Relatório do Trabalho Final

Disciplina de Banco de Dados I

Arthur Scarpatto Rodrigues, Leonardo Luiz Gambalonga

Contents

1. Sistema de Gerenciamento de RH

Nosso trabalho segue o esquema de um software de gerenciamento de RH simples, que oferece recursos à empresa para considerar seus custos, tempo, e alocar recursos efetivamente.

1.1. Descrição do Objetivo Geral do Sistema

O objetivo do sistema é providenciar um sistema de gerenciamento de RH de uma empresa fictícia, onde constam dados sobre os funcionários, sobre a hierarquia de comando, compensação, calendários, projetos e alocações de fundos, horários de projetos e de funcionários, alocação de tempo e funcionários para projetos específicos, assim como algumas outras funcionalidades, seguindo o mesmo padrão.

1.2. Descrição Detalhada do Sistema

O sistema contém sete tabelas, que giram em torno da tabela principal, *employees*, responsável por armazenar dados de funcionários específicos. As outras tem como objetivo tratar da alocação de tempo e recursos para projetos, quais funções os funcionários desempenham, e seus horários individuais. Seguem abaixo as especificações técnicas de cada tabela.

1.2.1 Tabela *employees*

A tabela *employees* tem como colunas os valores mostrados na tabela abaixo.

Nome da Coluna	Tipo	Modificadores	Referencia
employee_id	serial	PRIMARY KEY	
department_id	integer	NOT NULL	departments (FK)
name	varchar	NOT NULL	
salary	integer	NOT NULL	
date_of_birth	date	NOT NULL	
gender	varchar		
contact_number	varchar		
email	varchar	NOT NULL	

Table 1: Informações técnicas sobre a tabela *employees*.

A tabela tem uma chave estrangeira para a tabela *departments*, a ser discutida a diante. Essa relação é uma ***many-to-one***, já que vários funcionários podem estar associados a um único departamento cada.

A maioria das colunas é auto-explicatória, de acordo com seus nomes, em relação à função que desempenham. Vale mencionar que *gender* e *contact-number* não são necessários admnistrativamente (no caso de *contact-number*, porque já temos o *email*), e os funcionários podem preferir não mencioná-las, então são consideradas anuláveis.

1.2.2 Tabela *departments*

Essa é uma tabela simples que contém informações básicas sobre o departamento, para fins de organização. Encontram-se na tabela abaixo as especificações.

Nome da Coluna	Tipo	Modificadores	Referencia
department_id	serial	PRIMARY KEY	
department_name	varchar	NOT NULL	

Table 2: Informações técnicas sobre a tabela *departments*.

Essa tabela não contém referências a nenhuma outra. Também não contém nenhum **CASCADE** associado, já que podemos fechar um departamento sem necessáriamente demitir os funcionários.

1.2.3 Tabela *roles*

Outra simples tabela que contém informações sobre as funções que podem estar associadas a um certo funcionário, de acordo com a especificação na tabela abaixo.

Nome da Coluna	Tipo	Modificadores	Referencia
role_id	serial	PRIMARY KEY	
role_name	varchar	NOT NULL	

Table 3: Informações técnicas sobre a tabela *roles*.

Também não referencia nenhuma outra tabela. No entanto, possui efeito de **CASCADE** ao ser deletada uma entrada, efetivamente deletando a(s) entrada(s) correspondente(s) na tabela *employee_roles*, que representa uma associação ***many-to-many*** entre *employees* e *roles*.

Nesse sentido, *employee_roles* representa uma função de ***many-to-one*** para com *roles*, já que cada entrada na primeira tabela contém uma referência para alguma entrada da segunda.

1.2.4 Tabela *employee_roles*

Como dito anteriormente, essa tabela representa uma relação ***many-to-many*** entre *employees* e *roles*. Dessa maneira, vários funcionários podem desempenhar vários papéis na empresa, e vários papéis podem ser desempenhados por vários funcionários.

