Revisão + Visões + Sub-Consultas + JOINS

Professor: Ricardo Luis dos Santos

IFSUL – Campus Sapucaia do Sul

© 2015 Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense

Campus Sapucaia do Sul

Agenda

- História
- Vantagens e Desvantagens
- Conceitos
- Tipos de chaves
- SQL (Structured Query Language)
- DML (Data Manipulation Language)
- DQL – Linguagem de Consulta de Dados
- Visões
- Sub-consultas

História

- Anos 1950 e 1960:
  - Armazenamento baseado em fitas magnéticas
  - Acesso sequencial, Cartões perfurados
- Anos 1970:
  - Discos rígidos permitem o acesso direto aos dados
  - Modelo relacional (Ted Codd)
  - Processamento de transações de alto desempenho
- Anos 1980:
  - Protótipos relacionais viram sistemas comerciais
  - SQL se torna o padrão da indústria

História

- Anos 1990:
  - Suporte à decisão e data-mining
  - Armazenamento em escala muito alta (terabytes)
  - Comércio eletrônico
- Anos 2000:
  - XML and XQuery
  - Administração automática
  - SGBDs altamente paralelizados
  - Sistemas de armazenamento distribuídos na Web (DropBox)

Vantagens e Desvantagens

- Algumas das principais vantagens
  - Disponibilidade
    - Compartilhamento dos dados entre aplicações
  - Controle da redundância dos dados
    - A informação pode aparecer apenas uma vez
  - Precisão e Integridade dos dados
    - Alterações nos dados só podem ser realizadas em um lugar (BD)
  - Segurança das Informações
  - Ambiente amigável

Vantagens e Desvantagens

- Algumas das principais vantagens
  - Interoperabilidade
    - Entre aplicações e plataformas (com relação aos dados salvos)
  - Desempenho
  - Escalabilidade
  - Tolerância a Falhas
  - Padronização

Vantagens e Desvantagens

- Algumas das principais desvantagens
  - Custos
    - Iniciais de software e hardware altos
    - Operacionais podem ser elevados (e.g., mão de obra qualificada)
  - Complexidade requerida mesmo para aplicações simples

Conceitos

- O que é um banco de dados?
  - São coleções organizadas de dados que se relacionam de forma a criar algum sentido (Informação) e dar mais eficiência durante uma pesquisa ou estudo
  - São de vital importância para empresas e há duas décadas se tornaram a principal peça dos sistemas de informação
- O que é um SGBD (Sistema Gerenciador de Banco de Dados)?
  - É um sistema de software de finalidade genérica que facilita o processo de definição, construção e manipulação de banco de dados

Conceitos

- Tabelas
  - São os objetos que contém os tipos de dados e os dados reais
- Colunas ou Campos
  - São as partes das tabelas que armazenam os dados. Devem receber um tipo de dados e ter um nome único
- Tipos de dados
  - Há vários tipos de dados para serem utilizados como: caractere, número, data. Um único tipo de dados é atribuído a uma coluna dentro de uma tabela

Conceitos

- Stored Procedures (procedimentos armazenados)
  - São como macros em que o código Transact-SQL pode ser escrito e armazenado sob um nome.
- Triggers (gatilhos)
  - São como stored procedures que são automaticamente ativados quando os dados são inseridos, alterados ou apagados
- Regras (rules)
  - São atribuídas a colunas de modo que os dados que estão sendo inseridos devem se adaptar aos padrões definidos
    - Por exemplo, pode-se utilizar regras para permitir que um campo que irá armazenar a UF contenha somente Estados válidos

Conceitos

- Chaves Primárias (PK)
  - Promovem a unicidade das linhas, proporcionando uma maneira de identificar de forma única cada item que você queira armazenar
- Chaves Estrangeiras (FK)
  - Novamente, não são objetos em si, as chaves estrangeiras são colunas que fazem referências as chaves primárias de outras tabelas
- Índices
  - Podem ajudar os dados de modo que as consultas executem mais rápido

