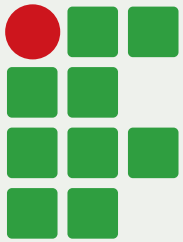
Consulta sobre múltiplas relações

BCD29008 – Engenharia de Telecomunicações

Prof. Emerson Ribeiro de Mello

mello@ifsc.edu.br

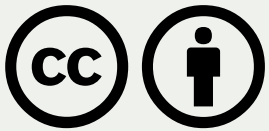
19 de maio de 2022



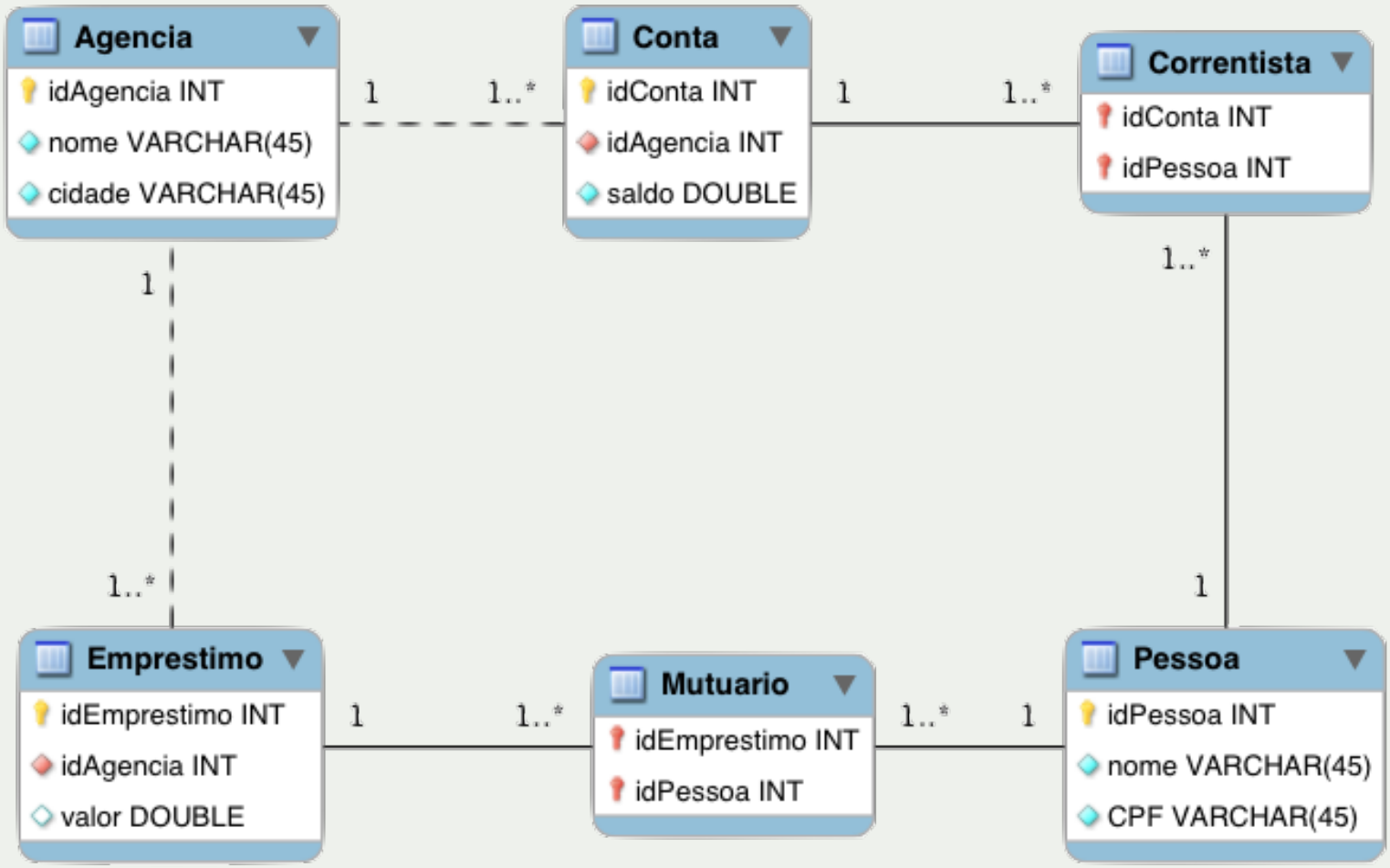
INSTITUTO FEDERAL
Santa Catarina

Câmpus
São José

Licenciamento

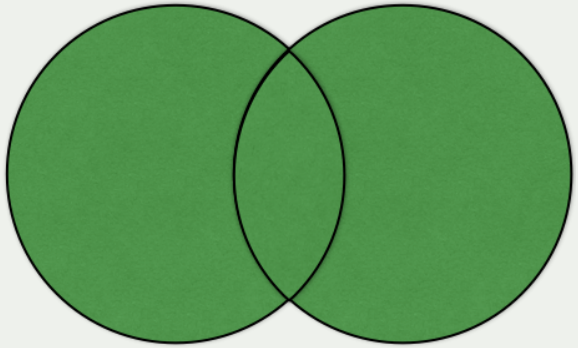


Esquema usado nos próximos exemplos



- Esquema MySQL disponível em:
  http://docente.ifsc.edu.br/mello/bcd/labs/lab04-banco.sql

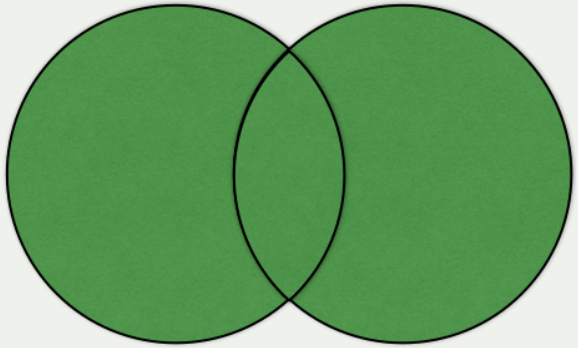
Operações com conjuntos
Operador UNION



