

Subconsulta SQL, Joins e Operações com Conjuntos

André L. R. Madureira <andre.madureira@ifba.edu.br>
Doutorando em Ciência da Computação (UFBA)
Mestre em Ciência da Computação (UFBA)
Engenheiro da Computação (UFBA)

Conteúdo do Slide

- Subconsultas SQL e Consultas aninhadas
- Testes de existência de registros (**IN** e **EXISTS**)
- Comparação de registros de uma mesma tabela (**SOME**, **ALL**)
- Operações de conjuntos (**UNION**, **INTERSECT**, **EXCEPT**)
- Operações de junção (inclusiva e exclusiva)
 - **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**
 - **FULL OUTER JOIN**

Subconsultas ou Consultas Aninhadas

- São consultas SQL que são usadas por outras consultas
- **Sintaxe:**
 - **SELECT** <atributos> **FROM** (<consulta_SQL>)
 - **SELECT** <atributos> **FROM** <tabela> **WHERE** <atributo> [**IN, SOME, ANY, EXISTS**] (<consulta_SQL>)
 - (<consulta_SQL>) [**UNION, INTERSECT, EXCEPT**] (<consulta_SQL>)

Exemplo de Subconsultas SQL

- **SELECT** conta, nova_quantia
FROM (
 SELECT conta, quantia * (1+tx_juros) **AS** nova_quantia
 FROM Devedor
) **AS** max_quantia_tabela
 - Calcule o total do empréstimo que o tomador deve ao banco

O parêntese separando as consultas SQL é essencial para o DBMS executar a operação.
(“DBMS precisa saber onde começa o **SELECT** conta, nova_quantia ...
e o **SELECT** conta, quantia*(1+tx_juros) ...”)

Exemplo de Subconsultas SQL

- **SELECT** numero, agencia, quantia
FROM (
 SELECT MAX(quantia) **AS** max_quantia
 FROM Devedor
) **AS** max_qtd_tabela , Devedor
WHERE quantia = max_quantia
 - Encontre a conta com maior quantia de emprestimo

Comando IN

- Verifica a participação de uma tupla em uma relação (tabela)
- **Sintaxe:**
 - **SELECT** <atributos> **FROM** <tabela> **WHERE** <atributo> [**NOT**] **IN** (<conjunto_enumerado_ou_consulta_SQL>)
- **Ex:**
 - **SELECT** nome **FROM** Aluno **WHERE** nome **IN** ('João', 'Claudia')
 - **SELECT** faltas **FROM** Aluno **WHERE** faltas **IN** (14, 15, 16, 17)

Conjunto enumerado



Exemplo de Comando IN

- **SELECT** nome **FROM** Aluno **WHERE** cpf **IN** (
 SELECT cpf **FROM** Professor
)
 ○ Encontre os Professores que também são Alunos
- Uma consulta SQL pode ser escrita de infinitas formas no SQL
 - **SELECT** Aluno.nome
 FROM Aluno, Professor
 WHERE Aluno.cpf = Professor.cpf

Consulta aninhada

Como escolher a melhor consulta SQL?

- **SELECT** nome **FROM** Aluno **WHERE** cpf **IN** (**SELECT** cpf **FROM** Professor)
 - OU
- **SELECT** Aluno.nome **FROM** Aluno, Professor **WHERE** Aluno.cpf = Professor.cpf

SUGESTÃO:

Crie a consulta que seja mais simples para você entender (e usar). Teste o desempenho e verifique se atende às necessidades do seu sistema. Veremos como avaliar o desempenho de consultas SQL em breve.