Conceitos

- Padrões (Defaults)
  - Podem ser configurados em campos de modo que, se nenhum dado for inserido durante uma operação de Insert, **os valores padrão serão utilizados**
- Views (visualizações)
  - Consistem basicamente em consultas armazenadas nos bancos de dados que podem fazer referência a uma ou muitas tabelas.
    - Podem ser criadas e salvas para utilização no futuro
    - Normalmente excluem certas colunas de uma tabela e vinculam duas ou mais tabelas entre si
    - Podem ser utilizadas também como mecanismo de segurança

Conceitos

- Transação
  - Conjunto de procedimentos que é executado num banco de dados, que para o usuário é visto como uma única ação
  - A integridade de uma transação depende de 4 propriedades, conhecidas como ACID
    - Atomicidade
    - Consistência
    - Isolamento
    - Durabilidade

Conceitos

- Atomicidade
  - Todas as ações que compõem a unidade de trabalho da transação devem ser concluídas com sucesso. Se durante a transação qualquer ação falhar, a transação inteira deve ser desfeita (rollback)
- Consistência
  - Todas as regras e restrições definidas no banco de dados devem ser obedecidas
    - Relacionamentos por chaves estrangeiras, checagem de valores para campos restritos ou únicos devem ser obedecidos
- Isolamento
  - Cada transação funciona completamente isolada de outras transações
    - Outras transações não podem visualizar os resultados parciais das operações de uma transação em andamento
- Durabilidade
  - Significa que os resultados de uma transação são permanentes e podem ser desfeitos somente por uma transação subsequente.

Tipos de Chaves

- Chaves
  - Primária
  - Estrangeira
  - Candidata

Tipos de Chaves

- Chave primária
  - Uma coluna ou combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela. Identifica univocamente uma linha.

Tabela Empregado

CodEmpregado	Nome	CodDepartamento	CategoriaFuncional
E5	Luiz	D1	C5
E3	Marcos	D2	C5
E2	Paula	D1	C2

Tipos de Chaves

- Chave Estrangeira
  - Uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela.
  - Mecanismo que permite a implementação de relacionamentos em um banco de dados relacional

Tipos de Chaves

Tabela Departamento

CodDepartamento	Descrição
D1	Engenharia
D2	Vendas

CodDepartamento em Empregado é Chave Estrangeira em relação a Departamento

Tabela Empregado

CodEmpregado	Nome	CodDepartamento	CategoriaFuncional
E5	Luiz	D1	C5
E3	Marcos	D2	C5
E2	Paula	D1	C2

Tipos de Chaves

- Chave Estrangeira na mesma tabela

Tabela Empregado

CodEmpregado	Nome	CodDepartamento	CategoriaFuncional	CodGerente
E5	Luiz	D1	C5	-
E3	Marcos	D2	C5	-
E2	Paula	D1	C2	E5
E1	José	D2	C2	E3

Tipos de Chaves

- Chave Candidata
  - Tabela relacional pode possuir alternativas de identificador único (colunas ou concatenações de colunas).

CodEmpregado	Nome	CodDepartamento	CategoriaFuncional	CodGerento	CPF
E5	Luiz	D1	C5	-	234.343.12
E3	Marcos	D2	C5	-	187.373.90
E2	Paula	D1	C2	E5	123.462.65
E1	José	D2	C2	E3	190.987.89

SQL (Structured Query Language)

- Os comandos SQL podem dividir-se em cinco grandes categorias:
  - DML - Data Manipulation Language
    - Linguagem de Manipulação de Dados
    - Utilizada para a recuperação, inclusão, remoção e modificação de informações em bancos de dados
  - DDL - Data Definition Language
    - Linguagem de Definição de Dados
    - Trabalha com objetos (ex: tabelas)
  - DCL - Data Control Language
    - Linguagem de Controle da Dados
    - Trabalha com usuários

SQL (Structured Query Language)