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

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

Operações com conjuntos
Operador UNION



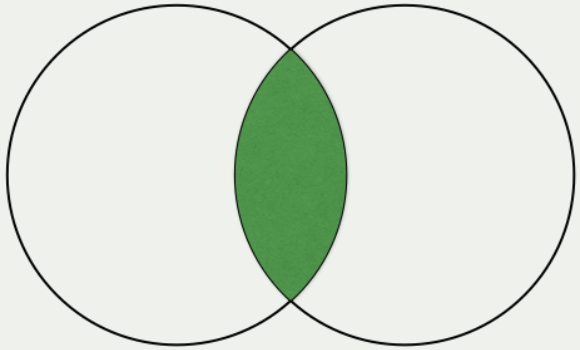
```
(SELECT ... ) UNION (SELECT ...)
```

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

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

Operações com conjuntos

Operador INTERSECT – (MySQL não possui esse operador)

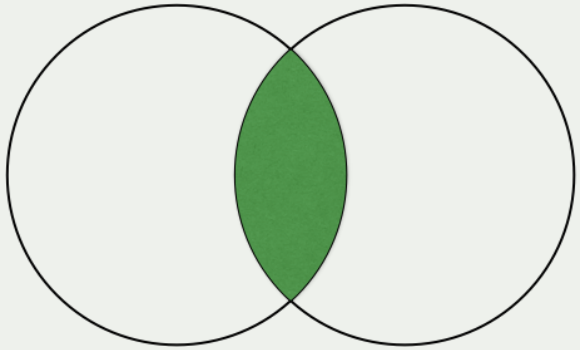


```
(SELECT ... ) INTERSECT (SELECT ...)
```

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

Operações com conjuntos

Operador INTERSECT – (MySQL não possui esse operador)



