Consulta sobre múltiplas relações

BCD29008 – Engenharia de Telecomunicações

Prof. Emerson Ribeiro de Mello

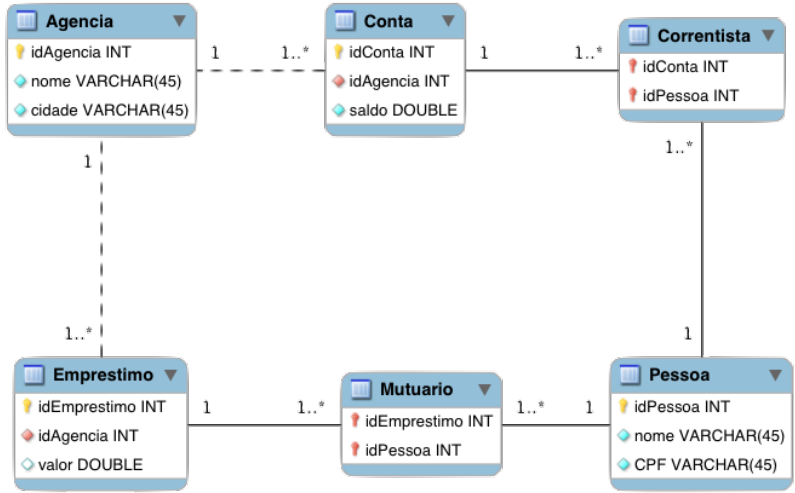
mello@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Campus São José

Licenciamento

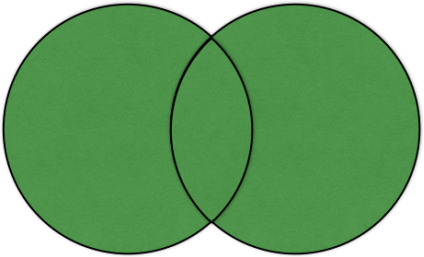
Esquema usado nos próximos exemplos

Disponível em https://emersonmello.me/ensino/bcd/labs/correntista-mutuario.sql



Operações com conjuntos

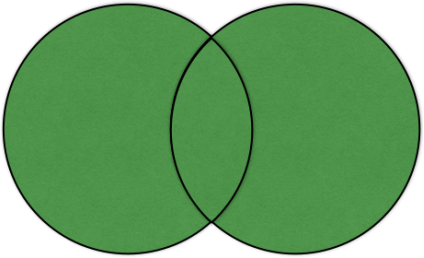
Operador UNION



```
(SELECT ... ) UNION (SELECT ...)
```

- Cada consulta deverá possuir o mesmo número de colunas e devem estar na mesma ordem

Operador UNION



```
(SELECT ... ) UNION (SELECT ...)
```

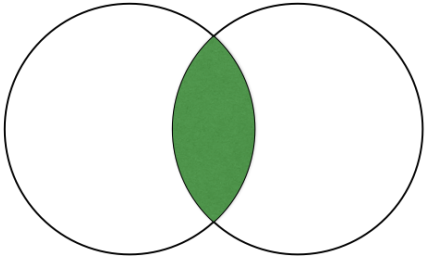
- Cada consulta deverá possuir o mesmo número de colunas e devem estar na mesma ordem

Listar o ID de todos os clientes que possuam um empréstimo, uma conta ou ambos

```
(SELECT idPessoa FROM Correntista) UNION (SELECT idPessoa FROM Mutuario)
```

Operador INTERSECT

MySQL não possui esse operador

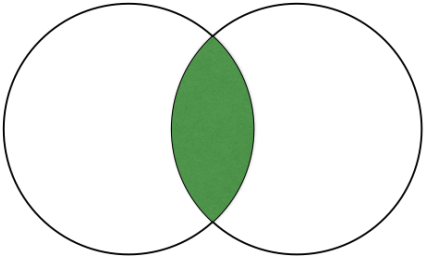


```
(SELECT ... ) INTERSECT (SELECT ...)
```