Exemplo de Comando IN

- **SELECT** nome_cliente **FROM** Devedor **WHERE** (num_conta, agencia) **IN** ((**SELECT** num_conta, agencia **FROM** Depositante) **AS** t_Devedor_conta_agencia)
 - Encontre os clientes que possuem conta no banco e possuem algum empréstimo
 - Localize esses clientes usando a tupla de atributos (num_conta, agencia) pois esses atributos são chaves candidatas das tabelas

Comparação entre Registros de uma Mesma Tabela

- O SQL permite que comparemos registros (instâncias) de uma mesma tabela para realizarmos consultas mais sofisticadas
 - **SELECT** T.nome_agencia
FROM agencia **AS** T, agencia **AS** S
WHERE T.qtd_func > S.qtd_func
 - Encontre os nomes de todas as agências que possuem uma quantidade de funcionários maior do que ALGUMA das agências do banco

Comando **SOME**

- Compara o registro com outros, retornando TRUE se existir ALGUM outro registro maior, menor, igual ou diferente.
 - **Sintaxe:**
 - **SELECT** <lista_atributos> **FROM** <tabela> **WHERE** <atributo> [**>SOME**, **<SOME**, **=SOME**, **<>SOME**, **>=SOME**, **<= SOME**] (<consulta_sql>)

Exemplo do comando **SOME**

- **SELECT** nome_agencia
FROM agencia
WHERE qtd_func >**SOME** (
 SELECT qtd_func
 FROM agencia
)
 - Encontre os nomes de todas as agências que possuem uma quantidade de funcionários maior do que **ALGUMA** das agências

Comando **ALL**

- Compara o registro com outros, retornando TRUE se esse registro for maior, menor, igual ou diferente de TODOS os registros.
 - **Sintaxe:**
 - **SELECT** <atributo> **FROM** <tabela> **WHERE** <atributo> [**<ALL,<=ALL,=ALL,>ALL,>=ALL, <>ALL**] (<consulta_SQL>)

Exemplo do comando **ALL**

- **SELECT** nome_agencia
FROM agencia
WHERE qtd_func <**ALL** (
 SELECT qtd_func
 FROM agencia
 WHERE cidade_agencia = 'Salvador'
)
 - Encontre os nomes de todas as agências que possuem uma quantidade de funcionários **menor do que TODAS** as agências de Salvador

Comando EXISTS

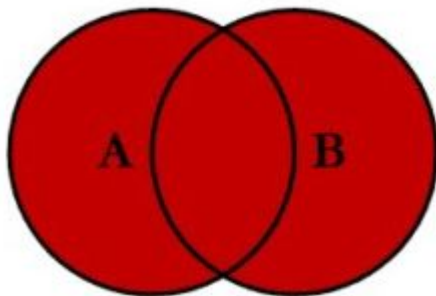
- Retorna TRUE se existe pelo menos um registro em uma subconsulta SQL.
 - **Sintaxe:**
 - **SELECT** <atributo> **FROM** <tabela> **WHERE [NOT] EXISTS** (<consulta_SQL>)

Exemplo do comando **EXISTS**

- **SELECT** Pnome, Unome
FROM Funcionario
WHERE NOT EXISTS (
 SELECT *
 FROM Dependente
 WHERE Dependente.cpf_tutor = Funcionario.cpf
)
 - Recuperar os nomes de funcionários que não possuem dependentes

Union

- Combina os registros de duas consultas com mesmos atributos
- **Sintaxe:**
 - **(SELECT <atributos> FROM <tabela_A>) UNION (SELECT <atributos> FROM <tabela_A>);**



Exemplo UNION

- **(SELECT nome_cliente FROM depositante) UNION (SELECT nome_cliente FROM tomador)**
 - Encontre todos os clientes que possuem uma conta no banco, um empréstimo ou as duas coisas

A operação **UNION** descarta registros duplicados automaticamente.

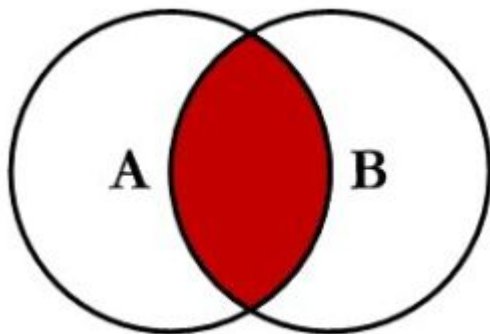
Ex: dois clientes com nome “Ana”, um na tabela *depositante* e outro na tabela *tomador*

Para mostrar os registros duplicados usamos a operação **UNION ALL**.