Nome da Coluna	Tipo	Modificadores	Referencia
employee_role_id	serial	PRIMARY KEY	
employee_id	integer	NOT NULL	employees (FK)
role_id	integer	NOT NULL	roles (FK)
start_date	date	NOT NULL	
end_date	date	NOT NULL	

Table 4: Informações técnicas sobre a tabela *employee_roles*.

Não possui efeito de **CASCADE** próprio, mas é deletada caso uma entrada da tabela *roles* ou uma da tabela *employees* seja deletada, pois não faz sentido existir uma associação de um funcionário com uma função se algum dos dois não existe.

1.2.5 Tabela *schedules*

Representa os horários dos funcionários e os dias em que começaram a trabalhar/dias em que vão acabar, ou sair da empresa. Essa última informação, claro, pode ser anulável, pois na maioria dos casos, não se sabe ao certo previamente quanto tempo um funcionário vai ficar na empresa.

Nome da Coluna	Tipo	Modificadores	Referencia
schedule_id	serial	PRIMARY KEY	
employee_id	integer	NOT NULL	employees (FK)
start_date	date	NOT NULL	
end_date	date		
start_time	timestamp	NOT NULL	
end_time	timestamp		

Table 5: Informações técnicas sobre a tabela *schedules*.

Aqui, o *start_time* e *end_time* se referem aos horários de um funcionário, em um determinado dia.

A tabela *schedules* é deletada por efeito de **CASCADE** caso alguma entrada de *employees* seja deletada. Nessa tabela, temos uma relação de ***many-to-one*** para com *employees*.

1.2.6 Tabela *projects*

Essa tabela representa projetos desenvolvidos pela empresa, assim como suas respectivas verbas, e metadados como a data de inicio e data de fim prevista (opcional).

Nome da Coluna	Tipo	Modificadores	Referencia
project_id	serial	PRIMARY KEY	
project_name	varchar	NOT NULL	
budget	integer	NOT NULL	
start_date	date	NOT NULL	
end_date	date		

Table 6: Informações técnicas sobre a tabela *projects*.

A tabela *projects* possui efeito de **CASCADE** em relação à tabela *project_assignments*, que representa uma relação ***many-to-many*** de *employees* para *projects*. Dessa forma, *project_assignments* também relaciona um papel de ***many-to-one*** para com *projects*.

1.2.7 Tabela *project_assignments*

Como dito anteriormente, essa tabela representa um ***many-to-many*** de *employees* e *projects*, com a função lógica de que um funcionário pode estar em vários projetos, e um projeto pode ser executado por vários funcionários.

Nome da Coluna	Tipo	Modificadores	Referencia
assignment_id	serial	PRIMARY KEY	
project_id	integer	NOT NULL	projects (FK)
employee_id	integer	NOT NULL	employees (FK)
assigned_at	date	DEFAULT CURRENT_DATE	

Table 7: Informações técnicas sobre a tabela *project_assignments*.

Aqui, *assigned_at* significa a data em que um funcionário foi alocado para um dado projeto.

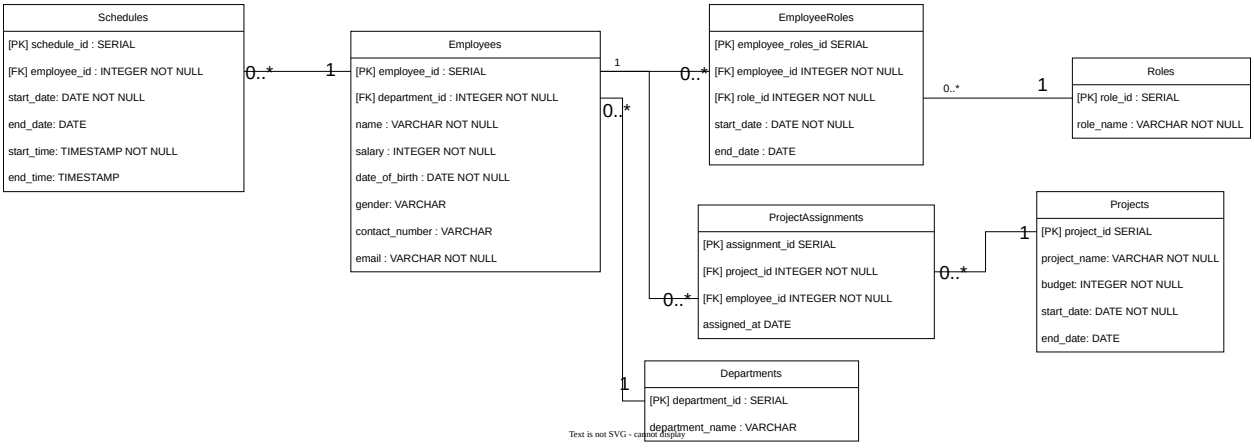
Essa tabela sofre efeitos de **CASCADE** tanto de *employees* quanto de *projects*: se alguma entrada em uma dessas tabelas for deletada, e tiver entradas relacionadas em *project_assignments*, então essas entradas também serão deletadas.

