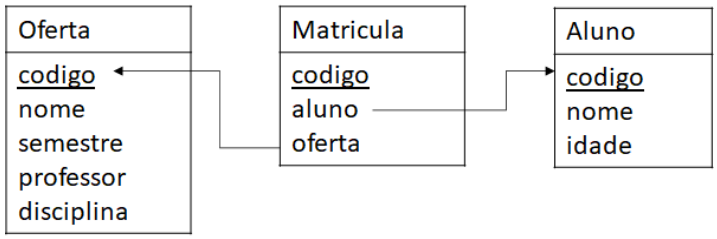
Junções Externas (OUTER JOINS) e Subconsultas

Junções Externas

Considere as tabelas abaixo representando a relação de matrícula entre alunos e ofertas de disciplinas.



Vamos supor que o conteúdo das tabelas é o seguinte:

Matrícula

codigo integer	aluno integer	oferta integer
1	1	1
3	4	1
4	2	3
5	4	3
2	2	1

Aluno

codigo [PK] integer	nome character varying (64)	idade integer
5	brandao	24
4	arlindo	27
3	marcos	18
2	maria	35
1	jose	80

Oferta

codigo [PK] integer	nome character varying (64)	semestre character varying (4)	professor character varying (64)	disciplina character varying (64)
3	biologia	2023	martha	biologia
2	portugues	2022	eduardo	portugues
1	matematica	2022	filipe	matematica

Queremos listar o número de ofertas em que cada aluno está matriculado. Inicialmente, poderíamos tentar alcançar este objetivo usando a query:

SELECT aluno.codigo, aluno.nome, **COUNT**(matricula.aluno) **FROM** aluno

INNER JOIN matricula **ON** matricula.aluno = aluno.codigo

GROUP BY aluno.codigo;

Contudo, o resultado da consulta não lista alunos que não se matricularam em ofertas. Isto acontece porque não existem matrículas cujo código é igual ao código do aluno.

codigo [PK] integer	nome character varying (64)	count bigint
4	arlindo	2
2	maria	2
1	jose	1

Junções externas (OUTER JOINS) garantem que todos as linhas de uma tabela serão retornadas, mesmo nos casos em que um casamento com outra tabela não existir. Existem três tipos de junções externas:

- **LEFT OUTER JOIN**: garante que todas as linhas da tabela à esquerda da junção são retornadas.
- **RIGHT OUTER JOIN**: garante que todas as linhas da tabela à direita da junção são retornadas.
- **FULL OUTER JOIN**: garante que todas as linhas de ambas as tabelas envolvidas na junção são retornadas.

***Nota**: Alguns SGBDs omitem a palavra OUTER nas junções externas.*

Podemos resolver o problema do número de ofertas por aluno usando o comando LEFT OUTER JOIN:

SELECT aluno.codigo, aluno.nome, **COUNT**(matricula.aluno) **FROM** aluno

LEFT OUTER JOIN matricula **ON** matricula.aluno = aluno.codigo

GROUP BY aluno.codigo;

Como pode ser observado abaixo, todos os alunos são retornados e aqueles que não possuem matrícula aparecem com contagem 0:

codigo [PK] integer	nome character varying (64)	count bigint
5	brandao	0
4	arlindo	2
2	maria	2
1	jose	1
3	marcos	0

De forma similar, poderíamos listar o número de estudantes em cada oferta usando o comando RIGHT OUTER JOIN:

SELECT oferta.codigo, oferta.nome, **COUNT**(matricula.oferta) **FROM** matricula

RIGHT OUTER JOIN oferta **ON** matricula.oferta = oferta.codigo

GROUP BY oferta.codigo;

A consulta traz todos as ofertas e aquelas que não possuem alunos cadastrados aparecem com contagem igual a zero:

codigo [PK] integer	nome character varying (64)	count bigint
3	biologia	2
2	portugues	0
1	matematica	3

Nota: Os comandos LEFT OUTER JOIN e RIGHT OUTER JOIN podem ser substituídos entre si desde que a ordem das tabelas na junção também seja invertida.

Subconsultas (Subqueries)

Exemplo extraído de: https://learnsql.com/blog/sql-subquery-examples/

Subconsultas são consultas SELECT que possuem em sua construção outras consultas SELECT. Abaixo são listados alguns exemplos de uso do recurso.

