UNIVERSIDADE DE BRASÍLIA

Faculdade do Gama

Sistemas de Banco de Dados 1

Pesquisa JOIN

Pesquisa sobre como funciona o comando JOIN e seus diferentes tipos

Ian Lucca Soares Mesquita - 211045140

Brasília, DF

Junções em SQL (JOINs)

O que é um JOIN em SQL?

Um JOIN em SQL é uma operação que permite combinar registros de duas ou mais tabelas em um banco de dados baseado em uma condição relacionada entre elas. O JOIN é essencial para consultas complexas onde os dados estão distribuídos em várias tabelas.

Vantagens dos JOINs

- Combinação de Dados Relacionados: Permite unir dados distribuídos em diferentes tabelas de maneira eficiente e lógica.
- Flexibilidade: Oferece várias opções para combinar tabelas com base em diferentes condições e requisitos de consulta.
- Redução de Redundância: Evita duplicação de dados ao normalizar tabelas e, posteriormente, combinar dados quando necessário.

Desvantagens dos JOINs

- Complexidade: Consultas JOIN podem se tornar complexas e difíceis de ler e manter, especialmente em bases de dados grandes e complexas.
- Desempenho: JOINs podem ser intensivos em termos de processamento, especialmente quando combinam grandes volumes de dados ou envolvem múltiplas tabelas.
- Erros de Resultados: JOINs mal formulados podem resultar em conjuntos de dados incorretos ou inesperados, como duplicação ou perda de dados.

Tipos de JOINs em SQL

1. INNER JOIN

O INNER JOIN retorna apenas os registros que possuem valores correspondentes em ambas as tabelas.

```
SELECT employees.name, departments.name
FROM employees
INNER JOIN departments
ON employees.department_id = departments.id;
```

```
Result of INNER JOIN:
+----+
| Name | Dept |
+----+
| John | HR |
| Jane | Finance |
| Alice | IT |
```

2. LEFT JOIN (ou LEFT OUTER JOIN)

O LEFT JOIN retorna todos os registros da tabela à esquerda e os registros correspondentes da tabela à direita. Se não houver correspondência, retorna NULL.

Ex.

```
SELECT employees.name, departments.name
FROM employees

LEFT JOIN departments

ON employees.department_id = departments.id;
```

```
Employees Table

+---+---+

| ID | Name | Dept | | ID | Name |

+---+----+

| 1 | John | 10 | | 10 | HR |

| 2 | Jane | 20 | | 20 | Finance |

| 3 | Alice | 30 | | 30 | IT |

| 4 | Bob | 50 | +---+-----+
```

```
Result of LEFT JOIN:
+-----+
| Name | Dept |
+-----+
| John | HR |
| Jane | Finance |
| Alice | IT |
| Bob | NULL |
+-----+
```

3. RIGHT JOIN (ou RIGHT OUTER JOIN)

O RIGHT JOIN retorna todos os registros da tabela à direita e os registros correspondentes da tabela à esquerda. Se não houver correspondência, retorna NULL.

Ex.

```
SELECT employees.name, departments.name
FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.id;
```

```
Result of RIGHT JOIN:
Employees Table
                             Departments Table
                                                          Dept
       Name
             Dept
                              ID |
                                     Name
                                                  John
                                                          | HR
  1 John
             10
                             | 10 | HR
                                                   Jane
                                                            Finance
  2 | Jane
             20
                              20 | Finance
                                                 Alice
                                                           IT
  3 | Alice
             30
                              30 | IT
                                                 NULL
                                                            Support
    Bob
               50
                              40 | Support
```

4. FULL JOIN (ou FULL OUTER JOIN)

O FULL JOIN retorna todos os registros quando há uma correspondência em uma das tabelas. Se não houver correspondência, retorna NULL para aquela tabela.

Ex.

```
SELECT employees.name, departments.name
FROM employees
FULL OUTER JOIN departments
ON employees.department_id = departments.id;
```

```
Employees Table
                            Departments Table
                            | ID |
ID Name
            Dept
                                    Name
      John
             1 10
                            | 10 | HR
      Jane
             20
                            20 | Finance
     Alice
             30
                            | 30 | IT
  4 | Bob
             | 50
                            | 40 | Support
```

5. CROSS JOIN

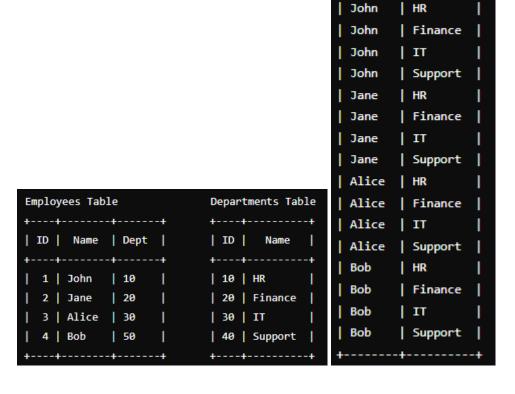
O CROSS JOIN retorna o produto cartesiano das duas tabelas, ou seja, combina cada registro de uma tabela com todos os registros da outra tabela.

Result of CROSS JOIN:

Dept

Ex.

```
SELECT employees.name, departments.name
FROM employees
CROSS JOIN departments;
```



6. SELF JOIN

O SELF JOIN é uma junção de uma tabela consigo mesma. É útil para comparar registros dentro da mesma tabela.

```
Employees Table
| ID | Name | ManagerID |
1 John
          2
 2 Jane
          NULL
 3 | Alice
          2
 4 Bob
Result of SELF JOIN:
+----+
| Employee1 | Employee2 |
John
         Jane
Alice
         Jane
Bob
         John
```

```
SELECT A.name AS Employee1, B.name AS Employee2
FROM employees A, employees B
WHERE A.manager_id = B.id;
```