1.3. Modelagem Conceitual

Aqui vai a modelagem conceitual da paradinha mermão i love org

1.4. Modelagem Lógica

Segue abaixo uma imagem representando a modelagem lógica do nosso trabalho final. A mesma pode ser encontrada no diretório base deste trabalho.



1.5. *Script - Data Definition Language* (DDL)

Segue abaixo o *script* DDL de criação das tabelas e relacionamentos do nosso sistema de banco de dados.

```
CREATE TABLE "employees" (
    "employee_id" serial PRIMARY KEY,
    "department_id" integer NOT NULL,
    "name" varchar NOT NULL,
    "salary" integer NOT NULL,
    "date_of_birth" date NOT NULL,
    "gender" varchar,
    "contact_number" varchar,
    "email" varchar NOT NULL
```

```
);

CREATE TABLE "departments" (
    "department_id" serial PRIMARY KEY,
    "department_name" varchar NOT NULL
);

CREATE TABLE "roles" (
    "role_id" serial PRIMARY KEY,
    "role_name" varchar NOT NULL
);

CREATE TABLE "employee_roles" (
    "employee_role_id" serial PRIMARY KEY,
    "employee_id" integer NOT NULL,
    "role_id" integer NOT NULL,
    "start_date" date NOT NULL,
    "end_date" date
);

CREATE TABLE "schedules" (
    "schedule_id" serial PRIMARY KEY,
    "employee_id" integer NOT NULL,
    "start_date" date NOT NULL,
    "end_date" date,
    "start_time" timestamp NOT NULL,
    "end_time" timestamp
);

CREATE TABLE "projects" (
    "project_id" serial PRIMARY KEY,
    "project_name" varchar NOT NULL,
    "budget" integer NOT NULL,
    "start_date" date NOT NULL,
    "end_date" date
);

CREATE TABLE "project_assignments" (
    "assignment_id" serial PRIMARY KEY,
    "project_id" integer NOT NULL,
    "employee_id" integer NOT NULL,
    "assigned_at" date DEFAULT CURRENT_DATE,
);

ALTER TABLE "employees"
    ADD FOREIGN KEY ("department_id") REFERENCES "departments" ("department_id");

ALTER TABLE "employee_roles"
    ADD FOREIGN KEY ("employee_id") REFERENCES "employees" ("employee_id") ON DELETE CASCADE,
    ADD FOREIGN KEY ("role_id") REFERENCES "roles" ("role_id") ON DELETE CASCADE;

ALTER TABLE "schedules"
    ADD FOREIGN KEY ("employee_id") REFERENCES "employees" ("employee_id") ON DELETE CASCADE;
```

```
ALTER TABLE "project_assignments"
    ADD FOREIGN KEY ("project_id") REFERENCES "projects" ("project_id") ON DELETE CASCADE,
    ADD FOREIGN KEY ("employee_id") REFERENCES "employees" ("employee_id") ON DELETE CASCADE;
```

1.6. *Script - Data Manipulation Language* (DML)