- Cada consulta deverá possuir o mesmo número de colunas e devem estar na mesma ordem

Operador INTERSECT

MySQL não possui esse operador



```
(SELECT ... ) INTERSECT (SELECT ...)
```

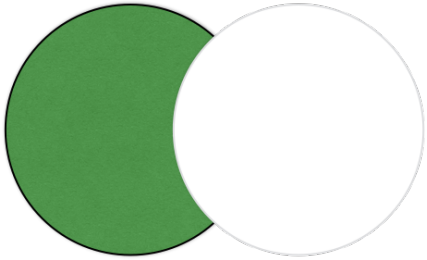
- Cada consulta deverá possuir o mesmo número de colunas e devem estar na mesma ordem

Listar o ID de todos os clientes que são correntistas e que possuam um empréstimo

```
(SELECT idPessoa FROM Correntista) INTERSECT (SELECT idPessoa FROM Mutuario)
```

Operador EXCEPT

MySQL não possui esse operador

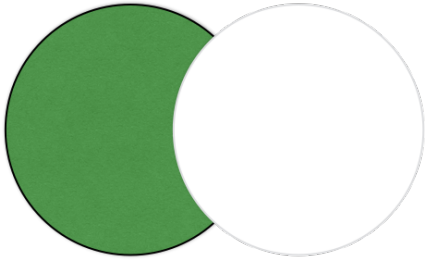


```
(SELECT ... ) EXCEPT (SELECT ...)
```

- Cada consulta deverá possuir o mesmo número de colunas e devem estar na mesma ordem

Operador EXCEPT

MySQL não possui esse operador



```
(SELECT ... ) EXCEPT (SELECT ...)
```

- Cada consulta deverá possuir o mesmo número de colunas e devem estar na mesma ordem

Listar o ID de todos os clientes que são correntistas e que não possuam um empréstimo

```
(SELECT idPessoa FROM Correntista) EXCEPT (SELECT idPessoa FROM Mutuario)
```

Operador ALL

- O comportamento padrão do operador `UNION` é de remover tuplas duplicadas da relação resultante
- Operador ALL permite tuplas duplicadas na relação resultante

```
(SELECT idPessoa FROM Correntista)
  UNION ALL (SELECT idPessoa FROM Mutuario)
```

Operador IN

- Operador `IN` permite verificar se o valor de um atributo está contido em um conjunto

```
SELECT idAgencia, nome FROM Agencia
  WHERE cidade IN ('São José', 'Florianópolis');
```

Operador IN

- Operador `IN` permite verificar se o valor de um atributo está contido em um conjunto

```
SELECT idAgencia, nome FROM Agencia
  WHERE cidade IN ('São José', 'Florianópolis');
```

- Operador IN pode ser usado para obter o comportamento o `INTERSECT`. (veja subconsultas aninhadas)
- Operador `EXISTS` pode ser usado para obter o comportamento do `EXCEPT`

Valores nulos

Valores nullos (NULL)

Valor do atributo é desconhecido ou ainda não existe

- Listar os empréstimos que tenham nulo no atributo valor

```
SELECT idEmprestimo FROM Emprestimo WHERE valor IS NULL
```

- Resultado de operações aritméticas com nulo sempre será nulo

```
SELECT 5 + NULL; -- retorna NULL
```

- Qualquer comparação relacional com valores nulos sempre resultará em nulo

```
SELECT (123 > NULL);
SELECT (NULL = 456);
SELECT (NULL <> NULL);
```

Valores nullos (NULL)

Operadores lógicos

Operador lógico OR

```
SELECT (NULL OR TRUE); -- TRUE
SELECT (NULL OR FALSE); -- NULL
SELECT (NULL OR NULL); --  NULL
```

Operador lógico AND

```
SELECT (NULL AND TRUE); -- NULL
SELECT (NULL AND FALSE); -- FALSE
SELECT (NULL AND NULL); --  NULL
```