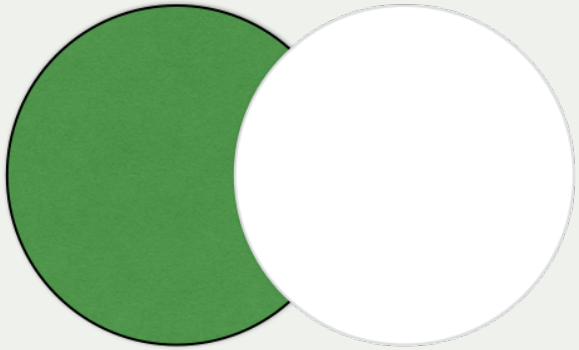
```
(SELECT ... ) INTERSECT (SELECT ...)
```

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

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

Operações com conjuntos

Operador EXCEPT – (MySQL não possui esse operador)

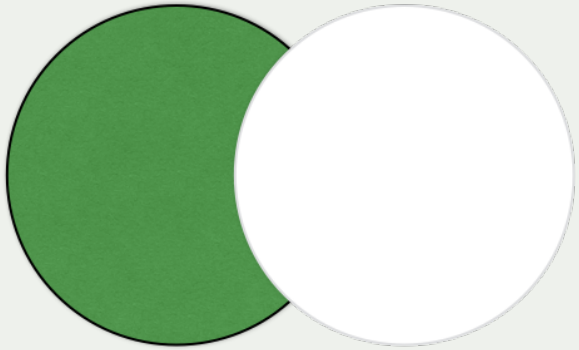


```
(SELECT ... ) EXCEPT (SELECT ...)
```

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

Operações com conjuntos

Operador EXCEPT – (MySQL não possui esse operador)



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

- 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)
```

Operações com conjuntos

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

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

  - Poderia ser usado para ter o comportamento o `INTERSECT`. (veja subconsultas aninhadas)
  - Operador `EXISTS` poderia ser usado para ter o comportamento do `EXCEPT`

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
```

- 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;
```


Valores nullos (NULL)

Operadores lógicos

- Qualquer comparação com valores nulos sempre resultará em nulo

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

- **Operador lógico OR**

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

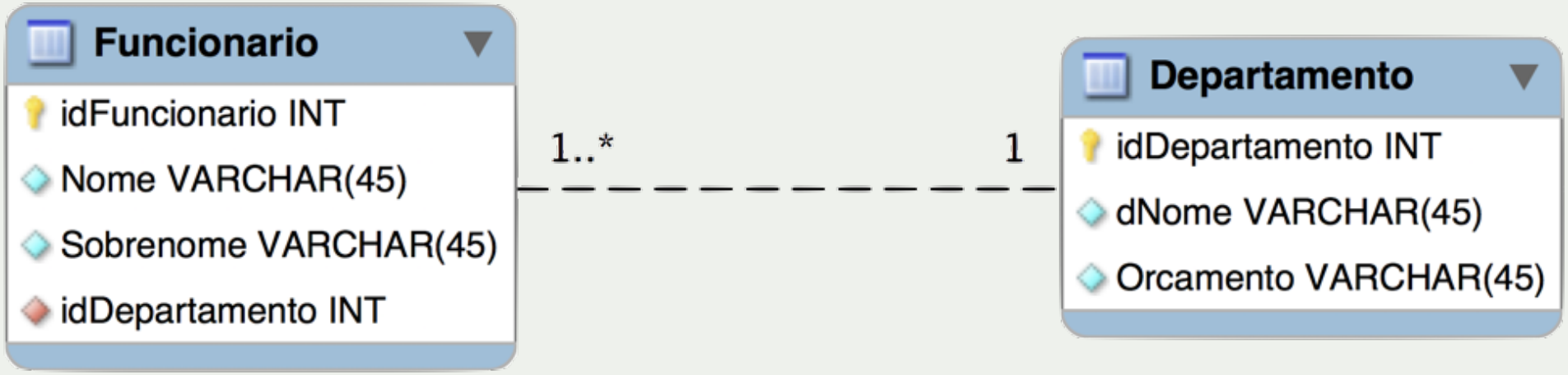
- **Operador lógico AND**

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

- **Operador lógico NOT**

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


Esquema usado nos próximos exemplos



Consulta sobre múltiplas relações

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

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

Consulta sobre múltiplas relações

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

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