Segue abaixo o *script* DML de população das tabelas e campos do nosso banco de dados.

```
-- Inserting data into 'departments'
INSERT INTO "departments" ("department_name") VALUES
                                                ('Human Resources'),
                                                ('IT'),
                                                ('Marketing'),
                                                ('Finance');

-- Inserting data into 'roles'
INSERT INTO "roles" ("role_name") VALUES
                                    ('Manager'),
                                    ('Developer'),
                                    ('Analyst'),
                                    ('Administrator');

-- Inserting data into 'employees'
INSERT INTO "employees" ("department_id", "name", "salary", "date_of_birth", "gender",
                                                            "contact_number", "email") VALUES
    (1, 'John Doe', 50000, '1980-01-15', 'Male', '555-0101', 'john.doe@example.com'),
    (2, 'Jane Smith', 15000, '1985-05-20', 'Female', '555-0102', 'jane.smith@example.com'),
    (3, 'Alice Johnson', 10000, '1990-07-25', 'Female', '555-0103', 'alice.johnson@example.com'),
    (4, 'Marie Johnson', 22000, '1990-07-25', 'Female', '555-0104', 'marie.jhn@example.com'),
    (1, 'Julie Smith', 34500, '1999-01-20', 'Female', '555-0105', 'julsmith@example.com'),
    (2, 'Johnathan Lima', 17500, '2001-08-01', 'Male', '555-0106', 'jlima123@example.com'),
    (4, 'Lian Bark', 3500, '2002-03-27', 'Male', '555-0107', 'barklian@example.com'),
    (3, 'Oswald Guy', 3000, '2000-07-07', 'Male', '555-0108', 'oswaldguy@example.com'),
    (1, 'Evan Peters', 45000, '1988-04-15', 'Male', '555-0110', 'evan.peters@example.com'),
    (2, 'Sara Connor', 20000, '1992-12-20', 'Female', '555-0111', 's.connor@example.com'),
    (3, 'Raj Patel', 5500, '1995-09-30', 'Male', '555-0112', 'raj.patel@example.com'),
    (4, 'Luna Star', 7000, '1986-11-05', 'Female', '555-0113', 'luna.star@example.com');


-- Inserting data into 'employee_roles'
INSERT INTO "employee_roles" ("employee_id", "role_id", "start_date", "end_date") VALUES
    (1, 1, '2020-01-01', NULL),
    (2, 1, '2020-01-01', NULL),
    (3, 1, '2020-01-01', NULL),
    (4, 1, '2020-01-01', NULL),
    (5, 2, '2021-01-01', NULL),
    (6, 3, '2021-02-01', NULL),
    (7, 3, '2021-03-01', NULL),
    (8, 2, '2021-04-01', NULL),
    (9, 2, '2021-02-01', NULL),
    (10, 4, '2021-03-01', NULL),
    (11, 2, '2021-04-01', NULL),
    (12, 2, '2021-04-01', NULL);

    -- Inserting data into 'schedules'
```

```
INSERT INTO "schedules" ("employee_id", "start_date", "end_date", "start_time", "end_time") VALUES
    (1, '2023-01-01', '2023-12-31', '2023-01-01 09:00:00', '2023-01-01 17:00:00'),
    (2, '2023-01-01', '2023-12-31', '2023-01-01 09:00:00', '2023-01-01 17:00:00'),
    (3, '2023-01-02', '2023-12-31', '2023-01-02 09:00:00', '2023-01-02 17:00:00'),
    (4, '2023-01-02', '2023-12-31', '2023-01-02 09:00:00', '2023-01-02 17:00:00'),
    (5, '2023-01-03', '2023-12-31', '2023-01-03 08:45:00', '2023-01-03 17:15:00'),
    (6, '2023-01-04', '2023-12-31', '2023-01-04 09:15:00', '2023-01-04 17:30:00'),
    (7, '2023-01-05', '2023-12-31', '2023-01-05 09:00:00', '2023-01-05 17:00:00'),
    (8, '2023-01-06', '2023-12-31', '2023-01-06 08:50:00', '2023-01-06 17:05:00'),
    (9, '2023-01-07', '2023-12-31', '2023-01-07 09:10:00', '2023-01-07 17:10:00'),
    (10, '2023-01-02', '2023-12-31', '2023-01-02 09:00:00', '2023-01-02 17:00:00'),
    (11, '2023-01-03', '2023-12-31', '2023-01-03 08:45:00', '2023-01-03 17:15:00'),
    (12, '2023-01-04', '2023-12-31', '2023-01-04 09:15:00', '2023-01-04 17:30:00');

-- Inserting data into 'projects'
INSERT INTO "projects" ("project_name", "budget", "start_date", "end_date") VALUES
    ('Website Redesign', 50000, '2023-01-01', '2023-06-30'),
    ('Market Research', 150000, '2023-02-01', '2023-08-31'),
    ('Internal Software Development', 300000, '2023-02-01', '2023-12-31'),
    ('Employee Engagement Survey', 5000, '2023-03-01', '2023-05-31'),
    ('Attracting New Investors', 150000, '2023-01-01', '2025-12-31');

-- Inserting data into 'project_assignments'
INSERT INTO "project_assignments" ("project_id", "employee_id") VALUES
    (5, 1),
    (5, 2),
    (5, 4),
    (5, 7),
    (5, 11),
    (4, 2),
    (4, 5),
    (4, 9),
    (4, 12),
    (3, 12),
    (1, 2),
    (1, 6),
    (2, 3),
    (2, 8),
    (2, 10),
    (2, 6),
    (3, 2),
    (3, 6),
    (3, 10);
```