Operador lógico NOT

```
SELECT (NOT NULL); -- NULL
```

Valores nullos (NULL)

- Operações de agregação (i.e. SUM, COUNT) ignoram nulos, com exceção do COUNT(*)

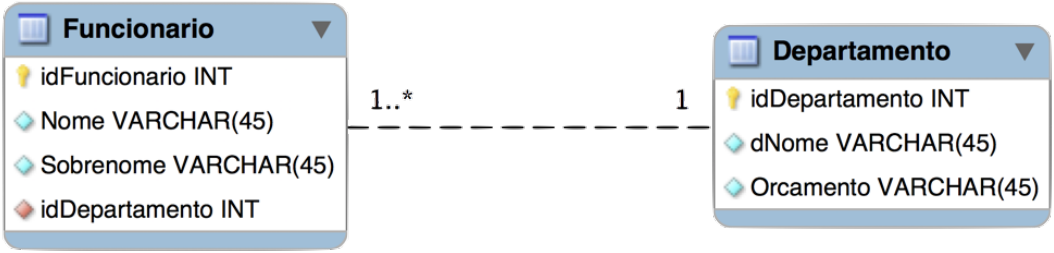
```
-- total de linhas cujo valor no atributo cidade não seja NULO
SELECT COUNT(cidade) FROM Agencia;

-- total de linhas da relação
SELECT COUNT(*) FROM Agencia;
```

Consulta sobre múltiplas relações

Esquema usado nos próximos exemplos

Disponível em https://emersonmello.me/ensino/bcd/labs/funcionario-departamento.sql



Como obter o nome e sobrenome de todos os funcionários, juntamente com os nomes dos departamentos onde estão lotados?

idFuncionario	Nome	Sobrenome	idDepartamento
123	Julio	Silva	1
326	João	Silveira	2
331	George	de la Rocha	3
332	José	Oliveira	1

idDepartamento	dNome	Orcamento
1	Financeiro	15000
2	TI	60000
3	Gestão de Pessoas	150000

Selecionando colunas de duas relações

Produto cartesiano

```
SELECT Nome, Sobrenome, dNome
  FROM Funcionario, Departamento;

+-----------+-------------+----------------------------+
| Nome      | Sobrenome   | dNome                      |
+-----------+-------------+----------------------------+
| Julio     | Silva       | Financeiro                 |
| Julio     | Silva       | TI                         |
| Julio     | Silva       | Gestão de Pessoas          |
| Arnaldo   | Coelho      | Financeiro                 |
.....
```

Produto cartesiano gera perda de informação. Ao fazer junção de relações é necessário combinar chave primária com chave estrangeira

Junção natural

NATURAL JOIN

```
SELECT Nome, Sobrenome, dNome
  FROM Funcionario NATURAL JOIN Departamento;

+-----------+-------------+--------------------+
| Nome      | Sobrenome   | dNome              |
+-----------+-------------+--------------------+
| Julio     | Silva       | Financeiro         |
| Arnaldo   | Coelho      | Financeiro         |
| George    | de la Rocha | Gestão de Pessoas  |
...
```

- Requer uma coluna com nome comum em todas as relações e combina todas as tuplas que possuem valores iguais na coluna de nome comum
- No exemplo, as relações `Funcionário` e `Departamento` possuem uma coluna com o mesmo nome "`idDepartamento`"

Junção interna

INNER JOIN, ou simplesmente JOIN

- Pode ser usada quando a chave estrangeira, na tabela referenciadora, não possuir o mesmo nome da coluna chave primária da tabela referenciada

```
SELECT f.Nome, f.Sobrenome, d.dNome
  FROM Funcionario f
    INNER JOIN Departamento d ON f.idDepartamento = d.idDepartamento

+-----------+-------------+--------------------+
| Nome      | Sobrenome   | dNome              |
+-----------+-------------+--------------------+
| Julio     | Silva       | Financeiro         |
| Arnaldo   | Coelho      | Financeiro         |
| George    | de la Rocha | Gestão de Pessoas  |
...
```

Junção interna

INNER JOIN, ou simplesmente JOIN

- Retorna a coluna idDepartamento uma única vez

```
SELECT * FROM Funcionario NATURAL JOIN Departamento
```

- Retorna a coluna idDepartamento duas vezes, uma de cada relação

```
SELECT * FROM Funcionario f
   INNER JOIN Departamento d ON f.idDepartamento = d.idDepartamento
```

```
SELECT * FROM Funcionario f, Departamento d
   WHERE f.idDepartamento = d.idDepartamento
```

Usar ON ou WHERE?