Ex: **(SELECT nome_cliente FROM depositante) UNION ALL (SELECT nome_cliente FROM tomador)**

Intersect

- Registros que pertencem às duas consultas SQL
- **Sintaxe:**
 - **(SELECT <atributos> FROM <tabela_A>) INTERSECT (SELECT <atributos> FROM <tabela_A>);**



Exemplo INTERSECT

- **Ex:** (**SELECT** nome_cliente **FROM** depositante) **INTERSECT** (**SELECT** nome_cliente **FROM** tomador)
 - Encontre todos os clientes que possuem uma conta no banco e um empréstimo também

A operação **INTERSECT** descarta registros duplicados automaticamente.

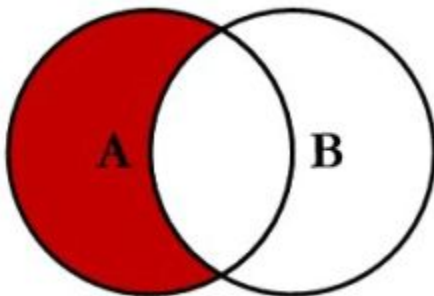
Ex: dois clientes com nome “Ana”, um na tabela *depositante* e outro na tabela *tomador*

Para mostrar os registros duplicados usamos a operação **INTERSECT ALL**.

Ex: (**SELECT** nome_cliente **FROM** depositante) **INTERSECT ALL** (**SELECT** nome_cliente **FROM** tomador)

Except

- Retorna os registros que pertencem a primeira consulta, mas não pertencem à segunda
- **Sintaxe:**
 - **(SELECT <atributos> FROM <tabela_A>) EXCEPT (SELECT <atributos> FROM <tabela_A>);**



Exemplo **EXCEPT**

- **Ex:** (**SELECT** nome_cliente **FROM** depositante) **EXCEPT** (**SELECT** nome_cliente **FROM** tomador)
 - Encontre todos os clientes que possuem uma conta no banco, mas não possuem empréstimos

A operação **EXCEPT** descarta registros duplicados automaticamente.

Ex: se existirem dois clientes com nome “Ana”, apenas um aparecerá no resultado

Para mostrar os registros duplicados usamos a operação **EXCEPT ALL**.

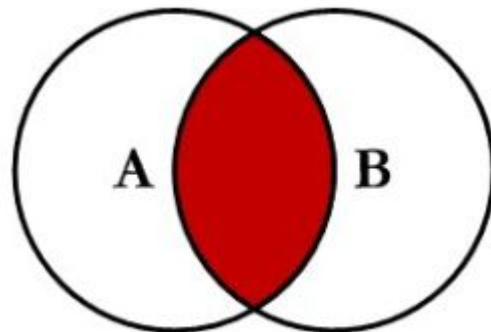
Ex: (**SELECT** nome_cliente **FROM** depositante) **EXCEPT ALL** (**SELECT** nome_cliente **FROM** tomador)

Inner Join

- Retorna os registros que são comuns à duas tabelas

- **Sintaxe:**

- **SELECT** <atributos>
FROM <tabela_A>
[INNER] JOIN <tabela_B>
 ON <tabela_A>.<atributo> = <tabela_B>.<atributo>
[WHERE <condição>
[GROUP BY, HAVING, ORDER BY]



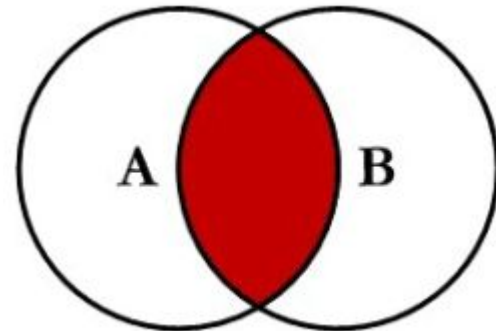
Funcionarios Gerentes

#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios **INNER JOIN** Gerentes **ON** Funcionarios.nome = Gerentes.nome ;

