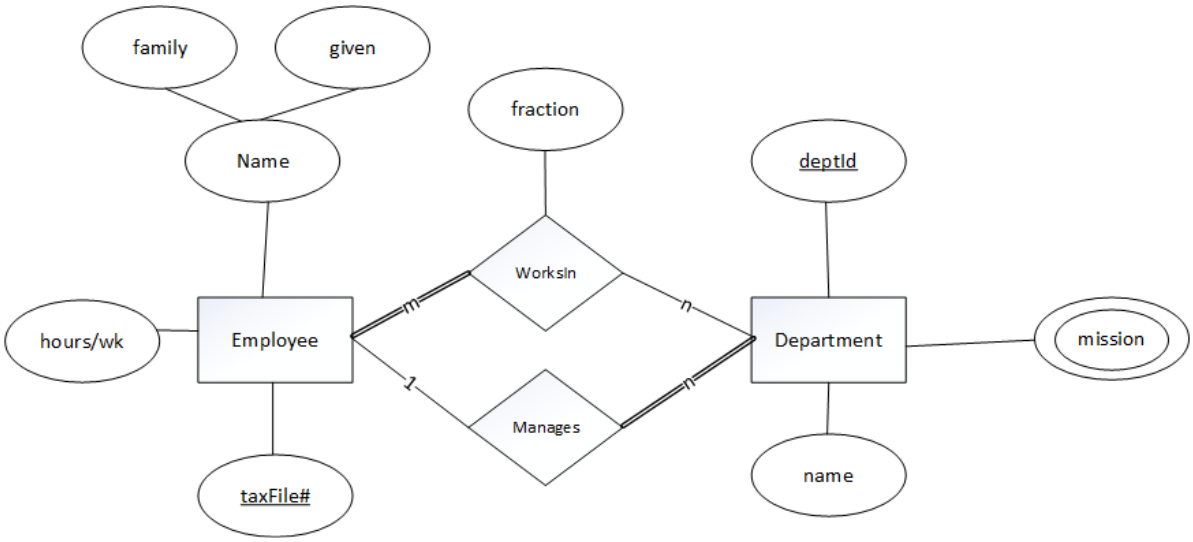
COMP9311 Lab02

Background

We wish to build a simple database for a company which has a number of departments. Each department has a manager and a mission statement, which is defined by a number of key words (e.g. commitment, service, innovation, etc.). The company also uses numeric codes to identify each department. For each employee, we need to know their name and tax file number (for payroll purposes), and also the total number of hours that they work each week. Employees may work in several departments, and the percentage of their total hours spent in each department needs to be recorded; they have to work in at least one department. Each department has a manager, and they work full-time in that role.

A possible ER design for this company is as follows:



Use this design as the basis for the rest of the Lab.

```sql
create table Employees (
    tfn         char(11),
    givenName   varchar(30),
    familyName  varchar(30),
    hoursPweek  float
);

create table Departments (
    id          char(3),
    name        varchar(100),
    manager     char(11)
);

create table DeptMissions (
```

```
    department  char(3),
    keyword     varchar(20)
);

create table WorksFor (
    employee    char(11),
    department  char(3),
    percentage  float
);
```

Questions

> Which employee works the longest hours each week?

```
SELECT *
FROM   employees
WHERE  hourspweek = (SELECT Max(hourspweek)
                     FROM   employees);
```

> What is the family name of the manager of the Sales department?

```
SELECT familyname
FROM   departments d,
       employees e
WHERE  d.manager = e.tfn;
```

> How many hours per week does each employee spend in each department?

```
SELECT w.employee,
       w.department,
       e.hourspweek * w.percentage/100
FROM   employees e,
       worksfor w
WHERE  e.tfn = w.e mployee;
```

Constraints

1. All TFN's are of the form '*ddd-ddd-ddd* ', where each *d* represents a single digit (Take a look at the PostgreSQL Manual for details on Pattern Matching and Regular Expression)
    - Using `check` keyword to add constraint.

```
create table Employees (
    tfn         char(11)  check (tfn ~ '[0-9]{3}-[0-9]{3}-[0-9]{3}'),
    givenName   varchar(30),
    familyName  varchar(30),
    hoursPweek  float
);
```

Note that, you can change the constraint name by adding `constraint NAME` before `check`

2. Every person has a given name, but may not have a family name (e.g. Prince)
    - Using `NOT NULL` to check that

```
create table Employees (
    tfn         char(11)  check (tfn ~ '[0-9]{3}-[0-9]{3}-[0-9]{3}'),
    givenName   varchar(30) NOT NULL,
    familyName  varchar(30),
    hoursPweek  float
);
```

3. Nobody can work more hours per week than there are hours in a week (each week has 7*24 = 168 hours), it is meaningless to work negative hours per week
    - Using `check` keyword to add constraint.

```
create table Employees (
    tfn         char(11)  check (tfn ~ '[0-9]{3}-[0-9]{3}-[0-9]{3}'),
    givenName   varchar(30) NOT NULL,
    familyName  varchar(30),
    hoursPweek  float check (hoursPweek ⩾ 0 and hoursPweek ⩽ 168)
);
```