1.7. Um Sumário do Sistema em Três Consultas

Nessa seção, mostramos três consultas no banco de dados, seguindo os seguintes objetivos:

1. Cada consulta deve projetar pelo menos duas colunas.
2. Duas das consultas devem ter uma das colunas com uma função de agregação.
3. A última consulta deve utilizar *left* ou *right join*.
4. Deve-se, ainda, prover uma descrição do objetivo de cada consulta, assim como uma pequena amostra do resultado, ou seja, um conjunto de linhas recuperadas a partir da consulta.

1.7.1 Primeira Consulta - Gastos com Salário por Departamento

Esta consulta tem o objetivo de mostrar os gastos salariais por departamento, assim como informações sobre o número de empregados que trabalham em cada um, para que os gestores possam ter uma boa noção da organização e alocação de valores dentro da empresa.

Dessa forma, a seguinte query consta com duas funções de agregação: **SUM** e **COUNT**, para agregar o total de custos salariais e a contagem de empregados, respectivamente, sendo agrupados pelo nome do departamento.

Também temos um **LEFT JOIN** entre *departments* e *employees*, com o objetivo de obter as informações de todos os empregados que trabalham em um certo departamento. Isso é explicitado na cláusula ***ON e.department_id = d.department_id***.

```
SELECT d.department_name AS Department_Name,
       SUM(e.salary) AS Total_Salary_Expenses,
       COUNT(e.employee_id) AS Number_Of_Employees
FROM departments AS d
LEFT JOIN employees AS e ON e.department_id = d.department_id
GROUP BY d.department_name;
```

Um resultado relacionado a essa *query* pode ser obtido ao fazer um **GET** no endereço */api/departments/total-expenses*. Um exemplo de resultado é mostrado abaixo, em formato *json*.


```
[
    {
        "department_name": "Marketing",
        "total_salary_expenses": "18500",
        "number_of_employees": "3"
    },
    {
        "department_name": "Finance",
        "total_salary_expenses": "25500",
        "number_of_employees": "2"
    },
    {
        "department_name": "Human Resources",
        "total_salary_expenses": "509500",
        "number_of_employees": "14"
    },
    {
        "department_name": "IT",
        "total_salary_expenses": "52500",
        "number_of_employees": "3"
    }
]
```


Sendo *total_salary_expenses* o total gasto em salários em um dado departamento, e *number_of_employees* o número total de funcionários trabalhando sob aquele departamento.