Na cláusula FROM a *vírgula* é equivalente ao JOIN

- Em junções internas pode-se usar a condição ON ou WHERE

```
SELECT f.Nome, d.dNome FROM Funcionario f JOIN Departamento d
   ON f.idDepartamento = d.idDepartamento

SELECT f.Nome, d.dNome FROM Funcionario f, Departamento d
   WHERE f.idDepartamento = d.idDepartamento;
```

- Fica mais claro se usar ON para junção e as seleções de tuplas no WHERE

```
SELECT f.Nome, d.dNome FROM Funcionario f JOIN Departamento d
  ON f.idDepartamento = d.idDepartamento
    WHERE Orcamento > 13500;
```

Usar ON ou WHERE?

Na cláusula FROM a *vírgula* é equivalente ao `JOIN`

- Em junções internas pode-se usar a condição `ON` ou `WHERE`

```
SELECT f.Nome, d.dNome FROM Funcionario f JOIN Departamento d
   ON f.idDepartamento = d.idDepartamento

SELECT f.Nome, d.dNome FROM Funcionario f, Departamento d
   WHERE f.idDepartamento = d.idDepartamento;
```

- Fica mais claro se usar `ON` para junção e as seleções de tuplas no `WHERE`

```
SELECT f.Nome, d.dNome FROM Funcionario f JOIN Departamento d
  ON f.idDepartamento = d.idDepartamento
    WHERE Orcamento > 13500;
```

Como reescrever a instrução acima usando somente WHERE?

Junções externas

OUTER JOIN

- Com a junção natural não serão exibidos os departamentos que não possuírem um funcionário lotado

```
SELECT * FROM Departamento NATURAL JOIN Funcionario;
```

Junções externas

OUTER JOIN

- Com a junção natural não serão exibidos os departamentos que não possuírem um funcionário lotado

```
SELECT * FROM Departamento NATURAL JOIN Funcionario;
```

Junções externas

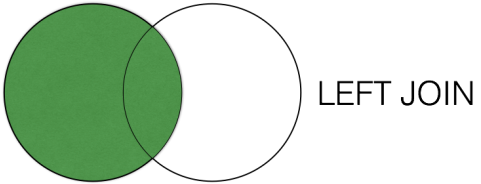
Preserva as tuplas que seriam perdidas em um junção, criando na relação resultante tuplas com colunas contendo valores nulos

- **Left outer join** – preserva somente as tuplas da relação à esquerda
- **Right outer join** – preserva somente as tuplas da relação à direita
- **Full outer join** – preserva as tuplas das duas relações

Junção externa

LEFT OUTER JOIN

Retorna todas as tuplas da relação à esquerda, mesmo aquelas que não possuam correspondentes na tabela à direita



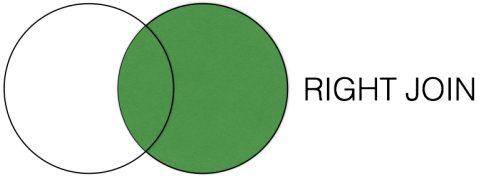

```sql
SELECT * FROM Departamento d
  LEFT JOIN Funcionario f ON d.idDepartamento = f.idDepartamento;

-- Ou também com NATURAL
SELECT * FROM Departamento NATURAL LEFT JOIN Funcionario;
```

Junção externa

RIGHT OUTER JOIN

Retorna todas as tuplas da relação à direita, mesmo aquelas que não possuam correspondentes na tabela à esquerda