```
for each tupla t1 in relação r1
  for each tupla t2 in relação r2
    ...
      for each tupla tm in relação rm
        Concatene t1, t2, ..., tm em uma única tupla t
        Adicione t à relação resultante
```

Condições de junção: NATURAL JOIN

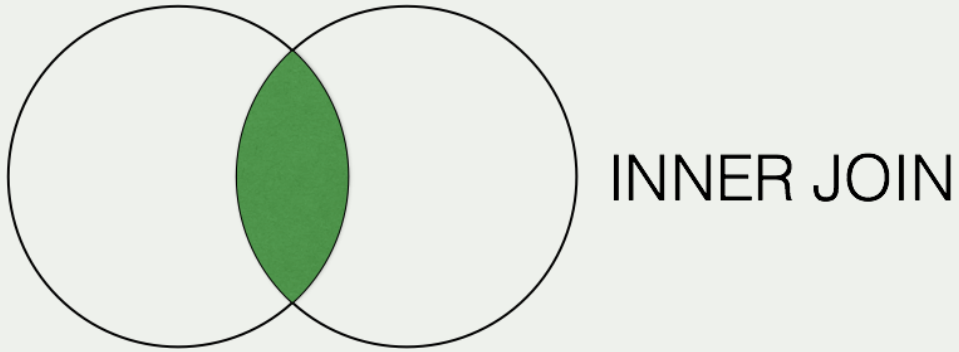
```
SELECT Nome, Sobrenome, dNome
  FROM Funcionario NATURAL JOIN Departamento;

+----------+------------+--------------------+
| 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
  - No exemplo, as relações `Funcionário` e `Departamento` possuem uma coluna chamada `idDepartamento`
- Combina todas as tuplas que possuem valores iguais na coluna de nome comum

Condições de junção: INNER JOIN - (ou simplesmente JOIN)

Retorna registros que possuam valores correspondentes em ambas tabelas




```sql
SELECT colunas
  FROM tabela1
    INNER JOIN tabela2 ON tabela1.colunaA = tabela2.colunaB;

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

Condições de junção: INNER 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
```

- Semelhante ao INNER JOIN, retorna a coluna idDepartamento duas vezes

```
SELECT * FROM Funcionario, Departamento
  WHERE Funcionario.idDepartamento = Departamento.idDepartamento;
```


Usar ON ou WHERE nas consultas com múltiplas relações?

- Em junções internas, pode-se usar a condição ON ou WHERE
  - Na cláusula FROM, a vírgula (,) seria equivalente ao `JOIN`

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

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


Usar ON ou WHERE nas consultas com múltiplas relações?

- Em junções internas, pode-se usar a condição ON ou WHERE
  - Na cláusula FROM, a vírgula (,) seria equivalente ao `JOIN`

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

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

- Contudo, a consulta SQL pode ser mais legível se usar ON como condição de junção e as demais condições na cláusula 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?

Usar ON ou WHERE nas consultas com múltiplas relações?

- Em junções internas, pode-se usar a condição ON ou WHERE
  - Na cláusula FROM, a vírgula (,) seria equivalente ao `JOIN`

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

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

- Contudo, a consulta SQL pode ser mais legível se usar ON como condição de junção e as demais condições na cláusula WHERE

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

Em junções externas, condições ON se comportam de maneira diferente das condições com WHERE

Junções externas (*outer join*)

- Com a junção natural não serão exibidas todas as tuplas de `Departamento` que não tiverem um `Funcionário` associado

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


Junções externas (*outer join*)

- Com a junção natural não serão exibidas todas as tuplas de `Departamento` que não tiverem um `Funcionário` associado

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

Junções externas

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

Junções externas (*outer join*)

- Com a junção natural não serão exibidas todas as tuplas de `Departamento` que não tiverem um `Funcionário` associado