#	Nome	Nome
1	Fernando	Fernando
2	Luiz	Luiz



Funcionarios Gerentes

Outra forma de fazer um **inner join** (junção) é através de uma cláusula **WHERE**

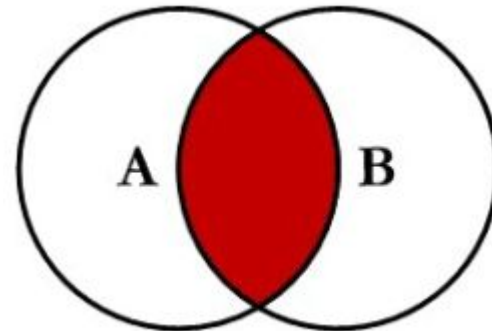
#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

Condição de junção

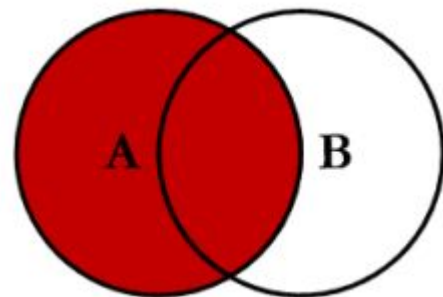
- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios, Gerentes **WHERE** Funcionarios.nome = Gerentes.nome;

#	Nome	Nome
1	Fernando	Fernando
2	Luiz	Luiz



Left Join

- Tem como resultado todos os registros que estão na tabela A e os registros da tabela B que são comuns à tabela A
- **Sintaxe:**
 - **SELECT** <atributos>
FROM <tabela_A>
LEFT JOIN <tabela_B>
ON <tabela_A>.<atributo> = <tabela_B>.<atributo>
[WHERE, GROUP BY, HAVING, ORDER BY]



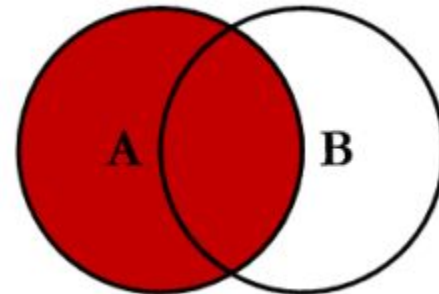
Funcionarios Gerentes

#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

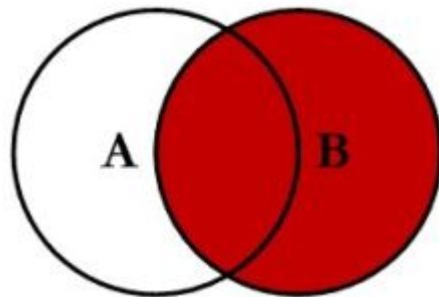
- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios **LEFT JOIN** Gerentes **ON** Funcionarios.nome = Gerentes.nome ;

#	Nome	Nome
1	Fernando	Fernando
2	Joao	NULL
3	Luiz	Luiz
4	Maria	NULL



Right Join

- Tem como resultado todos os registros que estão na tabela B e os registros da tabela A que são comuns à tabela B
- **Sintaxe:**
 - **SELECT** <atributos>
FROM <tabela_A>
RIGHT JOIN <tabela_B>
ON <tabela_A>.<atributo> = <tabela_B>.<atributo>
[WHERE, GROUP BY, HAVING, ORDER BY]



