Conceptual Design for Project Management Database

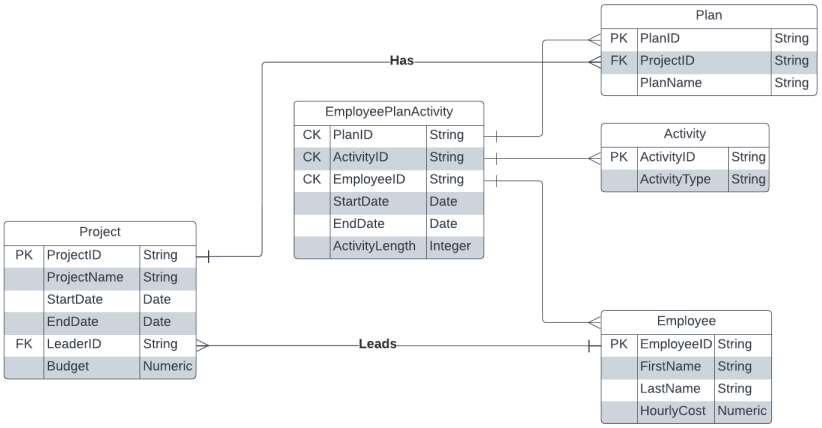
Group 8

Christian Nilsen, Ina Folland Hegg, Ine Sofie Løseth, Siri Sandnes

Problem analysis

It says the project needs a name, start- and end-date, leader and budget. It should also have an ID, so the name doesn't have to be unique. Employees have names and hourly cost. We split the name into first- and last names because we want to minimize independent parts. Activity can have different types. We also add an ID, because if we want to add other attributes later it would be useful to not use type as key. Plan contains the project, employees and activities with start- and end-dates. Here we assume the way that it's written that the start and end dates pertain to the activities, and not the plan. We also decided to give the plans names, so they're more easily identifiable. Since the queries wanted to know about employees' working hours, we also added activity length (hours) to see how long employees work. Since the task wants to use security features, we decided to use PostgreSQL to make constraints.

Conceptual design



A project can have multiple plans, but a plan can only be part of one project.
A plan can consist of multiple activities, and an activity can be part of multiple plans.
An employee can lead multiple projects, but a project can only be led by one employee.
An employee can participate in multiple plans, and a plan can employ multiple employees.
An employee can do multiple activities, and an activity can be done by multiple employees.

We chose to solve the many-to-many relationships between plan and activity, plan and employee, and employee and activity by creating a table with composite primary keys from plan, activity and employee. We thought anything else would be redundant and unnecessarily complex.

We added activity length in employee-plan-activity, because we needed a way to keep track of working time. We put it there, and not in activity, because it made more sense to keep it where the

start date and end date attributes were located. We put start- and end-date in employee-plan-activity, because it felt unique to every unique scenario where an employee is part of a plan and does an activity for that plan. If we put it in plan, then every activity would “inherit” it from their plan. If we put it in activity, then we would make a new activityID every time a plan wants to do an activity.

Query question solutions

How many employees of a project titled "A" are involved in its plan “B”?

```
SELECT COUNT(*) AS "Project_A_Employees_In_Plan_B"
FROM "Employee" LEFT JOIN "EmployeePlan" "ep"
USING("EmployeeID")
WHERE "ep"."PlanID" IN (
    SELECT "PlanID"
    FROM "Plan"
    WHERE "Name" = 'B'
        AND "ProjectID" IN (
        SELECT "ProjectID"
        FROM "Project"
        WHERE "ProjectName" = 'A'
    )
);
```

Retrieve the names of plans made for project “A” with least cost.

```
SELECT DISTINCT "Name" AS "PlanName"
FROM "Plan"
LEFT JOIN "Project" USING ("ProjectID")
LEFT JOIN (
        SELECT "ActivityLength", "HourlyCost", "PlanID"
        FROM "EmployeePlanActivity"
        LEFT JOIN "Employee" USING ("EmployeeID")
        GROUP BY "PlanID", "ActivityLength", "HourlyCost"
) AS "TableA" USING ("PlanID")
WHERE ("ActivityLength" * "HourlyCost") IN (
    SELECT "ActivityLength"*"HourlyCost"
    FROM "EmployeePlanActivity"
    JOIN "Employee" USING ("EmployeeID")
    JOIN "Plan" USING ("PlanID")
    JOIN "Project" ON "Project"."ProjectID" = "Plan"."ProjectID"
    WHERE "ProjectName" = 'A'
);
```

For each employee retrieve the name, project name and plan name with the most

working time.

```
SELECT "FirstName", "LastName", "EmployeeID", "Name" AS "PlanName",
"ProjectName", "ActivityLength"
FROM "EmployeePlanActivity"
JOIN "Employee" USING ("EmployeeID")
JOIN "Plan" USING ("PlanID")
JOIN "Project" ON "Project"."ProjectID" = "Plan"."ProjectID"
WHERE ("EmployeeID", "ActivityLength") IN (
    SELECT "EmployeeID", MAX("ActivityLength")
    FROM "EmployeePlanActivity"
    GROUP BY "EmployeeID"
);
```

Retrieve all the employee's name and their least working time with respect to different project.

Here we assume this means get the length of an employee's activity in plans per project.

```
SELECT "FirstName", "LastName", MIN(Total_Time)
FROM (
    SELECT "EmployeeID", "ProjectName", SUM("ActivityLength") AS Total_Time
    FROM "EmployeePlanActivity"
    JOIN "Plan" USING ("PlanID")
    JOIN "Project" ON "Project"."ProjectID" = "Plan"."ProjectID"
    GROUP BY "EmployeeID", "Project"."ProjectID"
) AS "TableA"
JOIN "Employee" USING ("EmployeeID")
GROUP BY "EmployeeID", "FirstName", "LastName";
```

Retrieve all the plans for project with order of their working period.

We assume here that "working period" means the project length in days, in ascending order from lowest to highest.

```
SELECT "Name" AS "PlanName"
FROM "Plan"
WHERE "ProjectID" IN (
   SELECT "ProjectID"
   FROM "Project"
   ORDER BY "EndDate" - "StartDate"
);
```