)
     ON DELETE CASCADE
     ON UPDATE CASCADE,
     PRIMARY KEY(employee, work)
);

   -- the following two will be covered soon.
    -- functional dependencies for each table
    -- Indicate 3NF or BCNF or 4NF for each table

-- ============ INSERT data


-- ############# Insert into Department
INSERT INTO Department VALUES ('IT', '123 North Street Jonesboro, AR');
INSERT INTO Department VALUES ('Marketing', '456 South Street Jonesboro, AR');
INSERT INTO Department VALUES ('Finance', '456 South Street Jonesboro, AR');
INSERT INTO Department VALUES ('Accounting', '456 South Street Jonesboro, AR');
INSERT INTO Department VALUES ('Sales', '456 South Street Jonesboro, AR');
INSERT INTO Department VALUES ('HR', '456 South Street Jonesboro, AR');


-- ############# Insert into employee
-- IT
INSERT INTO Employee VALUES('123-456-7890', 'Daiki Itoh', 'Software Developer',
'ditoh@dev.com', '8701231234');
INSERT INTO Employee VALUES('923-453-2891', 'Rachael Davis', 'Software Developer',
'rdavis@dev.com', '8701188930');
INSERT INTO Employee VALUES('625-156-6890', 'Joseph Adams', 'Software Developer',
'jadams@dev.com', '8401031212', 'Supervisor');

-- Marketing
```

```
INSERT INTO Employee VALUES('163-458-1265', 'Thomas Kim', 'Marketing Specialist',
'tkim@dev.com', '8700233765');
INSERT INTO Employee VALUES('173-156-8475', 'Elizabeth Kim', 'Marketing Specialist',
'ekim@dev.com', '8701231234');
INSERT INTO Employee VALUES('756-456-6590', 'David Causey', 'Marketing Manager',
'dcausey@dev.com', '8701261230', 'Supervisor');

-- Finance
INSERT INTO Employee VALUES('103-456-6847', 'Jerry Branch', 'Finance Specialist',
'jbrance@dev.com', '5701131237');
INSERT INTO Employee VALUES('109-096-6890', 'Courtney Jenness', 'Finance
Specialist', 'cjenness@dev.com', '340124123');
INSERT INTO Employee VALUES('173-156-8375', 'Elizabeth Bobo', 'Finance Manager',
'ebobo@dev.com', '8700234534', 'Supervisor');

-- Accounting
INSERT INTO Employee VALUES('153-856-7190', 'John Smith', 'Accountant',
'jsmith@dev.com', '8701336234');
INSERT INTO Employee VALUES('596-456-7845', 'Paige Causey', 'Accountant',
'pcausey@dev.com', '8703453263');
INSERT INTO Employee VALUES('673-956-2475', 'Elizabeth Kim', 'Accounting Manager',
'ekim2@dev.com', '8704858467', 'Supervisor');

-- Sales
INSERT INTO Employee VALUES('173-136-0245', 'Matthew Liu', 'Sales Representative',
'mliu@dev.com', '9801331234');
INSERT INTO Employee VALUES('373-556-2375', 'Tom Garcia', 'Sales Representative',
'tgarcia@dev.com', '8501931284');
INSERT INTO Employee VALUES('773-956-1377', 'Emma Jonas', 'Sales Manager',
'ejonas@dev.com', '8701244334', 'Supervisor');

-- HR
INSERT INTO Employee VALUES('173-156-1245', 'Jacob Gupta', 'HR Specialist',
'jgupta@dev.com', '6101251834');
INSERT INTO Employee VALUES('973-256-1358', 'Daniel Elrod', 'HR Specialist',
'delrod@dev.com', '8701336514');
INSERT INTO Employee VALUES('673-056-8005', 'Zack Mondy', 'HR Manager',
'zmondy@dev.com', '8700091265', 'Supervisor');


