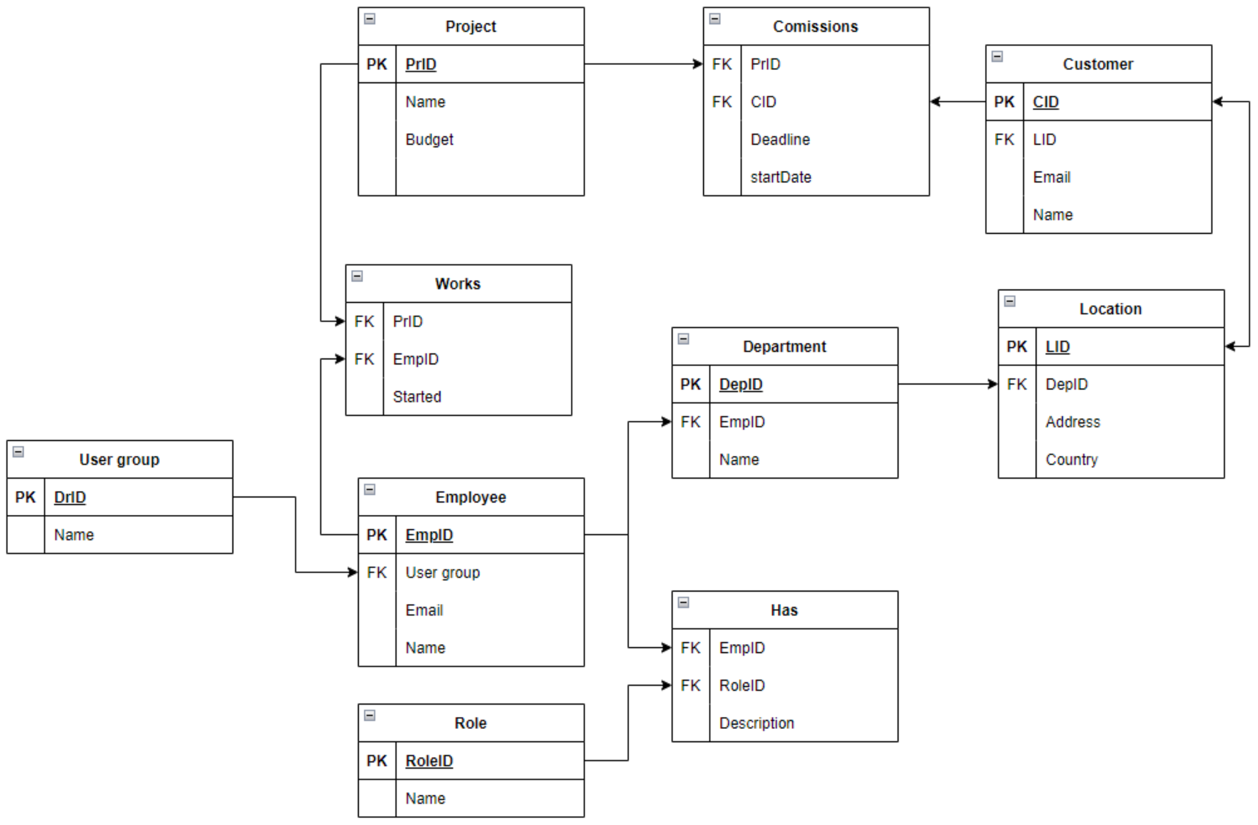
# Database systems management

# Final project

Made by: Otto Åhlfors

## 1. Relational model or database schema based on the ER model



## 2. Integrity rules

## Task 2.1.

If a primary key is deleted or updated, then all of its foreign keys will not be applicable anymore so here I have defined some rules defining what should be done to them.

ON UPDATE no primary key should be needed to update since it is a static ID but if for some reason you should want to do so a CASCADE to update all of its foreign keys should be used. This applies to all ON UPDATE commands.

ON DELETE is more complex because depending on what we want to delete it may have different consequences, so I have listed all ON DELETE scenarios for Project, Customer, User group and Employee tables. If a project is to be deleted its primary key should be set to NULL, this means that employees who worked on that project and customer that commissioned the project are not deleted. If a customer is to be deleted for example if a contract has been ended with them then all projects done to that customer could reasonably be deleted also. Unless there is a specific reason to have them saved up. If a user group is deleted, we do not want to delete all of the employees that are a part of said group so just set the GrID on the employees to NULL. If an employee was to leave the company and would have to be deleted from the database that employee should just be set to NULL on all of the projects and departments that they were a part of.

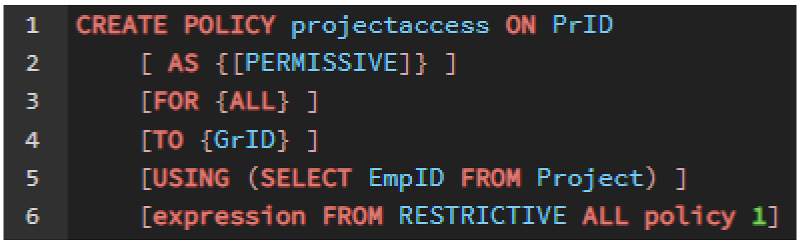
## 3. Partitioning

### Task 3.1.

Data could be partitioned by location to see what departments are int said location, what projects are being worked on in this location and which customers and employees are in each location. Another good partition would be by CID/ customer that way you can easily se all the projects that are being done to a single customer, where they are located and when these projects should be completed.