- Os comandos SQL podem dividir-se em cinco grandes categorias:
  - DTL - *Data Transaction Language* (Linguagem de Transação de Dados)
    - Utilizado pelos desenvolvedores em transações. Os principais comandos são COMMIT e ROLLBACK
  - DQL - *Data Query Language* (Linguagem de Consulta de Dados)
    - O mais importante dentre estes, pois consultas são realizadas a todo instante. O comando que é utilizado pelo DQL é o SELECT. Algumas literaturas consideram o SELECT também em DML

SQL (Structured Query Language)

Comando	Descrição	Grupo de Comandos
SELECT	Utilizado para extrair dados da base de dados	DML e DQL
INSERT	Introduzir novas linhas	DML
UPDATE	Alterar linhas já existentes	DML
DELETE	Apagar linhas já existentes	DML
CREATE	Criar objetos da base de dados (tabelas, índices, visões)	DDL
ALTER	Alterar objetos da base de dados (tabelas, índices, visões)	DDL
DROP	Apagar objetos da base de dados (tabelas, índices, visões)	DDL

SQL (Structured Query Language)

Comando	Descrição	Grupo de Comandos
GRANT	Conceder acesso à base de dados e aos seus objectos	DCL
REVOKE	Retirar acesso à base de dados e aos seus objectos	DCL
DENY	Utilizado para impedir explicitamente que um usuário receba uma permissão especial	DCL
COMMIT	Transforma todas as modificações de dados executadas desde o início da transação em parte permanente do banco de dados, liberando os recursos ocupados	DTL
ROLLBACK	Reverte uma transação explícita ou implícita ao começo da transação ou a um ponto de salvamento dentro da transação	DTL

DML (Data Manipulation Language)

- INSERT
  - Insere registros em uma tabela
  - INSERT INTO nome_tabela [(nome_coluna,...)] VALUES (valor1,...)
    - INSERT INTO Cliente (CodCliente, Nome, Cpf, Endereco, Cidade) VALUES (NULL,'Fábio','12344','Rua A, 1112',Sapucaia do Sul');
    - INSERT INTO Cliente VALUES (NULL,'Fábio','12344','Rua A, 1112',‘Sapucaia do Sul');

DML (Data Manipulation Language)

- UPDATE
  - Atualiza registros de uma tabela
  - UPDATE nome_tabela SET nome_coluna1=expr1 [,nome_coluna2=expr2 ...] [WHERE definição_where]
    - UPDATE Cliente SET Nome = 'Luís',Cpf ='12345',Endereco = 'Rua A, 112', Cidade = 'Canoas' WHERE CodCliente = 1;
    - UPDATE Cliente SET Cidade = 'Porto Alegre' WHERE CodCliente = 1;

DML (Data Manipulation Language)

- DELETE
  - Deleta registros de uma tabela
  - DELETE FROM table_name
    [WHERE definição_where]
    - DELETE FROM Cliente WHERE CodCliente = 1;
    - DELETE FROM Cliente WHERE Cpf = ‘12345’;

DML (Data Manipulation Language)

- Sintaxe SELECT

```
SELECT [DISTINCT] {lista_de_colunas | *}
FROM nome_tabela
[WHERE definição_where]
[GROUP BY {inteiro_sem_sinal | nome_col | formula} [ASC | DESC]
[HAVING where_definition]
[ORDER BY {inteiro_sem_sinal | nome_coluna | formula} [ASC | DESC]
[LIMIT [offset,] row_count | row_count OFFSET offset]
```

DML (Data Manipulation Language)

- Exemplo 1: encontrar todos os nomes dos clientes na tabela cliente:

```
SELECT nome
FROM cliente;
```

- O resultado é uma relação consistindo em um único atributo

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

DML (Data Manipulation Language)

- Exemplo 2: listar os modelos da tabela automóvel:

```
SELECT modelo
FROM automovel;
```

- Como resultado obtemos uma relação contendo um atributo. **Obtemos todos os modelos, apresentando duplicatas**, para eliminação das duplicatas utiliza-se a cláusula DISTINCT