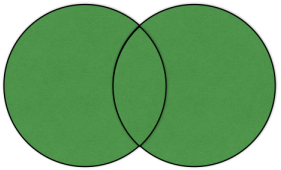

```sql
SELECT * FROM Departamento d
  RIGHT JOIN Funcionario f ON d.idDepartamento = f.idDepartamento;

-- Ou também com NATURAL
SELECT * FROM Departamento NATURAL RIGHT OUTER JOIN Funcionario;
```

Junção externa

FULL OUTER JOIN

Retorna todas as tuplas de ambas tabelas, mesmo se não houver correspondência de valores




```
SELECT * FROM Departamento d
  NATURAL FULL OUTER JOIN Funcionario f;
```

- MySQL não implementa FULL JOIN

Exercícios

Funcionário e Departamento

https://emersonmello.me/ensino/bcd/labs/funcionario-departamento.sql

1. Liste todos os dados de todos os funcionários, inclusive todos os dados de seus departamentos
2. Liste o nome do funcionário e o nome do departamento onde cada funcionário está lotado
3. Liste o nome e sobrenome de todos os funcionários que estejam lotados em departamentos com orçamento maior que $60.000,00$
4. Liste os nomes de todos os departamentos que possuam mais de dois funcionários

Funcionário e Departamento

https://emersonmello.me/ensino/bcd/labs/funcionario-departamento.sql

1. Liste todos os dados de todos os funcionários, inclusive todos os dados de seus departamentos

```
SELECT F.*, D.* FROM Funcionario F INNER JOIN Departamento D
  ON F.idDepartamento = D.idDepartamento;
```

2. Liste o nome do funcionário e o nome do departamento onde cada funcionário está lotado

3. Liste o nome e sobrenome de todos os funcionários que estejam lotados em departamentos com orçamento maior que $60.000,00$

4. Liste os nomes de todos os departamentos que possuam mais de dois funcionários

Funcionário e Departamento

https://emersonmello.me/ensino/bcd/labs/funcionario-departamento.sql

1. Liste todos os dados de todos os funcionários, inclusive todos os dados de seus departamentos

2. Liste o nome do funcionário e o nome do departamento onde cada funcionário está lotado

```
SELECT F.Nome, D.dNome FROM Funcionario F
  INNER JOIN Departamento D
    ON F.idDepartamento = D.idDepartamento;
```

3. Liste o nome e sobrenome de todos os funcionários que estejam lotados em departamentos com orçamento maior que $60.000,00$

4. Liste os nomes de todos os departamentos que possuam mais de dois funcionários

Funcionário e Departamento

https://emersonmello.me/ensino/bcd/labs/funcionario-departamento.sql

1. Liste todos os dados de todos os funcionários, inclusive todos os dados de seus departamentos

2. Liste o nome do funcionário e o nome do departamento onde cada funcionário está lotado

3. Liste o nome e sobrenome de todos os funcionários que estejam lotados em departamentos com orçamento maior que $60.000,00$

```
SELECT F.Nome, F.Sobrenome FROM Funcionario F
  INNER JOIN Departamento D
    ON F.idDepartamento = D.idDepartamento
      WHERE D.Orcamento > 60000;
```

4. Liste os nomes de todos os departamentos que possuam mais de dois funcionários

Funcionário e Departamento

https://emersonmello.me/ensino/bcd/labs/funcionario-departamento.sql

1. Liste todos os dados de todos os funcionários, inclusive todos os dados de seus departamentos
2. Liste o nome do funcionário e o nome do departamento onde cada funcionário está lotado
3. Liste o nome e sobrenome de todos os funcionários que estejam lotados em departamentos com orçamento maior que $60.000,00$
4. Liste os nomes de todos os departamentos que possuam mais de dois funcionários

```
SELECT D.dNome FROM Funcionario F INNER JOIN Departamento D
  ON D.idDepartamento = F.idDepartamento
    GROUP BY D.dNome HAVING COUNT(*) > 2
```