Contexto: Digamos que administramos uma galeria de arte. Temos um banco de dados com quatro tabelas: pinturas, artistas, colecionadores e vendas. Você pode ver os dados armazenados em cada tabela abaixo.

Exemplo 1 – Pinturas com Preço Acima da Média

First need to find this average price; here's where the scalar subquery comes into play:

```
SELECT name, listed_price
FROM paintings
WHERE listed_price > (
   SELECT AVG(listed_price)
   FROM paintings
);
```

Exemplo 2 - Colecionadores que Compraram Pinturas

Agora vamos examinar as subconsultas que retornam uma coluna com várias linhas. Essas subconsultas geralmente são incluídas na cláusula WHERE para filtrar os resultados da consulta principal.

Suponha que queremos listar todos os colecionadores que compraram pinturas de nossa galeria. Podemos obter a saída necessária usando uma subconsulta de várias linhas. Especificamente, podemos usar uma consulta interna para listar todos os IDs de colecionadores presentes na tabela de vendas – estes seriam IDs correspondentes aos colecionadores que fizeram pelo menos uma compra em nossa galeria. Em seguida, na consulta externa, solicitamos o nome e o sobrenome de todos os coletores cujo ID está na saída da consulta interna.

```
SELECT first_name, last_name
FROM collectors
WHERE id IN (
   SELECT collector_id
```

```
    FROM sales
);
```

Exemplo 3 – Uso de Subqueries em FROM e JOIN

Quando uma subconsulta retorna uma tabela com várias linhas e várias colunas, essa subconsulta geralmente é encontrada na cláusula FROM ou JOIN. Isso permite que você obtenha uma tabela com dados que não estavam prontamente disponíveis no banco de dados (por exemplo, dados agrupados) e, em seguida, junte essa tabela com outra do seu banco de dados, se necessário.

Digamos que queremos ver o valor total das vendas de cada artista que vendeu pelo menos uma pintura em nossa galeria. Podemos começar com uma subconsulta que se baseia na tabela de vendas e calcula o valor total das vendas para cada ID de artista. Então, na consulta externa, combinamos essas informações com os nomes e sobrenomes dos artistas para obter a saída necessária:

```
SELECT
  artists.first_name,
  artists.last_name,
  artist_sales.sales
FROM artists
JOIN (
    SELECT artist_id, SUM(sales_price) AS sales
    FROM sales
    GROUP BY artist_id
  ) AS artist_sales
  ON artists.id = artist_sales.artist_id;
```

Atribuímos um *alias* significativo à saída de nossa subconsulta (artist_sales). Dessa forma, podemos facilmente nos referir a ela na consulta externa, ao selecionar a coluna desta tabela e ao definir a condição de junção na cláusula ON. Observação: os bancos de dados gerarão um erro se você não fornecer um alias para a saída da sua subconsulta.

artists

id	first_name	last_name
1	Thomas	Black
2	Kate	Smith
3	Natali	Wein
4	Francesco	Benelli

collectors

id	first_name	last_name
101	Brandon	Cooper
102	Laura	Fisher
103	Christina	Buffet
104	Steve	Stevenson

paintings

id	name	artist_id	listed_price
11	Miracle	1	300.00
12	Sunshine	1	700.00
13	Pretty woman	2	2800.00
14	Handsome man	2	2300.00
15	Barbie	3	250.00
16	Cool painting	3	5000.00
17	Black square #1000	3	50.00
18	Mountains	4	1300.00

sales

id	date	painting_id	artist_id	collector_id	sales_price
1001	2021-11-01	13	2	104	2500.00
1002	2021-11-10	14	2	102	2300.00
1003	2021-11-10	11	1	102	300.00
1004	2021-11-15	16	3	103	4000.00
1005	2021-11-22	15	3	103	200.00
1006	2021-11-22	17	3	103	50.00

Parte da explicação e dos exemplos foram extraídas dos seguintes links:

[1] https://www.devmedia.com.br/inner-cross-left-rigth-e-full-joins/21016

[2] https://learnsql.com/blog/sql-subquery-examples/