SQL Case Study: Human Resources

MEBU Manuella Kevine

Case Study: DATA IN MOTION (Kedeisha Bryan)

Friday 2023/06/02

SQL Questions

1. Find the longest ongoing project for each department.
2. Find all employees who are not managers.
3. Find all employees who have been hired after the start of a project in their department.
4. Rank employees within each department based on their hire date (earliest hire gets the highest rank).
5. Find the duration between the hire date of each employee and the hire date of the next employee hired in the same department.

Data modeling

We have three tables:

- Departments (id, name, manager_id);
- Employees (id, name, hire_date, job_title, department_id);
- Projects (id, name, start_date, end_date, department_id).

The department_id is the common field between the three tables. It is a primary key in the Department table, and a foreing key in the tables Employees and Project.

Employee_id of Employees table is foreign in the Departments table, it bares the name manager_id

Data modeling

According to our field of study, let us bring out the relationships between the different tables « *classe d'entités du modèle conceptuel des données*».

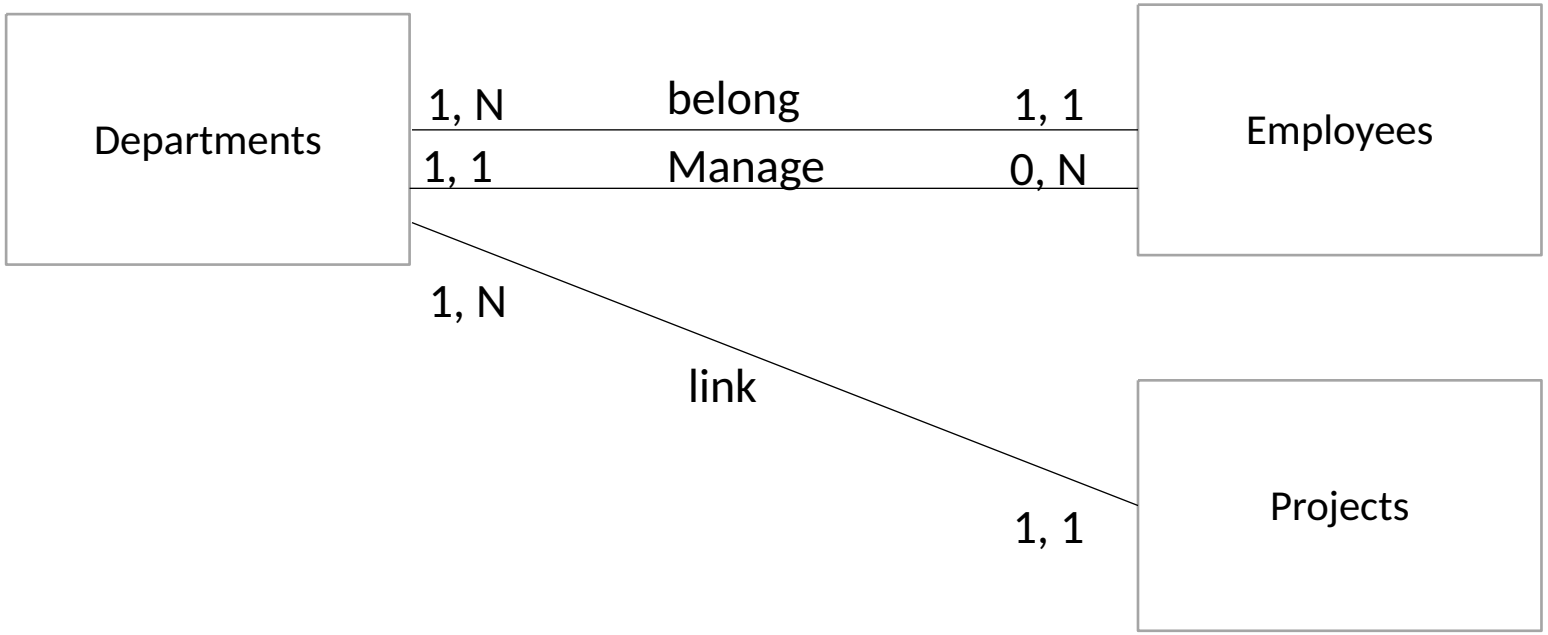

Figure 1 : Human Resources Entity-Relationship

CREATE THE DATABASE AND TABLES with MS SQL Server

```sql
USE datainmotion

GO
DROP TABLE IF EXISTS departments
CREATE TABLE departments (
   id int PRIMARY KEY,
   name VARCHAR(50),
   manager_id INT
);

GO
DROP TABLE IF EXISTS employees
CREATE TABLE employees (
   id int PRIMARY KEY,
   name VARCHAR(50),
   hire_date DATE,
   job_title VARCHAR(50),
   department_id INT
);
```

```sql
DROP TABLE IF EXISTS projects
CREATE TABLE projects (
   id int PRIMARY KEY,
   name VARCHAR(50),
   start_date DATE,
   end_date DATE,
   department_id INT
);

GO
ALTER TABLE employees
ADD FOREIGN KEY(department_id) REFERENCES departments(id)

GO
ALTER TABLE projects
ADD FOREIGN KEY(department_id) REFERENCES departments(id)

GO
ALTER TABLE departments
ADD FOREIGN KEY(manager_id) REFERENCES employees(id)
```