1.7.2 Segunda Consulta - Alocações de Fundos e Funcionários em Projetos

Esta consulta tem como objetivo obter o valor monetário alocado a cada um dos projetos, assim como o número de funcionários operando em cada um deles. A intenção é providenciar aos coordenadores e executivos da empresa maneiras de visualizar a alocação de recursos para diferentes objetivos, de modo a melhor verificar se os mesmos estão sendo usados corretamente, ou se podem ser realocados a outros projetos de forma segura. De certa forma, é um complemento da consulta anterior, e as duas podem ser utilizadas em conjunto para obter informações importantes acerca dos recursos e funcionários da empresa.

Na parte mais técnica, essa consulta utiliza dois **JOINs** para relacionar a tabela *projects* com as tabelas *project_assignments* e *employees*. Já que a tabela *project_assignments* representa uma relação ***many-to-one***, as cláusulas **JOIN** simplesmente retornam todos os funcionários que constam em um dado projeto, passando pela tabela da relação como intermediária.

Ademais, tem-se uma agregação do tipo **COUNT** para contar o número de funcionários alocados (*assignees*) a um dado projeto, sendo os mesmos agrupados lógicamnete pelo nome do projeto, e também pelos fundos alocados a um projeto, para garantir a consistência dos dados.

```
SELECT p.project_name AS Project_Name,
       p.budget AS Budget,
       COUNT(e.employee_id) AS Number_Of_Assignees
FROM projects AS p
JOIN project_assignments AS pa ON pa.project_id = p.project_id
JOIN employees AS e ON e.employee_id = pa.employee_id
GROUP BY p.project_name, p.budget;
```

Resultados dessa *query* podem ser visualizados ao fazer um **GET** no endereço */api/projects/info*. Um exemplo de resultado é demonstrado abaixo.


```
[
    {
        "project_name": "Employee Engagement Survey",
        "budget": 5000,
        "number_of_assignees": "3"
    },
    {
        "project_name": "Attracting New Investors",
        "budget": 150000,
        "number_of_assignees": "5"
    },
    {
        "project_name": "Internal Software Development",
        "budget": 300000,
        "number_of_assignees": "3"
    },
    {
        "project_name": "Website Redesign",
        "budget": 50000,
        "number_of_assignees": "2"
    },
    {
        "project_name": "Market Research",
        "budget": 150000,
        "number_of_assignees": "4"
    }
]
```


Onde *budget* é o valor total dedicado a um projeto em qualquer moeda, mas para fins de demonstração, podemos considerar que é o dólar. Já *number_of _assignees* representa o número de funcionários que trabalham naquele projeto.

1.7.3 Terceira Consulta - Obter Informações de Empregados e Suas Funções

Como última consulta, resolvemos apresentar um simples sumário dos dados de funcionários individuais e as funções desempenhadas pelos mesmos dentro da empresa. Isso proporciona a gestão e visualização de dados de um funcionário específico em função do que ele faz em seu dia a dia no trabalho.

Como pedido, essa consulta tem um **LEFT JOIN** como cláusula, além de um **JOIN** normal. Essas cláusulas associam funcionários específicos com suas respectivas funções, já que *employee_roles* é uma tabela representante de uma relação ***many-to-many*** entre *employees* e *roles*.

A função de agregação **STRING_AGG** simplesmente junta as ocorrências de *roles* em uma única linha de resposta, agrupando-as ao separá-las por vírgulas.

```
SELECT e.*, STRING_AGG(r.role_name, ', ') AS roles
FROM employees AS e
JOIN employee_roles AS er ON er.employee_id = e.employee_id
LEFT JOIN roles AS r ON er.role_id = r.role_id
GROUP BY e.employee_id;
```

Um exemplo pode ser obtido ao fazer uma requisição **GET** para endereço */api/employees/info*. Os resultados obtidos são mostrados abaixo.

