# COMP 353 Databases
# Assignment no.4

Duc Nguyen

*Gina Cody School of Computer Science and Software Engineering*
*Concordia University, Montreal, QC, Canada*

Summer 2020

# Contents

## 1 Problem description:

Consider a DB schema consisting of the following relation schemes:

- **Regions** (**Region_ID**, Region_Name)
- **Countries** (**Country_id**, Country_Name, Region_Id)
- **Locations** (**Location_Id**, Street_address, Postal_code, City, State_Province, Country_Id)
- **Jobs** (**Job_Id**, Job_title, Min_Salary, Max_salary)
- **Departments** (**Dep_Id**, Dep_Name, Manager_Id, Location_Id)
- **Employees** (**Emp_ID**, FirstName, Last_Name, E-mail, Phone_number, Hire_date, Job_Id, Salary, Comsn_pct, Manager_Id, Dep_Id)
- **Employee_History** (**Emp_ID**, **Joining_date**, last_date, Job_ID, Dep_ID)

**Keys are underlined**

Now, express the following queries in SQL:

## 1.1 Query 1.

### 1.1.1 Description

Find the count of departments, region name(s) and cities for the department(s) that have more than 500 employees.

### 1.1.2 SQL code

```sql
select count(Dep_Id), Region_Name, City
from Regions R
join Countries C on R.Region_ID = C.Region_ID
join Locations L on C.Country_Id = L.Country_Id
join Departments D on D.Location_Id = L.Location_Id
where Dep_Id in (
    select Dep_Id
    from Employees
    group by Dep_Id
    having count(Dep_Id) > 500
)
group by Region_Name, City ;
```

## 1.2 Query 2.

### 1.2.1 Description

For a department in which the max salary is greater than 100000 for employees who worked in the past, set the manager name as Picard

```sql
update Employees e
set FirstName = 'Picard'
where Dep_Id in
    (select  Dep_Id
    from Employees
    where Emp_ID in
        (select Emp_ID from Employee_History )
    group by Dep_Id
    having max(Salary) > 10000);
```

## 1.3 Query 3.

### 1.3.1 Description

Find month and year which witnessed lowest count of employees joining a department located in Vancouver.

### 1.3.2 SQL Query

```
select month(Joining_date), year(Joining_date)
from Employee_History E
join Departments D on E.Dep_Id = D.Dep_Id
join Locations L on D.Location_Id = L.Location_Id
where L.City = 'Vancouver'
having count(E.Emp_ID) = (
    select min(count(E.Emp_ID))
    from Employee_History H
    join Departments on H.Dep_Id = D.Location_Id
    join Locations on D.Location_Id = L.Location_Id
    where L.City = 'Vancouver'
);
```

## 1.4 Query 4.

### 1.4.1 Description

With the help of schema find the year which witnessed maximum number of employee intake.

### 1.4.2 SQL Query

```
select year(A.Joining_date)
from Employee_History A
join Employees B
on A.Emp_ID=B.Emp_ID
group by year(A.Joining_date)
group by count(*) desc
limit 1;
```

## 1.5 Query 5.

### 1.5.1 Description

For the year in query-4, find how many joined in each month in that specific year

### 1.5.2 SQL Query

```
select count(Emp_ID), month(Joining_date)
from Employee_History h
where year(Joining_date) = (
        select year(A.Joining_date)
        from Employee_History A
        join Employees B
        on A.Emp_ID=B.Emp_ID
        group by year(A.Joining_date)
        group by count(*) desc
        limit 1
)
group by month(Joining_date)
order by count(Emp_ID);
```

**PART 2: Use triggers for the queries below:**

## 1.6 Query 6.

### 1.6.1 Description

Create a trigger to ensure that a salary of an employee cannot exceed the salary of his/her manager. If the employee does not have a manager, then his/her salary cannot be more than 10% of the highest salary in the database.

### 1.6.2 SQL Query

```
CREATE TRIGGER A
BEFORE INSERT OR UPDATE OF Salary , Manager_Id ON Employees
NEW ROW AS new
FOR EACH ROW
WHEN (new.Salary > (SELECT Salary
    FROM Employees
    WHERE Emp_ID = Manager_Id))
    OR
    ( new.Emp_ID IN (SELECT Emp_ID FROM Employees WHERE Manager_Id = NULL)
    AND new.Salary > 1.1 * ( SELECT MAX(Salary) FROM Employees)
Begin
    ROLLBACK;
End;
```

## 1.7 Query 7.

### 1.7.1 Description

For changes in the job of an employee, updated details provided below must be written to Employee History: hire date of the employee for start date, old job ID, old department ID, Employee ID, todays' system date for end date. In case a row is already present for employee job history then the start date must be the end date of that (row +1).

### 1.7.2 SQL Query

```
CREATE TRIGGER B
AFTER INSERT OR UPDATE OF Job_Id ON Employees
FOR EACH ROW
referencing
    old row as Old
    new row as New
declare
    enddate date;
    startdate date;
begin
    select max(last_date) into enddate
    from Employee_History
    where Employee_History.Emp_ID = old.Emp_ID;

    if enddate is null:
        insert into
        Employee_History (Joining_date, Job_ID, Dep_ID, Emp_ID, last_date)
        values (Old.Hire_date, Old.Job_Id, Old.Dep_Id, Old.Emp_ID, sysdate);
    else:
        startdate = enddate + 1;
        update Employee_History
        set Joining_date = startdate, last_date = sysdate
        where Old.Emp_ID = Employee_History.Emp_ID;
    end if;
```

## 1.8 Query 8.

### 1.8.1 Description

Make a Trigger to ensure that the salary of the employee is never decreased while working in an organization.

### 1.8.2 SQL Query

```
create trigger C
after update of Salary on Employees
referencing old row as OldT, new row as NewT
for each row
when (OldT.Salary > NewT.Salary)
begin
    update Employees
    set Salary = OldT.Salary
end;
```

## 1.9 Query 9.

### 1.9.1 Description

Create a trigger to ensure that an increase of salary for an employee is conform with the following rules: If experience is more than 8 years, increase salary by max 20%; If experience is greater than 3 years, increase salary by max of 10%; Otherwise a max increase of 5%.

### 1.9.2 SQL Query

```
create trigger D
before update of Salary on Employees
referencing
    old row as Old
    new row as New
for each row
when (
    ((getdate() - new.Hire_date) < 8 and (new.Salary > 1.20 * old.Salary) )
    or ((getDate() - new.Hire_date) < 3 and (new.Salary > 1.10 * old.Salary))
    or (new.Salary > 1.05 * old.Salary)
    )
begin
    rollback;
end;
```

### 1.10 Query 10.

#### 1.10.1 Description

Create a trigger to ensure that Min_salary cannot exceed Max_salary for any job

#### 1.10.2 SQL Query

```
create trigger E
before update on Jobs
referencing
    new row as newTuple
for each row
when (newTuple.Min_Salary > newTuple.Max_Salary)
begin
    rollback;
end
```