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

DQL – Linguagem de Consulta de Dados

SELECT **DISTINCT** modelo

FROM automovel;

- O resultado é uma relação dos modelos da tabela automóvel sem duplicatas

DQL – Linguagem de Consulta de Dados

- Utilização do * na cláusula SELECT
  - O * é utilizado para indicar “todos os atributos” da tabela.
- Exemplo: listar a relação de todos os atributos da tabela cliente

```
SELECT *
FROM cliente;
```


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE

DQL – Linguagem de Consulta de Dados

- A cláusula SELECT também pode conter expressões aritméticas envolvendo +, - , * e / operando em constantes ou atributos.
- Exemplo: listar os nome e os salários da tabela corretor com um aumento de 10%:

```
SELECT nome, salario * 1.10
FROM corretor;
```

DQL – Linguagem de Consulta de Dados

- Cláusula WHERE
  - Exemplo: selecione o nome e os salários dos corretores com salários superiores a R$ 1.000,00:

```
SELECT nome, salario
FROM corretor
WHERE salario > 1000.00;
```

DQL – Linguagem de Consulta de Dados

- Exemplo: selecione marca, modelo e ano dos automóveis da marca VW com ano superior a 2005;

```
SELECT modelo, marca, ano
FROM automovel
WHERE marca = 'VW' AND ano > 2005;
```

Ou:

```
SELECT modelo, marca, ano
FROM automovel
WHERE marca LIKE 'VW' AND ano > 2005;
```

DQL – Linguagem de Consulta de Dados

- SQL utiliza os seguintes conectivos lógicos:
  - AND, OR e NOT na cláusula WHERE;
  - os operandos dos conectivos lógicos envolvendo comparação são <, <=, >, >=, = e <> (sendo que o diferente, no MySQL, também pode ser expresso como !=)

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

DQL – Linguagem de Consulta de Dados

- O operador de comparação **BETWEEN** especifica que um valor seja maior ou igual a um valor e menor ou igual a outro valor.
- Exemplo: selecione modelo, marca e ano dos automóveis com ano entre 2000 e 20012.

```
SELECT modelo, marca, ano
FROM automovel
WHERE marca LIKE 'VW' AND
    ano BETWEEN 2000 AND 2012;
```

DQL – Linguagem de Consulta de Dados

- Operações de String
  - Correspondência de padrões usando o operador LIKE são descritos usando dois caracteres especiais:
    - Porcentagem (%): corresponde a qualquer substring;
    - Sublinhado (_): corresponde a qualquer caractere.
  - Os padrões são *Case Sensitive*.

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE

DQL – Linguagem de Consulta de Dados

- Exemplo: localiza os clientes cujo nome inicia com a letra “J”:

```
SELECT *
FROM cliente
WHERE nome LIKE "J%";
```

DQL – Linguagem de Consulta de Dados

- Exemplo: localiza os cliente cujo nome contenha a string Santos.

```
SELECT *
FROM Cliente
WHERE nome LIKE "%Santos%";
```

DQL – Linguagem de Consulta de Dados

- Exemplo: localiza os clientes cujo o nome possua três caracteres:

```
SELECT *
FROM Cliente
WHERE nome LIKE "___";
```

DQL – Linguagem de Consulta de Dados

- Exemplo: selecione o nome e o cpf dos cliente e a marca, o modelo e o ano de seus respectivos automóveis da marca FIAT:

```
SELECT nome, cpf, marca, modelo, ano
FROM cliente,  automovel
WHERE cliente.CodCliente = automovel.CodCliente AND
            automovel.marca LIKE 'FIAT';
```

DQL – Linguagem de Consulta de Dados

- Exemplo: selecione o nome e o cpf dos cliente e a marca, o modelo e o ano de seus respectivos automóveis:

```
SELECT nome, cpf, marca, modelo, ano
FROM cliente, automovel
WHERE cliente.CodCliente = automovel.CodCliente;
```

- **Utiliza-se o nome da tabela a frente dos atributos para evitar ambiguidades**