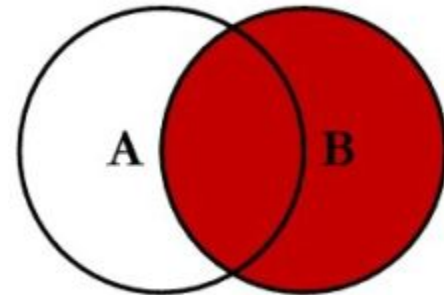
Funcionarios Gerentes

#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

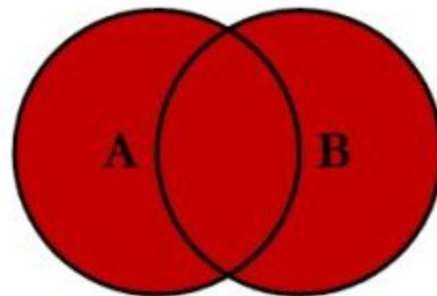
- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios **RIGHT JOIN** Gerentes **ON** Funcionarios.nome = Gerentes.nome ;

#	Nome	Nome
1	NULL	Carla
2	Fernando	Fernando
3	NULL	Francisco
4	Luiz	Luiz



Full Outer Join (ou FULL JOIN)

- Tem como resultado todos os registros que estão na tabela A e todos os registros da tabela B
- **Sintaxe:**
 - **SELECT** <atributos>
FROM <tabela_A>
FULL OUTER JOIN <tabela_B>
ON <tabela_A>.<atributo> = <tabela_B>.<atributo>
[WHERE, GROUP BY, HAVING, ORDER BY]

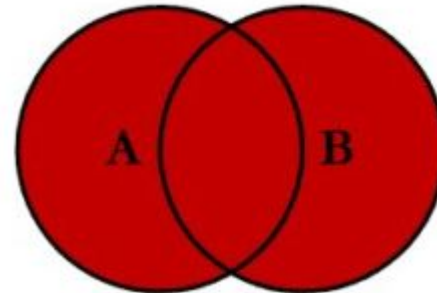


#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios **FULL OUTER JOIN** Gerentes **ON** Funcionarios.nome = Gerentes.nome ;

#	Nome	Nome
1	Fernando	Fernando
2	Joao	NULL
3	Luiz	Luiz
4	Maria	NULL
5	NULL	Carla
6	NULL	Francisco

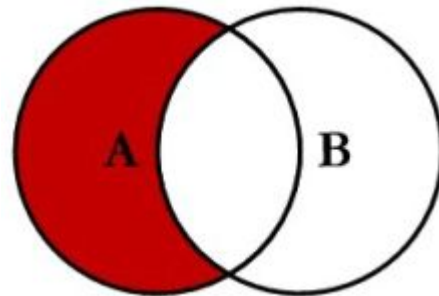


Exercício - Joins SQL

- Crie as tabelas Funcionarios e Gerentes no banco de dados SQL
- Insira registros de exemplo nessas tabelas
 - **DICA:** insira registros de pessoas com nomes diferentes e iguais também
- Execute operações JOIN, LEFT JOIN, RIGHT JOIN e FULL OUTER JOIN nessas tabelas. Anote os resultados obtidos.
 - Os resultados foram iguais aos que você esperava?

Left Excluding Join

- Tem como resultado todos os registros que estão na tabela A e não estão na tabela B
- **Sintaxe:**
 - **SELECT** <atributos>
FROM <tabela_A>
LEFT JOIN <tabela_B>
ON <tabela_A>.<atributo> = <tabela_B>.<atributo>
WHERE <tabela_B>.<atributo> **IS NULL**



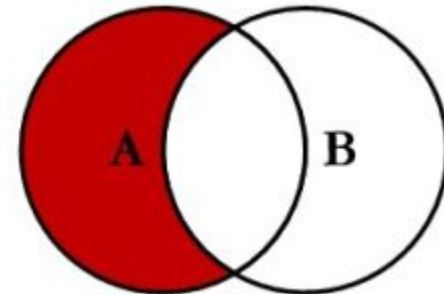
Funcionarios Gerentes

#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

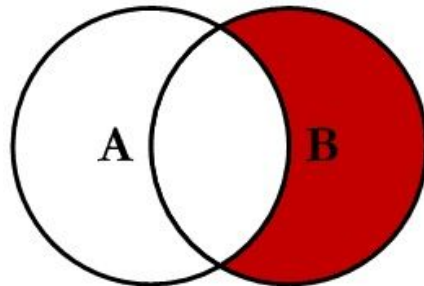
- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios **LEFT JOIN** Gerentes **ON** Funcionarios.nome = Gerentes.nome **WHERE** Gerentes.nome **IS NULL** ;

