UNIVERSIDADE DE BRASÍLIA
Faculdade do Gama

Sistemas de Banco de Dados 1

Trabalho Final (TF)

Junções em SQL *(JOINs)*

Marina Márcia Costa de Souza 200041606

Brasília, DF
2024

O JOIN é uma poderosa ferramenta no SQL utilizada para combinar registros de duas ou mais tabelas em um banco de dados relacional, com base em uma condição específica que relaciona colunas dessas tabelas. Essa cláusula permite a integração de informações de várias tabelas, possibilitando a recuperação de dados distribuídos e a realização de consultas complexas para obter informações integradas.

Vantagens e Desvantagens do JOIN:

Vantagens:

1. Combinação de Dados: Facilita a combinação e a análise de dados de várias tabelas.
2. Redução de Redundância: Ajuda a evitar a duplicação de dados, permitindo a normalização das tabelas.
3. Flexibilidade: Permite consultas complexas que podem fornecer insights detalhados.

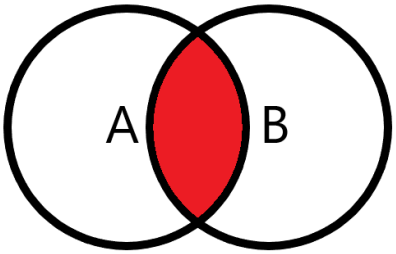
Desvantagens:

1. Complexidade: Consultas JOIN podem se tornar complexas, especialmente quando envolvem múltiplas tabelas.
2. Desempenho: JOINs podem afetar o desempenho em grandes bancos de dados se não forem otimizados corretamente.
3. Manutenção de Índices: Exige um planejamento cuidadoso de índices para otimizar a performance das consultas.

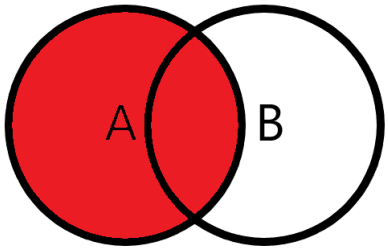
Tipos de JOIN:

Existem vários tipos de JOINs, cada um com um comportamento específico para combinar os registros das tabelas:

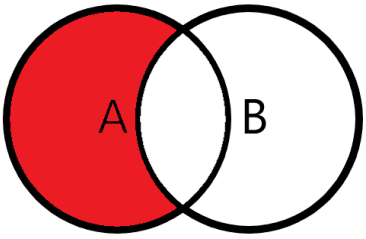
1. INNER JOIN: Retorna apenas os registros que têm correspondências em ambas as tabelas. Se não houver correspondência, os registros não são incluídos no resultado.
    a. Exemplo: SELEC * FROM E JOIN D



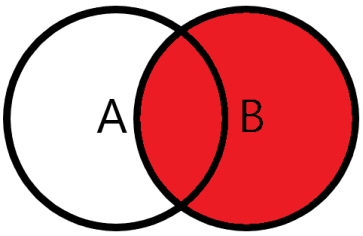
2. LEFT JOIN (ou LEFT OUTER JOIN): Retorna todos os registros da tabela à esquerda (tabela1), e os registros correspondentes da tabela à direita (tabela2). Se não houver correspondência, os valores das colunas da tabela à direita serão NULL.
    a. Exemplo: SELECT * FROM A LEFT JOIN A ON A.coluna = B.coluna



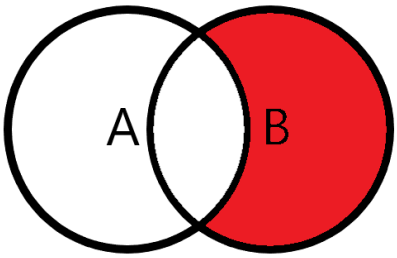
    b. Exemplo: SELECT * FROM A LEFT JOIN B ON A.coluna = B.coluna WHERE B.coluna IS NULL



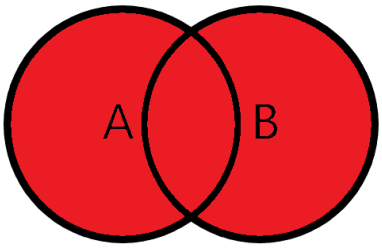
3. RIGHT JOIN (ou RIGHT OUTER JOIN): Retorna todos os registros da tabela à direita (tabela2), e os registros correspondentes da tabela à esquerda (tabela1). Se não houver correspondência, os valores das colunas da tabela à esquerda serão NULL.
    a. Exemplo: SELECT * FROM A RIGHT JOIN B ON A.coluna = B.coluna



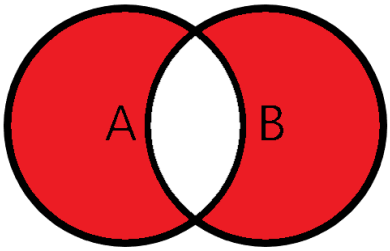
b. Exemplo: SELECT * FROM A RIGHT JOIN B ON A.coluna = B.coluna WHERE A.coluna IS NULL



4. FULL JOIN (ou FULL OUTER JOIN): Retorna todos os registros quando há uma correspondência em uma das tabelas. Se não houver correspondência, os valores das colunas da tabela sem correspondência serão NULL.
   a. Exemplo: SELECT * FROM D FULL JOIN E

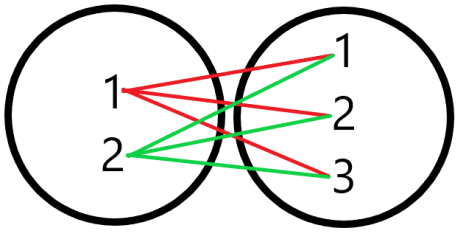


   b. Exemplo: SELECT * FROM D FULL JOIN E WHERE A.coluna OR B.coluna IS NULL

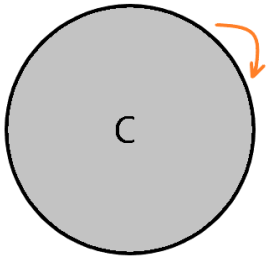


5. ROSS JOIN: Retorna o produto cartesiano das duas tabelas, combinando cada linha de uma tabela com todas as linhas da outra tabela.

a. Exemplo: SELECT * FROM D CROSS JOIN E



6. SELF JOIN: é uma autojunção, ou seja, a junção de uma tabela com ela mesma, utilizada para comparar linhas dentro da mesma tabela. Isso significa que cada linha de uma tabela é unida a si mesma e a todas as outras linhas dessa tabela, permitindo a realização de comparações e consultas complexas dentro da mesma tabela.



Um SELF JOIN funciona exatamente como um INNER JOIN, mas a junção é feita na mesma tabela. Ele retorna apenas as linhas que têm correspondência na condição especificada dentro da mesma tabela, essencialmente criando duas instâncias da tabela e unindo-as com base em uma condição.

a. Exemplo: SELECT nomes_colunas FROM Tabela1 t1 INNER JOIN Tabela1 t2 ON join_predicado