DQL – Linguagem de Consulta de Dados

- Operação de renomeação
  - A SQL oferece a possibilidade de renomeação das tabelas e atributos através da cláusula **AS** tanto na cláusula **SELECT** quanto na **FROM**, tomando a forma:

nome_antigo **AS** nome_novo

DQL – Linguagem de Consulta de Dados

- Retornando ao exemplo anterior: selecione o nome e o cpf dos cliente e a marca, o modelo, ano de fabricação e ano modelo de seus respectivos automóveis da marca VW. Como saída o ano deve ser mostrado como “Ano de Fabricação” e “Ano Modelo ”:

```
SELECT nome, cpf, marca, modelo, ano_f  AS 'Ano de Fabricação',
                                        ano_m AS 'Ano Modelo'
FROM cliente AS c, automovel AS a
WHERE c.CodCliente  =  a.CodCliente AND
          a.marca LIKE 'FIAT';
```

DQL – Linguagem de Consulta de Dados

- Cláusula **ORDER BY**
  - Faz com que as tuplas no resultado de uma consulta apareçam na ordem classificada.
- Exemplo: selecione o nome e o cpf dos cliente e a marca, o modelo, ano de fabricação e ano modelo de seus respectivos automóveis da marca VW. Como saída o ano deve ser mostrado como “Ano de Fabricação” e “Ano Modelo ”, a saída deve ser ordena por modelo:

```
SELECT nome, cpf, marca, modelo, ano_f  AS 'Ano de Fabricação',
                                 ano_m AS 'Ano Modelo'
FROM cliente AS c, automovel AS a
WHERE c.CodCliente  =  a.CodCliente AND
          a.marca LIKE 'VW';
ORDER BY a.modelo;
```

DQL – Linguagem de Consulta de Dados

- Cláusula **ORDER BY**
  - Para especificar a ordem de classificação, pode-se especificar **DESC** para ordem decrescente e **ASC** para ordem crescente.

```
SELECT nome, cpf, marca, modelo, ano_f  AS 'Ano de Fabricação',
                                        ano_m AS 'Ano Modelo'
FROM cliente AS c, automovel AS a
WHERE c.CodCliente  =  a.CodCliente AND
          a.marca LIKE 'VW';
ORDER BY a.modelo DESC;
```

DQL – Linguagem de Consulta de Dados

- Cláusula **ORDER BY**
  - **ASC** para ordem crescente.

```
SELECT nome, cpf, marca, modelo, ano_f  AS 'Ano de Fabricação',
                                        ano_m AS 'Ano Modelo'
FROM cliente AS c, automovel AS a
WHERE c.CodCliente  =  a.CodCliente AND
          a.marca LIKE 'VW';
ORDER BY a.modelo ASC;
```

DQL – Linguagem de Consulta de Dados

- Cláusula **ORDER BY**
  - Ordenação em vários atributos:

```
SELECT nome, cpf, marca, modelo, ano_f AS 'Ano de Fabricação',
                                          ano_m AS 'Ano Modelo'
FROM cliente AS c, automovel AS a
WHERE c.CodCliente  =  a.CodCliente AND
          a.marca LIKE 'VW'
ORDER BY modelo DESC, ano_f ASC;
```

DQL – Linguagem de Consulta de Dados

- Cláusula **LIMIT**
  - Restringe o número de linhas retornadas pelo SELECT
- Exemplo: Listar os 3 primeiros clientes em ordem alfabética.

```
SELECT *
FROM cliente
ORDER BY nome
LIMIT 2;
```

DQL – Linguagem de Consulta de Dados

- Cláusula **LIMIT**
  - Restringe o número de linhas retornadas pelo SELECT
- Exemplo: Listar os 3 primeiros clientes em ordem aleatória.

```
SELECT *
FROM cliente
ORDER BY RAND()
LIMIT 3;
```