Subconsultas aninhadas

Subconsultas aninhadas

- Trata-se de de uma consulta aninhada a uma instrução `SELECT`, `INSERT`, `UPDATE` ou `DELETE` e resultado gerado é usado pela instrução onde essa está aninhada
- Geralmente é usada em operações para verificar se uma tupla pertence ao conjunto, para comparar conjuntos e verificar a cardinalidade de conjuntos

Subconsultas aninhadas

- Trata-se de de uma consulta aninhada a uma instrução `SELECT`, `INSERT`, `UPDATE` ou `DELETE` e resultado gerado é usado pela instrução onde essa está aninhada
- Geralmente é usada em operações para verificar se uma tupla pertence ao conjunto, para comparar conjuntos e verificar a cardinalidade de conjuntos

Listar o nome e orçamento do departamento que possui o maior orçamento

```
SELECT dNome, Orcamento FROM Departamento
  WHERE Orcamento = (SELECT MAX(Orcamento) FROM Departamento)
```

Subconsultas aninhadas

Para todo aluno que estiver com `idCurso = 2`, substituir pelo código pertencente ao curso chamado “Engenharia de Telecomunicações”

```
Aluno(idAluno, nome, idCurso)
Curso(idCurso, cNome)

UPDATE Aluno
  SET idCurso = (SELECT idCurso FROM Curso
                        WHERE cNome = 'Engenharia de Telecomunicações')
  WHERE idCurso = 2;
```

Para atribuição de valor na coluna `codigoCurso` a subconsulta deverá resultar uma única linha e coluna (subconsulta escalar)

Diferentes consultas SQL podem apresentar o mesmo resultado

Liste o nome de todos os funcionários de todos os departamentos que possuam orçamento maior que 5.000

- Exemplo com INNER JOIN

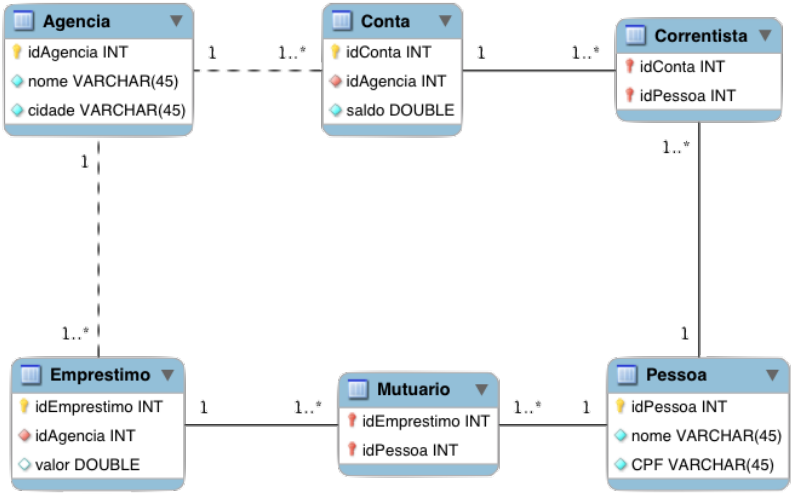
```
SELECT f.Nome FROM Funcionario f INNER JOIN Departamento d
  ON f.idDepartamento = d.Departamento
    WHERE d.Orcamento > 5000;
```

- Exemplo com subconsultas (operador IN)

```
SELECT Nome FROM Funcionario WHERE idDepartamento IN
  (SELECT idDepartamento FROM Departamento
     WHERE Orcamento > 5000);
```

Esquema usado nos próximos exemplos

Disponível em https://emersonmello.me/ensino/bcd/labs/correntista-mutuario.sql



Exemplos com operador IN

- Listar o ID de todos os clientes que possuam uma conta e um empréstimo

```
SELECT DISTINCT idPessoa FROM Mutuario
  WHERE idPessoa IN (SELECT idPessoa FROM Correntista)
```