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

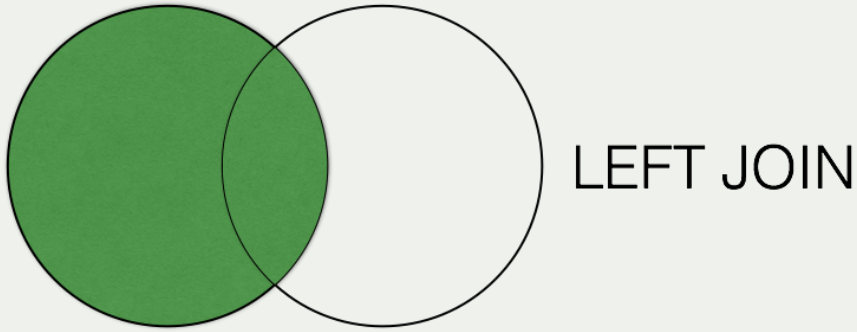
Junções externas

Preserva as tuplas que seriam perdidas em um junção, criando na relação resultante tuplas que contêm 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

Condições de junção: LEFT JOIN

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



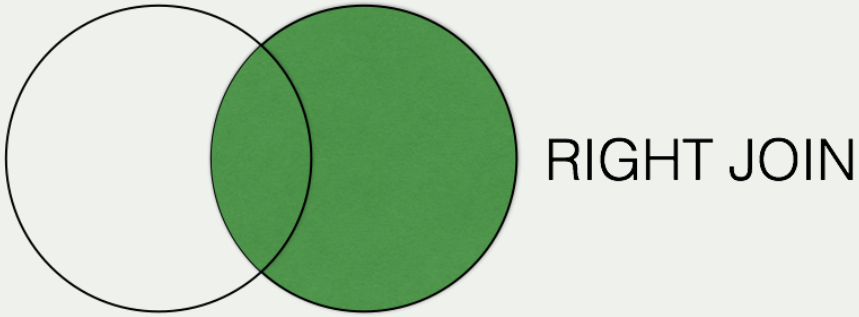

```sql
SELECT colunas FROM tabela1
  LEFT JOIN tabela2 ON tabela1.colunaA = tabela2.colunaB;

-- Exemplo
SELECT * FROM Departamento d
  LEFT JOIN Funcionario f ON d.idDepartamento = f.idDepartamento;

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

Condições de junção: RIGHT JOIN

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



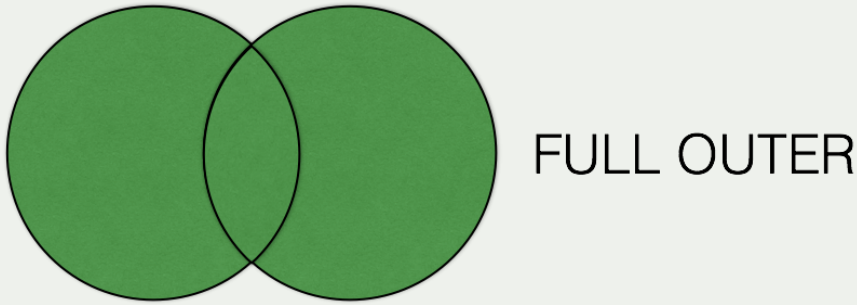

```sql
SELECT colunas FROM tabela1
  RIGHT JOIN tabela2 ON tabela1.colunaA = tabela2.colunaB;

-- Exemplo
SELECT * FROM Departamento d
  RIGHT JOIN Funcionario f ON d.idDepartamento = f.idDepartamento;