<- Função Randômica

DQL – Linguagem de Consulta de Dados

- Cláusula **LIMIT <início>, <quantos>**
  - Restringe também um intervalo de linhas retornadas pelo SELECT
- Exemplo: Listar os 4 clientes em ordem alfabética a partir do segundo cliente

```
SELECT * FROM cliente ORDER BY nome
LIMIT 2,4;
```

DQL – Linguagem de Consulta de Dados

- Funções agregadas
  - Média: avg()
    - Exemplo: média dos salário maiores que R$ 1.000,00
  - Mínimo: min();
    - Exemplo: selecionar o menor salário:
  - Máximo: max();
    - Exemplo: selecionar o maior salário:
  - Total: sum();
    - Exemplo: mostrar a soma dos salário:
  - Contar: count();
    - Exemplo: mostrar quantos funcionários ganham acima de R$ 1.000,00:

DQL – Linguagem de Consulta de Dados

- Cláusula **GROUP BY**
  - Permite a aplicação das funções agregadas em grupos.
  - Exemplo: selecionar a média dos ativos das agências agrupados por cidade:

```
SELECT Cidade, AVG(ativo)
FROM agencia
GROUP BY Cidade;
```

DQL – Linguagem de Consulta de Dados

- Cláusula **HAVING**
  - Define uma condição aplicada a um grupo
  - **Pode ser utilizada com funções enquanto o where não!**
  - Exemplo: selecionar o a media do ativo das agências agrupadas por Cidade, senda á média maior que R$ 300.000,00.

```
SELECT Cidade, AVG(ativo)
FROM agencia
GROUP BY Cidade
HAVING AVG(ativo) > 300000;
```

DQL – Linguagem de Consulta de Dados

- Valores Nulos
  - Podemos usar a palavra-chave especial NULL em um predicado para testar a presença de um valor nulo.
  - Exemplo: selecionar os empréstimos com valores nulos para Quantia:

```
SELECT *
FROM emprestimo
WHERE Quantia IS NULL;
```

VISÕES

Visões

- É uma relação que não armazena dados
- É composta por uma consulta que é previamente analisada e otimizada
- Diferente de tabelas, **visões não são objetos físicos, não ocupam espaço em disco**
- Alterações nos dados de tabelas que são acessadas por visões, consequentemente alteram os resultados gerados pelas consultas armazenadas nessas visões

Visões

- Vantagens
  - Aumento de **segurança por propiciar uma visão limitada** e controlada dos dados que podem ser obtidos da base
  - Aumento do **desempenho por utilizar uma consulta previamente otimizada**, tornando desnecessário este processo de otimização quando for realizada
  - Fornece mecanismo de segurança, restringindo o acesso de usuários
  - **Simplifica** a interação entre usuário final e banco de dados

Visões

- Sintaxe:
  - Para criar uma visão

CREATE VIEW nome_visao AS

(<expressão de consulta>);

  - Para apagar uma visão

DROP VIEW nome_visao;

Visões

- Exemplo:
  - Criar uma visão chamada todos_clientes contendo todos os clientes que possuem uma conta e um empréstimo no banco, sem duplicatas:

```
CREATE VIEW todos_clientes AS
(SELECT  c.Nome, c.Endereco
FROM cliente c, depositante d
WHERE c.CodCliente = d.CodCliente);
```

Visões

- Depois de criarmos uma visão podemos fazer consulta na mesma:

```
SELECT *
FROM todos_clientes;
```

SUB-CONSULTAS

Sub-Consultas

- Uma sub-consulta é uma consulta SELECT aninhada dentro de outro comando SQL
- Uma sub-consulta deve ser delimitada entre parênteses e é avaliada apenas uma vez
- O resultado de uma sub-consulta retorna um conjunto de linhas para a consulta principal
  - A consulta mais externa **depende** da sub-consulta

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

Sub-Consultas

- Retorno de uma sub-consulta
  - Uma sub-consulta de valor único retorna **apenas um valor** e pode ser usada no lugar de qualquer expressão utilizando **operadores (=, <, >, <>)**