## 4. Access rights for users

### Task 4.1.



```
CREATE POLICY projectaccess ON PrID
    [ AS {[PERMISSIVE]} ]
    [FOR {ALL} ]
    [TO {GrID} ]
    [USING (SELECT EmpID FROM Project) ]
    [expression FROM RESTRICTIVE ALL policy 1]
```

## 5. Management of values

### Task 5.1.

Default values are, Budget = 0,  StartDate = current date, Country (assuming that the company is entirely within one county) = Finland

### Task 5.2.

The only CHECK constraint that I find to be useful for this database is NOT NULL. Others could be used if the database had more information or was otherwise more complex. For example, if the company's total budget was in the database a CHECK could be made so that the projects budget cannot exceed total remaining budget.

CHECK constraint NOT NULL:

CommisionsDeadline, EmployeeName, LocationAddress, CustomerName, CustomerEmail, ProjectName, UserGroupName, RoleName, EmployeeEmail

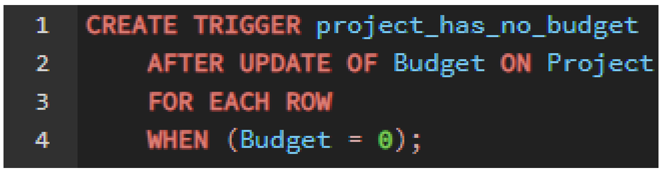
### Task 5.3.

For this database there is no reason to accept NULL values so it is easiest to simply not allow them at all.

## 6. Triggers and trigger graph

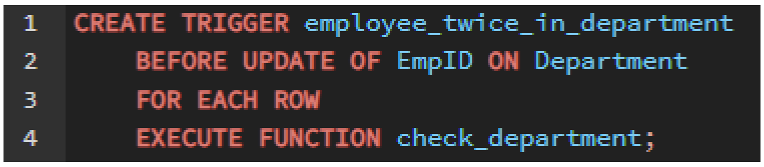
### Task 6.1.

This trigger checks weather or not the project has any budget assigned to it.



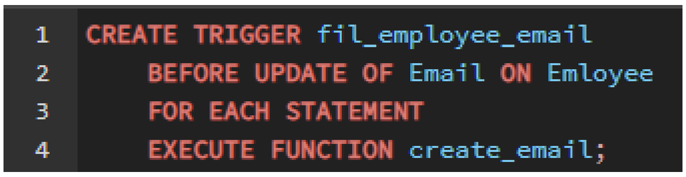
```
CREATE TRIGGER project_has_no_budget
    AFTER UPDATE OF Budget ON Project
    FOR EACH ROW
    WHEN (Budget = 0);
```

This trigger checks if the same employee is logged twice in the same department.



```
CREATE TRIGGER employee_twice_in_department
    BEFORE UPDATE OF EmpID ON Department
    FOR EACH ROW
    EXECUTE FUNCTION check_department;
```

This trigger automatically fills the employees work email to the email column.



```
CREATE TRIGGER fil_employee_email
    BEFORE UPDATE OF Email ON Emloyee
    FOR EACH STATEMENT
    EXECUTE FUNCTION create_email;
```

### Task 6.2.

## 7. Security issues and measures

### Task 7.1.

Security issues and their countermeasures:

Stolen computer: It is always possible that an employee either is robbed or accidentally forgets or otherwise loses their computer and someone who is not supposed to gets accesses to the database. This can be prevented with the use of passwords. These passwords should be long, complex and should not use any personal information. A password given to each employee that fulfils all the requirements for a strong password should make this kind of security issues a lot less likely to happen.

Hacking: To prevent hacking an employees account and getting into the database that way an encryption for the database or the data transit should be used. This way if the database is hacked and someone gets to take a peek at it they still need an encryption key to access the data.

Malware: Malware is aways a serious issue when it comes to computers. To prevent malware the basic rules of using computers safely are usually enough and when that is accompanied by an employee having a separate work computer that is used only for work the risk of malware getting in by accident is small. These basic practises include not plugging in unknown devices or memory sticks, having some sort of malware protection on the computer preferably installed there by the company and avoiding dubious websites.

In addition to all of these a label based access control is good to have so that even if compromised the unauthorized personnel can only access some of the data.

# 8. Backup and recovery, database disaster plan

## Task 8.1.

A full backup could be used to back up the database every time a project is completed and once a month. The project completion is a good time for a backup since that is some of the most important information to keep. It is good to have a timed backup also so that any changes done during the project are also kept if there is a long timeframe where no projects are completed. These changes are for example the addition or deletion of new and old employees or changes in department compositions.

## Task 8.2.

Realistic disasters:

A fire that burns the servers is a very likely occurrence since a server room contains a large amount of electric devices any of which can light up accidentally.

Corruption of data can be caused by a number of things for example a blackout may do some damage or just a human error.

DDoS attack, for any reason or for no reason someone might do a DDoS attack against the company's servers would lead to the employees not being able to connect to the database and using the backup would be the only option.

Extremely unlikely disaster:

An earthquake could destroy a server facility completely and thus make it necessary to use backups.

## Task 8.3.

All of the disasters I have identified except for the corruption of data would be best if they were recovered to current after the disaster in question is over. In the case of a DDoS attack it might not be necessary to recover the data at all if the situation can be solved and no data has been corrupted once it is over. In the case of a corruption of data a point-in-time recovery to the most recent not corrupted backup would be the best.