- Listar o ID de todos os clientes que possuam um empréstimo, mas que não possuam uma conta

```
SELECT DISTINCT idPessoa FROM Mutuario
  WHERE idPessoa NOT IN (SELECT idPessoa FROM Correntista)
```

Exercício com operador IN

Liste o nome de todas as disciplinas ministradas no segundo semestre de 2014 e no primeiro semestre de 2017

```
Disciplina(idDisc, nome, ano, semestre)
```

Exercício com operador IN

Liste o nome de todas as disciplinas ministradas no segundo semestre de 2014 e no primeiro semestre de 2017

```
Disciplina(idDisc, nome, ano, semestre)
```

```
SELECT DISTINCT nome FROM Disciplina
    WHERE semestre = 2 AND ano = 2014
    AND nome IN (SELECT nome FROM Disciplina WHERE semestre = 1 AND ano = 2017)
```

Exercício com operador IN

Liste o CPF dos presidentes das empresas que fabricam produtos nos estados de SP e SC (fabricar em ambos)

```
Produto(idProduto, idEmpresa, ufFabrica)
Empresa(idEmpresa, cpfPresidente)
```

Exercício com operador IN

Liste o CPF dos presidentes das empresas que fabricam produtos nos estados de SP e SC (fabricar em ambos)

```
Produto(idProduto, idEmpresa, ufFabrica)
Empresa(idEmpresa, cpfPresidente)
```

```
SELECT DISTINCT cpfPresidente FROM Empresa e
    WHERE
    e.idEmpresa IN (SELECT idEmpresa FROM Produto WHERE UFFabrica = 'SP')
    AND
    e.idEmpresa IN (SELECT idEmpresa FROM Produto WHERE UFFabrica = 'SC')
```

Subconsultas com a cláusula FROM

Soma dos valores médios de empréstimos realizados por cada agência

```
-- Obtendo o valor médio agrupado por agência
SELECT AVG(valor) AS valorMedio FROM Emprestimo GROUP BY idAgencia;
+------------+
| valorMedio |
+------------+
|        750 |
|       1000 |
|       3000 |
+------------+
```

```
-- instrução inválida!
SELECT SUM(AVG(valor)) AS valorMedio
  FROM Emprestimo
  GROUP BY idAgencia;
```

```
SELECT SUM(valorMedio)
  FROM (SELECT AVG(valor) AS valorMedio
          FROM Emprestimo
          GROUP BY idAgencia) AS Resultado;
```

Subconsultas

Operações com conjuntos

- **Pelo menos um (SOME)**
  - `> some`
  - `< some`
  - `>= some`
  - `<= some`
  - `= some`
  - `<> some`
- **Que todos (ALL)**
  - `> all`
  - `< all`
  - `>= all`
  - `<= all`
  - `= all`
  - `<> all`

Operações com conjuntos

```
Funcionario(idFuncionario, nome, cargo, salario)
```

- Listar nome, cargo e salário dos funcionários cujo salário **seja maior que o salário de pelo menos um** Analista

```
SELECT nome, cargo, salário
    FROM Funcionario
    WHERE  salario > SOME (SELECT salario FROM Funcionario
                           WHERE cargo = 'Analista');
```

- Listar nome, cargo e salário dos funcionários cujo salário **seja maior do que o salário de todos** os Analistas

```
SELECT nome, cargo, salário
    FROM Funcionario
    WHERE  salario > ALL (SELECT salario FROM Funcionario
                          WHERE cargo = 'Analista');
```

Cláusula `SOME`, pelo menos um

1
5
8

`(5 < SOME)`

Verdade

1
5
8

`(5 = SOME)`

Verdade

1
5
8

`(5 <> SOME)`

Verdade

1
2
5

`(5 < SOME)`

Falso

Equivalências

- `(= SOME)` $\equiv$ `IN`
- `(<> SOME)` $\not\equiv$ `NOT IN`

Cláusula `ALL`, que todos

1
5
8

`(5 < ALL)`

Falso

1
5
8

`(5 = ALL)`

Falso

1
6
8

`(5 <> ALL)`

Verdade

6
7
8

`(5 < ALL)`

Verdade

Equivalências

- `(<> ALL)` $\equiv$ `NOT IN`
- `(= ALL)` $\not\equiv$ `IN`

Teste de relação vazia com EXISTS

- Se a relação resultante da subconsulta for $\neq \emptyset$, então retorna TRUE, senão retorna FALSE