#	Nome	Nome
1	Joao	NULL
2	Maria	NULL



Right Excluding Join

- Tem como resultado todos os registros que estão na tabela B e não estão na tabela A
- **Sintaxe:**
 - **SELECT** <atributos>
FROM <tabela_A>
RIGHT JOIN <tabela_B>
ON <tabela_A>.<atributo> = <tabela_B>.<atributo>
WHERE <tabela_A>.<atributo> **IS NULL** ;



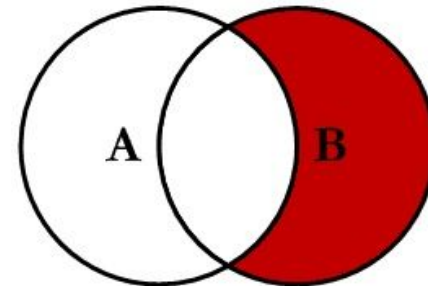
Funcionarios Gerentes

#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios **RIGHT JOIN** Gerentes **ON** Funcionarios.nome = Gerentes.nome **WHERE** Funcionarios.nome **IS NULL** ;

#	Nome	Nome
1	NULL	Carla
2	NULL	Francisco



Outer Excluding Join

- Retorna todos os registros que estão na tabela B e não estão na tabela A, e todos os registros que estão na tabela A e não estão na tabela B

- **Sintaxe:**

- **SELECT** <atributos>

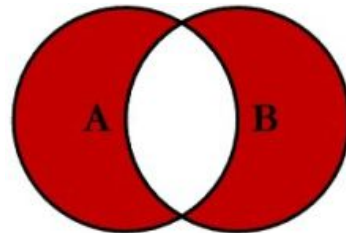
- FROM** <tabela_A>

- FULL OUTER JOIN** <tabela_B>

- ON** <tabela_A>.<atributo> = <tabela_B>.<atributo>

- WHERE** <tabela_A>.<atributo> **IS NULL**

- OR** <tabela_B>.<atributo> **IS NULL**



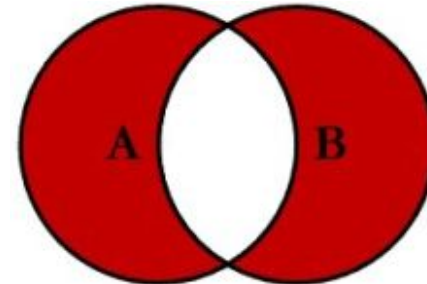
Funcionarios Gerentes

#	Nome
1	Fernando
2	Joao
3	Luiz
4	Maria
*	NULL

#	Nome
1	Carla
2	Fernando
3	Francisco
4	Luiz
*	NULL

- **SELECT** Funcionarios.nome, Gerentes.nome **FROM** Funcionarios **FULL OUTER JOIN** Gerentes **ON** Funcionarios.nome = Gerentes.nome **WHERE** Funcionarios.nome **IS NULL OR** Gerentes.nome **IS NULL**;

#	Nome	Nome
1	Joao	NULL
2	Maria	NULL
3	NULL	Carla
4	NULL	Francisco



Exercício - Excluding Joins SQL

- Crie as tabelas Funcionários e Gerentes e execute as operações LEFT EXCLUDING JOIN, RIGHT EXCLUDING JOIN e OUTER EXCLUDING JOIN
 - Anote os resultados obtidos
 - Qual a diferença dos resultados dessas operações para os resultados das operações outras operações JOIN comuns?

Referencial Bibliográfico

- KORTH, H.; SILBERSCHATZ, A.; SUDARSHAN, S. **Sistemas de bancos de dados**. 5. ed. Rio de Janeiro: Ed. Campus, 2006.
- DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Ed. Campus, 2004. Tradução da 8ª edição americana.