-- ############ insert into Work
INSERT INTO Work VALUES('Develop calender app', 'Jonesboro');
INSERT INTO Work VALUES('Develop iOS app', 'Jonesboro');
INSERT INTO Work VALUES('Advertise calender app', 'Jonesboro');
```

```
INSERT INTO Work VALUES('Sell calender app', 'Jonesboro');
INSERT INTO Work VALUES('Calculate budget', 'Jonesboro');
INSERT INTO Work VALUES('Recruit new employees', 'Jonesboro');


-- ############ insert into Married
INSERT INTO Married VALUES('163-458-1265', '173-156-8475');


-- ############ insert into BelongTo

-- IT
INSERT INTO BelongTo VALUES('123-456-7890', 'IT');
INSERT INTO BelongTo VALUES('923-453-2891', 'IT');
INSERT INTO BelongTo VALUES('625-156-6890', 'IT');

-- Marketing
INSERT INTO BelongTo VALUES('163-458-1265', 'Marketing');
INSERT INTO BelongTo VALUES('173-156-8475', 'Marketing');
INSERT INTO BelongTo VALUES('756-456-6590', 'Marketing');

-- Finance
INSERT INTO BelongTo VALUES('103-456-6847', 'Finance');
INSERT INTO BelongTo VALUES('109-096-6890', 'Finance');
INSERT INTO BelongTo VALUES('173-156-8375', 'Finance');

-- Accounting
INSERT INTO BelongTo VALUES('153-856-7190', 'Accounting');
INSERT INTO BelongTo VALUES('596-456-7845', 'Accounting');
INSERT INTO BelongTo VALUES('673-956-2475', 'Accounting');

-- Sales
INSERT INTO BelongTo VALUES('173-136-0245', 'Sales');
INSERT INTO BelongTo VALUES('373-556-2375', 'Sales');
INSERT INTO BelongTo VALUES('773-956-1377', 'Sales');

-- HR
INSERT INTO BelongTo VALUES('173-156-1245', 'HR');
INSERT INTO BelongTo VALUES('973-256-1358', 'HR');
INSERT INTO BelongTo VALUES('673-056-8005', 'HR');


-- ############ insert into Manage
INSERT INTO Manage VALUES('IT', 'Develop calender app', 20000.00);
```

```
INSERT INTO Manage VALUES('IT', 'Develop iOS app', 10500.05);
INSERT INTO Manage VALUES('Marketing', 'Advertise calender app', 15000.00);
INSERT INTO Manage VALUES('Sales', 'Sell calender app', 10000.50);
INSERT INTO Manage VALUES('Finance', 'Calculate budget', 9000.82);
INSERT INTO Manage VALUES('Accounting', 'Calculate budget', 9000.82);
INSERT INTO Manage VALUES('HR', 'Recruit new employees', 5000.00);


-- ############ insert into DoJob
-- IT
INSERT INTO DoJob VALUES('123-456-7890', 'Develop calender app');
INSERT INTO DoJob VALUES('923-453-2891', 'Develop calender app');
INSERT INTO DoJob VALUES('625-156-6890', 'Develop iOS app');

-- Marketing
INSERT INTO DoJob VALUES('163-458-1265', 'Advertise calender app');
INSERT INTO DoJob VALUES('756-456-6590', 'Advertise calender app');

-- Finance
INSERT INTO DoJob VALUES('103-456-6847', 'Calculate budget');
INSERT INTO DoJob VALUES('109-096-6890', 'Calculate budget');
INSERT INTO DoJob VALUES('173-156-8375', 'Calculate budget');

-- Accounting
INSERT INTO DoJob VALUES('153-856-7190', 'Calculate budget');
INSERT INTO DoJob VALUES('596-456-7845', 'Calculate budget');

-- Sales
INSERT INTO DoJob VALUES('173-136-0245', 'Sell calender app');
INSERT INTO DoJob VALUES('373-556-2375', 'Sell calender app');
INSERT INTO DoJob VALUES('773-956-1377', 'Sell calender app');