Liste o nome de todos as disciplinas ministradas no segundo semestre de 2014 e no primeiro semestre de 2017

```
Disciplina(nome, ano, semestre)

SELECT DISTINCT nome FROM Disciplina
  WHERE semestre = 2
    AND ano = 2014
    AND EXISTS (SELECT nome FROM Disciplina
                  WHERE semestre = 1 AND ano = 2017)
```

MySQL não possui INTERSECT e EXCEPT

- Listar o ID de todos os clientes que são correntistas e que possuam um empréstimo

```
(SELECT idPessoa FROM Correntista)
  INTERSECT
   (SELECT idPessoa FROM Mutuario)
```

```
SELECT idPessoa FROM Correntista c
 WHERE EXISTS
  (SELECT idPessoa FROM Mutuario m
   WHERE c.idPessoa = m.idPessoa);
```

- Listar o ID de todos os clientes que são correntistas e que não possuam um empréstimo

```
(SELECT idPessoa FROM Correntista)
  EXCEPT
   (SELECT idPessoa FROM Mutuario)
```

```
SELECT idPessoa FROM Correntista c
 WHERE NOT EXISTS
  (SELECT idPessoa FROM Mutuario m
   WHERE c.idPessoa = m.idPessoa);
```

Subconsulta escalar retorna uma única tupla com um único atributo

Subconsulta escalar vs GROUP BY

Liste o total em R$ das vendas de cada produto

```
Vendas(Produto, qtde, preco)
```

- Fazendo uso de subconsulta

```
SELECT e.Produto,
       (SELECT SUM(i.qtde * i.preco) FROM Vendas i
          WHERE e.Produto = i.Produto) AS TotalVendas
  FROM Vendas e
```

- Fazendo uso de `GROUP BY` (recomendada)

```
SELECT Produto, SUM(qtde * preco) AS TotalVendas
  FROM Vendas GROUP BY Produto
```

DELETE e múltiplas relações

Apagando linhas de uma ou mais tabelas

```
Aluno(idAluno, nome, idCurso)
 curso referencia Curso
Disciplina(idDisc, nome, idCurso)
 curso referencia Curso
Curso(idCurso, nome)
```

- Exclua, da tabela Curso, o curso cujo código seja igual a 123

```
DELETE FROM Curso WHERE idCurso = 123
```

- Exclua todas as linhas das tabelas Curso, Disciplina e Aluno cujo código do curso seja igual a 123

```
DELETE Curso, Disciplina, Aluno FROM Curso c
  JOIN Disciplina d
  JOIN Aluno a
    WHERE c.idCurso = 123 AND c.idCurso = d.idCurso
      AND c.idCurso = a.idCurso
```

Exemplos com NATURAL JOIN e subconsulta

```
Aluno(idAluno, nome, idCurso)
 curso referencia Curso
Curso(idCurso, nome)
```

- Exemplo com NATURAL JOIN

```
DELETE Aluno FROM Aluno NATURAL JOIN Curso
    WHERE Curso.Nome = 'Curso XYZ'
```

- Exemplo com subconsulta

```
DELETE Aluno FROM Aluno
  WHERE Aluno.idCurso IN
    (SELECT idCurso FROM Curso WHERE Curso.Nome = 'Curso XYZ')
```

Aulas baseadas em

Henry F.; Sudarshan Silberschatz, Abraham; Korth.
Sistemas de banco de dados.
6a. Edição - Editora Campus, 2012

Sullivan, D. G.
Computer Science – Harvard University