INSERT & UPDATE VALUES

```sql
INSERT INTO departments (id,name, manager_id)
VALUES (1,'HR', 1), (2,'IT', 2), (3,'Sales', 3)

INSERT INTO employees (id,name, hire_date, job_title,
department_id)
VALUES
(1,'John Doe', '2018-06-20', 'HR Manager', 1),
(2,'Jane Smith', '2019-07-15', 'IT Manager', 2),
(3,'Alice Johnson', '2020-01-10', 'Sales Manager', 3),
(4,'Bob Miller', '2021-04-30', 'HR Associate', 1),
(5,'Charlie Brown', '2022-10-01', 'IT Associate', 2),
(6,'Dave Davis', '2023-03-15', 'Sales Associate', 3)

INSERT INTO projects (id,name, start_date, end_date,
department_id)
VALUES
 (1,'HR Project 1', '2023-01-01', '2023-06-30', 1),
 (2,'IT Project 1', '2023-02-01', '2023-07-31', 2),
 (3,'Sales Project 1', '2023-03-01', '2023-08-31', 3),
 (4,'HR Projects 2','2023-01-01','2023-12-20',1),
 (5,'Sales Project 2', '2023-03-01', '2023-06-30', 3)
```

```sql
UPDATE departments
SET manager_id = (SELECT id FROM employees
WHERE name = 'John Doe')
WHERE name = 'HR';

UPDATE departments
SET manager_id = (SELECT id FROM employees
WHERE name = 'Jane Smith')
WHERE name = 'IT';

UPDATE departments
SET manager_id = (SELECT id FROM employees
WHERE name = 'Alice Johnson')
WHERE name = 'Sales';
```

Data Cleaning

- On the previous slide (slide 6), we did insert data manually.
- Tables have a maximum of 6 records.
- The values inserted are consistent, we did respect data integrity (primary and foreign key).
- We will go directly to the part, which involves answering business questions.

1) Find the longest ongoing project for each department

```
WITH a AS(
SELECT id, department_id, name, DATEDIFF(day,start_date,end_date) duration_days,
MAX(DATEDIFF(day,start_date,end_date)) OVER (PARTITION BY department_id) max_duration_days
FROM projects
WHERE end_date >= GETDATE())

SELECT id, name, department_id FROM a
WHERE duration_days = max_duration_days
```

Résultats | Messages

	id	name	department_id
1	4	HR Projects 2	1
2	2	IT Project 1	2
3	3	Sales Project 1	3

Thought process:

- **NB**: I added two extra records in the projects table with project ID 4 and 5 (I wanted to see if my code still holds for another scenario);
- Find the total days between start date and end date for each project;
- Partition the above by each department to have the max duration;
- Add the WHERE clause to obtain only ongoing projects;
- Enclose it in a WITH clause;
- Select the project name and id where duration for each project = max duration days obtained with the partition clause.

2) Find all employees who are not managers

```
SELECT id, name, job_title FROM  employees
WHERE id not in (SELECT manager_id FROM departments)
```

Thought process:

- Query managers' ID;
- Use it as a sub-query in the main query;
- Select id, name, if you want job title to have employees who are not managers.

Results:

Résultats | Messages

	id	name	job_title
1	4	Bob Miller	HR Associate
2	5	Charlie Brown	IT Associate
3	6	Dave Davis	Sales Associate

3) Find all employees who have been hired after the start of a project in their department

```
SELECT DISTINCT a.id, a.name  FROM employees a
JOIN projects b ON a.department_id = b.department_id
WHERE a.hire_date > b.start_date
```

Thought process:

- Tables employees and projects both have department id;
- Do an inner join;
- Use a where clause to have only employees hired after the start of a project.

Results:

Résultats | Messages

	id	name
1	6	Dave Davis

4) Rank employees within each department based on their hire date (earliest hire gets the highest rank)

```
SELECT id, name, hire_date, department_id, DENSE_RANK() OVER (PARTITION BY department_id
ORDER BY hire_date ) rank_employees_hiring_date FROM employees
```

Thought process:

- Use the window function DENSE_RANK () to attribute a rank for each record in the different subgroups;
- Partition employees table by department and order each group in an ascending order by the column of interest which is “hire_date”.

Results:

Résultats | Messages

	id	name	hire_date	department_id	rank_employees_hiring_date
1	1	John Doe	2018-06-20	1	1
2	4	Bob Miller	2021-04-30	1	2
3	2	Jane Smith	2019-07-15	2	1
4	5	Charlie Brown	2022-10-01	2	2
5	3	Alice Johnson	2020-01-10	3	1
6	6	Dave Davis	2023-03-15	3	2

5) Find the duration between the hire date of each employee and the hire date of the next employee hired in the same department

```
SELECT id, name, hire_date, department_id, LEAD(hire_date,1) OVER (PARTITION BY department_id ORDER BY hire_date ) next_employee_hiring_date,

CASE

WHEN DATEDIFF(day, hire_date, LEAD(hire_date,1) OVER (PARTITION BY department_id ORDER BY hire_date )) IS NULL THEN 0

ELSE DATEDIFF(day, hire_date, LEAD(hire_date,1) OVER (PARTITION BY department_id ORDER BY hire_date )) END  duration_days

FROM employees
```

Résultats | Messages

	id	name	hire_date	department_id	next_employee_hiring_date	duration_days
1	1	John Doe	2018-06-20	1	2021-04-30	1045
2	4	Bob Miller	2021-04-30	1	NULL	0
3	2	Jane Smith	2019-07-15	2	2022-10-01	1174
4	5	Charlie Brown	2022-10-01	2	NULL	0
5	3	Alice Johnson	2020-01-10	3	2023-03-15	1160
6	6	Dave Davis	2023-03-15	3	NULL	0

Thought process:

- Since we are interested in the next and not the previous employee, I thought of using the LEAD function;
- We add a partition by, because the question says, the employees should be in thesame department.

MERCI