-- HR
INSERT INTO DoJob VALUES('173-156-1245', 'Recruit new employees');


-- ============ Queris

-- ============ 8 simple queries (similar to the examples below)
--          operators includes (and,or,not)
--          patterns

-- SELECT ... FROM ... WHERE
```

```
-- 1. Find all employees.
SELECT * FROM Employee;

-- 2. Find all emplyees that are supervisor
SELECT * FROM Employee WHERE supertitle = 'Supervisor';

-- 3. Find all the departments that are on 456 Sounth Street
SELECT * FROM Department WHERE location LIKE '%456 South Street%';

-- 4. Find all the work that has specific location
SELECT * FROM WORK WHERE location is NOT NULL;

-- 5. Find all the work managed by Finance AND Accounting
SELECT * FROM Manage WHERE dpt = 'Finance' OR dpt = 'Accounting';

-- 6. Find the employees(ssn) that belong to IT department & Finance
SELECT * FROM BelongTo WHERE dpt = 'IT' OR dpt = 'Finance';

-- 7. Find all the employees who are software developer
SELECT * FROM Employee WHERE title = 'Software Developer';

-- 8. Find all the employees who are software developer but not supervisor
 SELECT * FROM Employee WHERE title = 'Software Developer' AND supertitle is
NULL;

-- ============= 6 Multirelation queries (two or more relations
--                          in the FROM-clause)
-- (similar to the examples below)

-- 9. Find department, location work they manage and its budget
SELECT name, location, work, budget FROM Department, Manage WHERE name =
dpt;

-- 10. Find the names of employee who belong to Finance department
SELECT name FROM Employee, BelongTo WHERE dpt = 'Finance' AND ssn =
employee;

-- using operators: and or not

-- 11. Find the location and name work managed by IT dpt
SELECT name, location FROM Work, Manage WHERE dpt = 'IT' AND name = work;

-- 12. Find names of employee that work to sell calender app and who is not
supervisor
```

```
  SELECT name FROM Employee, DoJob WHERE ssn = employee AND supertitle is NULL
AND work = 'Sell calender app';

  -- 13. Find the title of employees who are married
  SELECT title FROM Employee, Married WHERE ssn = husband OR ssn = wife;

  -- 14. Find names of employees that work to advertise calender app and who is
supervisor and married
  SELECT name FROM Employee e, DoJob d, Married m WHERE ssn = employee AND
(d.employee = m.husband OR d.employee = m.wife) AND d.work = 'Advertise calender
app';


 -- ============= 6 Subqueries like below with your questions and SQL's

  -- Queries with subquery

  --15. has subquery in the clause "FROM"  display name of work and IT work
  SELECT name, itWork FROM Work, (SELECT work FROM Manage WHERE dpt = 'IT')
itWork;

  --16. has subquery with keyword "IN" display employee who belongs to HR
  SELECT name FROM Employee WHERE ssn IN (SELECT employee FROM BelongTo
WHERE dpt = 'HR');

  --17. has subquery with keyword "EXISTS" display employee where title is unique
  SELECT * FROM Employee e1 WHERE NOT EXISTS(SELECT * FROM Employee WHERE
e1.title = title AND e1.name <> name);

  --18. has subquery with keyword "ANY" display any budget information where dpt is
IT
  SELECT * FROM Manage WHERE budget = ANY(SELECT budget FROM Manage WHERE
dpt = 'IT');
  --19. has subquery with keyword "ALL" dislay the highest budget information
  SELECT * FROM Manage WHERE budget >= ALL(SELECT budget FROM Manage);

  --20. has something else display if there exists an employee whose title is not unique
and is supervisor
  SELECT * FROM Employee e1 WHERE EXISTS(SELECT * FROM Employee WHERE
e1.title = title AND e1.name <> name) AND e1.supertitle = 'Supervisor';


 -- ==============  5 SQL-statements using union, intersect, difference (except) with
your questions and SQL's
```