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

Condições de junção: FULL OUTER JOIN

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




```sql
SELECT colunas FROM tabela1
  FULL OUTER JOIN tabela2 ON tabela1.colunaA = tabela2.colunaB;

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

- MySQL não implementa FULL JOIN

Exercícios

Funcionário e Departamento

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

- Esquema MySQL disponível em: http://docente.ifsc.edu.br/mello/bcd/labs/lab03-funcionario-departamento.sql

Funcionário e Departamento

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

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

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

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
  - A tupla resultante é então usada pela instrução onde essa está aninhada
- Geralmente é usada em operações com conjuntos
  - 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
  - A tupla resultante é então usada pela instrução onde essa está aninhada
- Geralmente é usada em operações com conjuntos
  - 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

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

- Na tabela Aluno, atualizar todas tuplas com valor = 2 no atributo "codigoCurso" para o código que está associado ao curso chamado "Engenharia de Telecomunicações" na tabela Curso

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


Subconsultas

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

- Fazendo uso de INNER JOIN

- Fazendo uso de subconsultas (operador IN)

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

- Fazendo uso de INNER JOIN

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

- Fazendo uso de subconsultas (operador IN)

Subconsultas

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

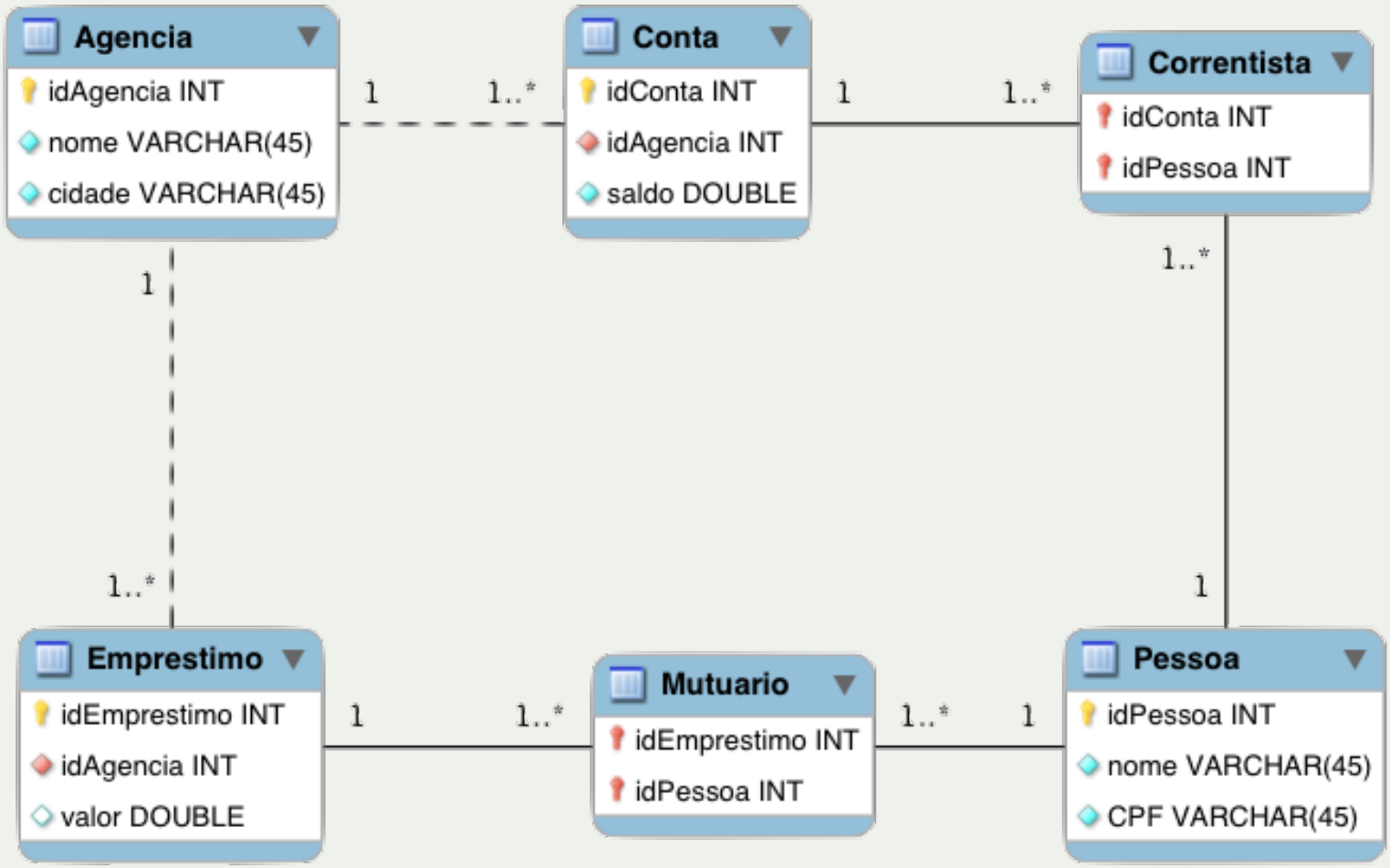
- Fazendo uso de INNER JOIN

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

- Fazendo uso de subconsultas (operador IN)

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


Esquema usado nos próximos exemplos



- Esquema MySQL disponível em:
  http://docente.ifsc.edu.br/mello/bcd/labs/lab04-banco.sql

Subconsultas

Exemplos com operador IN

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

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

- 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

- 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)
```