4. All Departments codes consist of exactly three digits
    - Using `check` keyword to add constraint.

```
create table Departments (
    id          char(3) check (id ~ '[0-9]{3}'),
    name        varchar(100),
    manager     char(11)
);
```

5. Two Departments cannot have the same name or the same manager

- Using `unique` keyword to add constraint

```
create table Departments (
    id          char(3) check (id ~ '[0-9]{3}'),
    name        varchar(100) unique,
    manager     char(11) not null unique
);
```

6. The percentage of time that an employee works in a department must be greater than zero. An employee may spend up to and including 100% of their time in a given department
   - Using `check` keyword to add constraint.

```
create table WorksFor (
    employee    char(11),
    department  char(3),
    percentage  float check (percentage > 0.0 and percentage ⩽ 100.0)
);
```

Add the `primary key` and `foreign key` for tables.

1. Add primary key for `Employees`

```
create table Employees (
    tfn         char(11) check (tfn ~ '[0-9]{3}-[0-9]{3}-[0-9]{3}'),
    givenName   varchar(30) not null,
    familyName  varchar(30),
    hoursPweek  float
            check (hoursPweek ⩾ 0 and hoursPweek ⩽ 168),
    primary key (tfn)
);
```

2. Add primary key for `Departments`

```
create table Departments (
    id          char(3) check (id ~ '[0-9]{3}'),
    name        varchar(100) unique,
    manager     char(11) not null unique,
    primary key (id)
);
```

3. Add foreign key for `DeptMissions`

`DeptMissions` is composite attributes, therefore the `department` and `keyword` composite as primary key.

```
create table DeptMissions (
    department  char(3)
                constraint ValidDepartment references
Departments(id),
    keyword     varchar(20),
    primary key (department,keyword)
);
```

4. Add foreign key for `WorksFor`

```
create table WorksFor (
    employee    char(11)
            constraint ValidEmployee references Employees(tfn),
    department  char(3)
            constraint ValidDepartment references
Departments(id),
    percentage  float
                constraint ValidPercentage
            check (percentage > 0.0 and percentage ⩽ 100.0),
    primary key (employee,department)
);
```

Test your constraint

```
dropdb company
createdb company
psql company -f schema.sql
psql company -f data.sql
```

This worked ok before, when there was no constraint checking, but you may be distressed to find that it now generates errors. Think about the dependencies between tables and work out how to **rearrange the statements** in the `data.sql` so that the data can load ok.

Challenge 1

Here's something to think about if you found the above exercise too easy.

Exercise: Consider how you might implement the following constraints:

- no worker can have more than 100% of their time allocated

To test these out you'll need to try to insert additional tuples that violate these constraints. For the first case, you could use the following insertion:

```
insert into WorksFor values ('747-400-123','003',10);
```

```
CREATE FUNCTION check_worksfor_insert () returns TRIGGER
AS $$
  DECLARE
    percentage1 FLOAT;
    percentage2 FLOAT;
  BEGIN
    SELECT SUM(percentage)
    INTO   percentage1
    FROM   worksfor
    WHERE  employee = NEW.employee;

    percentage2 = percentage1 + NEW.percentage;
    IF percentage2 > 100 THEN
      RAISE
    EXCEPTION
      'work percentage cannot exceed 100 percent';
    END IF;
    RETURN NEW;
  END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER sum_of_percentage BEFORE INSERT OR UPDATE
ON worksfor FOR EACH ROW EXECUTE PROCEDURE
check_worksfor_insert();
```

Challenge 2

DEFERRABLE The DEFERRABLE parameter controls whether the constraint can be delayed to take effect. And `INITIALLY DEFERRED` will only be checked at the end of the transaction.

If there are two tables, a and b, a has foreign key from b, b has foreign key from a. No matter which table is imported first, an error will be reported. Because of the existence of the foreign key constraint, these insert operations are violated. Unless we do not check this first, wait until both tables are entered into the database and then check the constraints.

```
create table Employees (
    tfn         char(11)
                constraint ValidTFN
                check (tfn ~ '[0-9]{3}-[0-9]{3}-[0-9]{3}'),
    givenName   varchar(30) not null,  -- must have a given
name
```

```
    familyName  varchar(30),           -- some people have only one name
    hoursPweek  float
            check (hoursPweek >= 0 and hoursPweek <= 168), --7*24
    primary key (tfn)
);

create table Departments (
    id          char(3)                         -- [[:digit:]] == [0-9]
                constraint ValidDeptId check (id ~ '[[:digit:]]{3}'),
    name        varchar(100) unique,
    manager     char(11) not null unique
                constraint ValidEmployee references Employees(tfn) DEFERRABLE INITIALLY DEFERRED,
    primary key (id)
);

alter table Employees
    add column worksIn char(3) not null
        constraint ValidDepartment references Departments(id) DEFERRABLE INITIALLY DEFERRED;


create table DeptMissions (
    department  char(3)
                constraint ValidDepartment references Departments(id),
    keyword     varchar(20),
    primary key (department,keyword)
);


begin;
insert into employees values ('111-111-111','YANG','YANG',40.0,'100');
insert into departments values ('100','Administration','111-111-111');
commit;
```