```
  -- 21. The employees(ssn) work in IT or Marketing
  (SELECT * From BelongTo WHERE dpt = 'IT') UNION (SELECT * From BelongTo WHERE
dpt = 'Marketing');

  -- 22. The work managed by both Finance and Accounting
  (SELECT work From Manage WHERE dpt = 'Finance') INTERSECT (SELECT work From
Manage WHERE dpt = 'Accounting');

  -- 23. The employees who are HR specialist but not supervisor
  (SELECT name FROM Employee WHERE title = 'HR Specialist') EXCEPT (SELECT name
FROM Employee WHERE title = 'HR Specialist' AND supertitle = 'Supervisor');

  -- 24. all the jobs managed by IT department but not done by supervisor
  (SELECT work from Manage WHERE dpt = 'IT') EXCEPT ALL (SELECT work FROM
DoJob, Employee WHERE ssn = employee AND supertitle = 'Supervisor');

  -- 25. all jobs which are managed by all the departments but involve supervisor
  (SELECT work FROM DoJob) EXCEPT ALL (SELECT work FROM DoJob, Employee
WHERE ssn = employee AND supertitle is NULL);


 -- ===============5 SQL-statements using Join
================================
  -- using CROSS JOIN, NATURAL JOIN, THETA JOIN (INNER JOIN)

  -- 26. Find employees who belong to IT dpt
  SELECT name FROM Employee JOIN BelongTo ON ssn = employee AND dpt = 'IT';

  -- 27. Find all the combination of work and employee
  SELECT * FROM work CROSS JOIN Employee;

  -- 28. Find the employee(ssn) work that he or she does and department to belong to
  SELECT * FROM DoJob NATURAL JOIN BelongTo;

  -- 29. Find an employee that belongs to Accounting dpt and calculates budget
  SELECT name FROM Employee JOIN DoJob ON ssn = employee AND work = 'Calculate
budget';

  -- 30. Find all departments, location, work and budget and the budget is greater than
$10000.00
  SELECT name, location, work, budget FROM Department JOIN Manage ON name =
dpt AND budget >= 10000.00;
```

```
-- ============================== OUTER JOIN
-- using LEFT, RIGHT, FULL OUTER JOIN

-- 31. List employees(name) with title and their work.
SELECT name, title, work FROM Employee LEFT OUTER JOIN DoJob ON ssn =
employee;

-- 32. List name of department, location, work budget with budget >= 10500.05 but
use right outer join to show the null value for those with < 10500.05
SELECT name, location, work, budget FROM Department RIGHT OUTER JOIN Manage
ON name = dpt AND budget >= 10500.05;

-- 33. List employees(name) with title and their work but the title should be software
developer. Use full outer join to separate the works with software developer title and
others
SELECT name, title, work FROM Employee FULL OUTER JOIN DoJob ON ssn =
employee AND title = 'Software Developer';

-- 34. List all married couples with job that husband does with right oughter join to
show all other work
SELECT * FROM Married RIGHT OUTER JOIN DoJob ON husband = employee;

-- 35. List all employees(ssn), dpt and location where the location is on 456 South
Street using left outer join
SELECT employee, dpt, location FROM belongto LEFT OUTER JOIN Department ON
dpt = name AND location LIKE '%456 South Street%';

-- ============ Queris
-- continue from Phase II
-- ============================== Aggregate Functions
-- MAX, MIN, SUM, AVG, COUNT

-- using GROUP BY

-- using HAVING

-- 31. average budget of all the work going on now by department
   SELECT dpt, AVG(budget) FROM manage GROUP BY dpt;
-- 32. total budget of work by dpt that is greater than $10000.00
   SELECT dpt, SUM(budget) FROM Manage GROUP BY dpt HAVING SUM(budget)
   > 10000.00;
-- 33. maximum price of budget
```