1 coluna → 1 valor

```
WHERE A = (SELECT b...)  /*Verdade se A=B*/
```

Sub-Consultas

- Exemplo: Exibir o nome e o valor da gratificação das funções que têm a menor gratificação da empresa

```
SELECT nome, gratificacao
FROM funcao
WHERE gratificacao = (SELECT MIN(gratificacao)
FROM funcao)
```

Sub-Consultas

- Exemplo: Exibir o código, o nome e a quantidade em estoque do produto que tem a maior quantidade em, estoque da empresa

```
SELECT codigo, nome, quantEst
FROM produto
WHERE quantEst = (SELECT MAX(quantEst) FROM produto)
```

Sub-Consultas

- Retorno de uma sub-consulta
  - Quando uma sub-consulta retorna **múltiplas linhas** somente pode ser usada em um WHERE **utilizando cláusulas especiais**

1 coluna → muitos valores

  - Esta lista de valores retornada pode ser usada em comparações com o operador **IN**

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

Sub-Consultas

- Exemplo: Exibir código e nome de todos os clientes estrangeiros

```
SELECT codigo, nome FROM cliente
WHERE codPais IN (SELECT codigo FROM pais WHERE
  codigo <> 'BRA')
```

```
SELECT c.codigo, c.nome
FROM cliente c, pais p
WHERE c.codPais = p.codigo AND c.codPais <>'BRA'
```


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

Sub-Consultas

- Cláusulas especiais
  - IN
    - Pode ser considerado um apelido para “= ANY”. Significa “retornar TRUE se existe **algum elemento** na coluna que a sub-consulta retorna”
  - ALL
    - Deve seguir um operador de comparação, significa "retornar TRUE se a comparação é verdadeiro para **todos os valores** na coluna que a sub-consulta retorna”
  - ANY
    - Deve seguir um operador de comparação, significa "retornar TRUE se a comparação é verdadeira para **qualquer dos valores** na coluna que a sub-consulta retorna”

Sub-Consultas

- Lista de Valores Especiais
  - > ALL : maior que todos
  - < ALL : menor que todos
  - <> ALL : diferente de todos (igual a NOT IN)
  - != ALL : diferente de todos (igual a NOT IN)
  - = ANY : igual a algum dos elementos da lista (o mesmo que IN)
  - > ANY : maior que algum dos elementos da lista
  - < ANY: menor que algum dos elementos da lista
  - <> ANY : diferente de algum dos elementos da lista

Sub-Consultas

- Exemplo: exibir nome, tipo e preço de venda dos produtos que não sejam dos tipos 3, 4 ou 5, e que tenham preço de venda maior que pelo menos o preço de um destes produtos

```
SELECT nome, tipo, preco_venda
FROM produto
WHERE tipo NOT IN (3,4,5) AND preco_venda > ANY
      (SELECT preco_venda FROM produto WHERE
tipo IN (3,4,5))
```

JOINS

JOINS

- INNER JOIN
  - é uma forma de unir (join) tabelas, e para isso é preciso que haja uma relação entre as mesmas.
  - Sintaxe:

SELECT <lista de colunas>

FROM <Tabela01>

INNER JOIN <Tabela02> ON Tabela01.nome_coluna = Tabela02.nome_coluna

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

JOINS

- INNER JOIN
  - Exemplo: mostrar somente os funcionários que têm dependentes, mostrar todos os dependentes. Ordenar por nome dos funcionários:

```
SELECT f.Nome, d.Nome
FROM funcionarios f
INNER JOIN dependentes d ON f.CodFuncionarios = d.CodFuncionarios
ORDER BY f.Nome;
```

JOINS

- LEFT JOIN
  - Sintaxe:

SELECT <lista de colunas>

FROM <Tabela01>

LEFT JOIN <Tabela02> ON Tabela01.nome_coluna = Tabela02.nome_coluna

JOINS

- LEFT JOIN
  - Exemplo: selecionar todos os funcionários e seus dependentes, inclusive os funcionários que não tem dependentes. Ordenados por nome de funcionário.

```
SELECT f.Nome, d.Nome
FROM funcionarios f
LEFT JOIN dependentes d ON f.CodFuncionarios = d.CodFuncionarios
ORDER BY f.Nome;
```