Note que *role_name* indica o nome da função desempenhada pelo funcionário, ou mais de um nome, caso o funcionário tenha vários papéis dentro da empresa.


```
{
    "employee_id": 4,
    "department_id": 4,
    "name": "Marie Johnson",
    "salary": 22000,
    "date_of_birth": "1990-07-25T00:00:00.000Z",
    "gender": "Female",
    "contact_number": "555-0104",
    "email": "marie.jhn@example.com",
    "roles": "Manager"
},
{
    "employee_id": 10,
    "department_id": 2,
    "name": "Sara Connor",
    "salary": 20000,
    "date_of_birth": "1992-12-20T00:00:00.000Z",
    "gender": "Female",
    "contact_number": "555-0111",
    "email": "s.connor@example.com",
    "roles": "Administrator"
},
{
    "employee_id": 6,
    "department_id": 2,
    "name": "Johnathan Lima",
    "salary": 17500,
    "date_of_birth": "2001-08-01T00:00:00.000Z",
    "gender": "Male",
    "contact_number": "555-0106",
    "email": "jlima123@example.com",
    "roles": "Analyst"
},
{
    "employee_id": 2,
    "department_id": 2,
    "name": "Jane Smith",
```

```
    "salary": 15000,
    "date_of_birth": "1985-05-20T00:00:00.000Z",
    "gender": "Female",
    "contact_number": "555-0102",
    "email": "jane.smith@example.com",
    "roles": "Manager, Analyst, Administrator"
},
{
    "employee_id": 11,
    "department_id": 3,
    "name": "Raj Patel",
    "salary": 5500,
    "date_of_birth": "1995-09-30T00:00:00.000Z",
    "gender": "Male",
    "contact_number": "555-0112",
    "email": "raj.patel@example.com",
    "roles": "Developer"
},
{
    "employee_id": 9,
    "department_id": 1,
    "name": "Evan Peters",
    "salary": 45000,
    "date_of_birth": "1988-04-15T00:00:00.000Z",
    "gender": "Male",
    "contact_number": "555-0110",
    "email": "evan.peters@example.com",
    "roles": "Developer, Manager, Analyst"
},
{
    "employee_id": 3,
    "department_id": 3,
    "name": "Alice Johnson",
    "salary": 10000,
    "date_of_birth": "1990-07-25T00:00:00.000Z",
    "gender": "Female",
    "contact_number": "555-0103",
    "email": "alice.johnson@example.com",
    "roles": "Manager"
},
{
    "employee_id": 12,
    "department_id": 4,
    "name": "Luna Star",
    "salary": 7000,
    "date_of_birth": "1986-11-05T00:00:00.000Z",
    "gender": "Female",
    "contact_number": "555-0113",
    "email": "luna.star@example.com",
    "roles": "Developer"
},
{
    "employee_id": 5,
    "department_id": 1,
```

```
    "name": "Julie Smith",
    "salary": 34500,
    "date_of_birth": "1999-01-20T00:00:00.000Z",
    "gender": "Female",
    "contact_number": "555-0105",
    "email": "julsmith@example.com",
    "roles": "Developer, Manager"
},
{
    "employee_id": 7,
    "department_id": 4,
    "name": "Lian Bark",
    "salary": 3500,
    "date_of_birth": "2002-03-27T00:00:00.000Z",
    "gender": "Male",
    "contact_number": "555-0107",
    "email": "barklian@example.com",
    "roles": "Analyst"
},
{
    "employee_id": 1,
    "department_id": 1,
    "name": "John Doe",
    "salary": 50000,
    "date_of_birth": "1980-01-15T00:00:00.000Z",
    "gender": "Male",
    "contact_number": "555-0101",
    "email": "john.doe@example.com",
    "roles": "Manager, Developer, Analyst"
},
{
    "employee_id": 8,
    "department_id": 3,
    "name": "Oswald Guy",
    "salary": 3000,
    "date_of_birth": "2000-07-07T00:00:00.000Z",
    "gender": "Male",
    "contact_number": "555-0108",
    "email": "oswaldguy@example.com",
    "roles": "Developer"
}
```