```
    SELECT MAX(budget) FROM Manage;
  -- 34. minimum price of budget
    SELECT MIN(budget) FROM Manage;
  -- 35. dpt that has more than or equal to 2 on going project
    SELECT dpt FROM Manage GROUP BY dpt HAVING COUNT(dpt) >= 2;

 -- ============================== Database Modification
==============================

  -- 36. Insert
    INSERT INTO Employee VALUES('123-000-0001', 'John Causey', 'HR Specialist',
'jcausey@dev.com', '8701111111');

  -- 37. Insert into a table from subquery
    INSERT INTO Married (SELECT h.ssn, w.ssn FROM Employee h, Employee w WHERE
h.ssn = '756-456-6590' AND w.ssn = '596-456-7845');

  -- 38. Delete
    DELETE FROM Employee WHERE ssn = '123-000-0001';

  -- 39. Delete
    DELETE FROM Employee e WHERE EXISTS(SELECT husband, wife FROM Married
    WHERE e.ssn = husband OR e.ssn = wife);

  -- 40. Update
    UPDATE Work SET location = 'Jonesboro' WHERE location IS NULL;

  -- 41. Update
    UPDATE Manage SET budget = 10000.00 WHERE budget > 10000.00;

 -- ============================== View ==============================
  -- 42. create view low budget work
    CREATE VIEW LowBWork(dpt, work, budget) AS SELECT dpt, work, budget FROM
Manage WHERE budget <= 5000.00;

 -- ============================== PSM ==============================
   -- 44. one PSM
    CREATE OR REPLACE FUNCTION UpdateB()
    RETURNS trigger
    AS $$
      BEGIN
        UPDATE Manage SET budget = OLD.budget + 5000.00
        WHERE dpt = OLD.dpt AND work = OLD.work;
        RETURN NEW;
```

```
    END;
  $$ LANGUAGE plpgsql;

-- 43. one trigger Maximum budget update is $5000
  CREATE TRIGGER BTrig
    AFTER UPDATE OF budget ON Manage
    FOR EACH ROW
    WHEN(OLD.budget + 5000.00 < NEW.budget)
    EXECUTE PROCEDURE UpdateB();
```

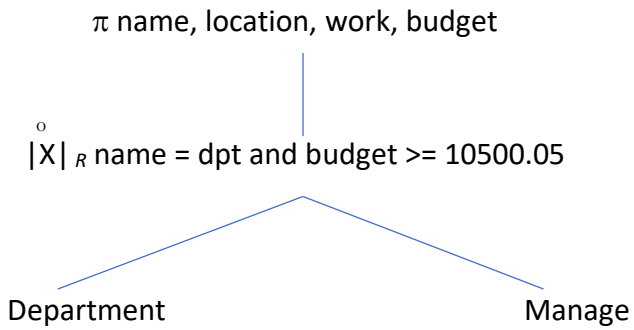
5. **Relational algebra**

32. SELECT name, location, work, budget FROM Department RIGHT OUTER JOIN Manage ON name = dpt AND budget >= 10500.05;

π name, location, work, budget (Department $|X|_R$ name = dpt and budget >= 10500.05 Manage)

Please forgive the right outer join sign. This is the best I could do.

6. **Relational algebra tree**



7. **Functional dependency & Normalization**

Employee(<u>ssn</u>, name, title, email, phone, supertitle)

ssn -> name email phone title
title -> supertitle

: 3NF

Department(<u>name</u>, location)

name -> location: BCNF

Work(name, location)

Name -> location: BCNF

Married(husband, wife)

No FD : BCNF

BelongTo(employee, dpt)

Employee -> dpt: BCNF

Manage(dpt, work, budget)

dpt, work -> budget : BCNF

DoJob(employee, work)

No FD: BCNF

8. **Interface**

http://147.97.156.240/~daiki.itoh/input.php

SSN	Name	Title	Email	Phone	Supertitle
123-456-7890	Daiki Itoh	Software Developer	ditoh@dev.com	8701231234	