JOINS

- RIGHT JOIN
  - Sintaxe:

SELECT <lista de colunas>

FROM <Tabela01>

RIGHT JOIN <Tabela02> ON Tabela01.nome_coluna = Tabela02.nome_coluna

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE

JOINS

- RIGHT JOIN
  - Exemplo: mostrar somente os funcionários que têm dependentes, mostrar todos os dependentes. Ordenar por nome dos funcionários.

```
SELECT f.Nome, d.Nome
FROM funcionarios f
RIGHT JOIN dependentes d ON f.CodFuncionarios = d.CodFuncionarios
ORDER BY f.Nome;
```

JOINS

- Com três tabelas
  - Exemplo: selecionar os funcionário que estão vinculado a pelo menos um projeto:

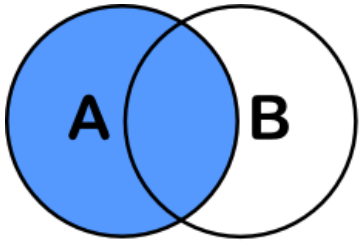
```
SELECT *
FROM trabalha t
INNER JOIN funcionarios f ON t.CodFuncionarios = f.CodFuncionarios
INNER JOIN projeto p ON t.CodProjeto = p.CodProjeto;
```

JOINS

- Com três tabelas
  - Exemplo: selecionar os funcionário que estão vinculado a pelo menos um projeto e mostrar os projetos mesmo sem nenhum funcionário vinculado:

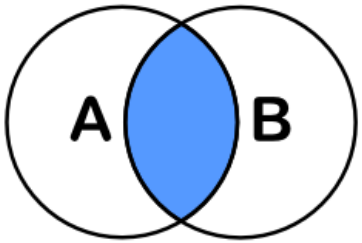
```
SELECT *
FROM trabalha t
INNER JOIN funcionarios f ON t.CodFuncionarios = f.CodFuncionarios
RIGHT JOIN projeto p ON t.CodProjeto = p.CodProjeto;
```

JOINS

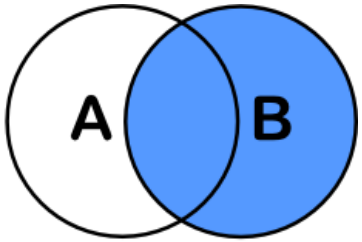



```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```

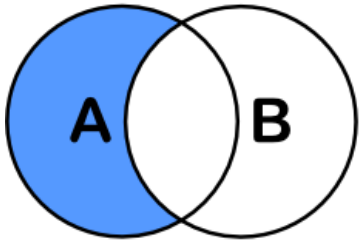
CHEATSHEET
SQL
JOINS



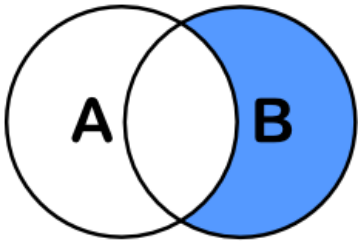

```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```



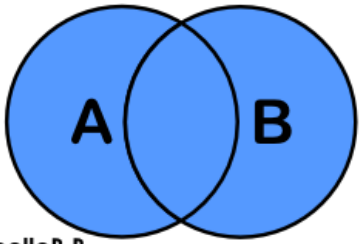

```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```



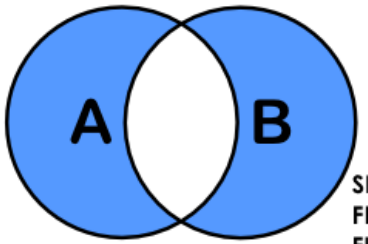

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

Revisão + Visões + Sub-Consultas + JOINS

Professor: Ricardo Luis dos Santos

IFSUL – Campus Sapucaia do Sul

© 2015 Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense

Campus Sapucaia do Sul