Subconsultas

Exercícios 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)
```

- 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')
```


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

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

- 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)
```

Subconsultas: Exemplo com cláusula `FROM`

- Soma dos valores médios de empréstimos realizados por cada agência

Subconsultas: Exemplo com 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 |
+------------+
```

Subconsultas: Exemplo com 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 |
+------------+
```

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

Subconsultas: Exemplo com 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 |
+------------+
```

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

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

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

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

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

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


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

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

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

Operações com conjuntos: cláusula SOME – pelo menos um

1
5
8

- (5 < SOME) $\equiv$ 5 < que alguma tupla na relação? TRUE!

1
5

- (5 < SOME) = FALSE!

1
5

- (5 = SOME) = TRUE!

1
5

- (5 <> SOME) = TRUE!, pois $1 \neq 5$

Operações com conjuntos: cláusula SOME – pelo menos um

1
5
8

- (5 < SOME) $\equiv$ 5 < que alguma tupla na relação? TRUE!

1
5

- (5 < SOME) = FALSE!

1
5

- (5 = SOME) = TRUE!

1
5

- (5 <> SOME) = TRUE!, pois $1 \neq 5$

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

Operações com conjuntos: cláusula `ALL` – que todos

1
5
8

- (5 < ALL) $\equiv$ 5 < que todas tupla na relação? FALSE!

1
6

- (5 < ALL) = FALSE!

1
5

- (5 = ALL) = FALSE!

1
6

- (5 <> ALL) = TRUE!

Operações com conjuntos: cláusula `ALL` – que todos

1
5
8

- (5 < ALL) $\equiv$ 5 < que todas tupla na relação? FALSE!

1
6

- (5 < ALL) = FALSE!

1
5

- (5 = ALL) = FALSE!

1
6

- (5 <> ALL) = TRUE!

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

Subconsultas: Teste de relação vazia – `EXISTS`

- Retorna TRUE se a relação resultante da subconsulta for $\neq \emptyset$, FALSE caso contrário

Subconsultas: Teste de relação vazia – EXISTS

- Retorna TRUE se a relação resultante da subconsulta for $\neq \emptyset$, FALSE caso contrário
- 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)
```

Subconsultas

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 vs GROUP BY

Subconsulta escalar retorna uma única tupla com um único atributo

- Liste o total em R$ das vendas de cada produto

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

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

- Mesma consulta, porém fazendo uso de `GROUP BY`

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


Exemplos com DELETE e múltiplas relações

Apagando linhas de uma ou mais tabelas

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

- Apagando todas as linhas da tabela Curso cujo código do curso seja igual a 123

```
DELETE FROM Curso WHERE curso = 123;
```

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

```
DELETE Curso, Disciplina, Aluno FROM Curso INNER JOIN Disciplina
  INNER JOIN Aluno
    WHERE Curso.idCurso = 123 AND Curso.idCurso = Disciplina.curso
      AND Curso.idCurso = Aluno.curso;
```


Exemplos com INNER JOIN e subconsulta

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

- Exemplo com INNER JOIN

```
DELETE Aluno FROM Aluno INNER JOIN Curso
  ON Aluno.curso = Curso.idCurso
    WHERE Curso.Nome = 'Curso XYZ';
```

- Exemplo com subconsulta

```
DELETE Aluno FROM Aluno
  WHERE Aluno.curso 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