



Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA
Departamento de Informática
Análise e Desenvolvimento de Sistemas / Licenciatura em Computação

Comandos SQL - PARTE 2

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)

Comando **SELECT**

- Consulta registros em uma tabela
- **Sintaxe:**
 - **SELECT** <atributo1>[, <atributo2>, ...] **FROM** <nome_tabela> [**WHERE** <condicao>] ;
 - **SELECT * FROM** <nome_tabela> [**WHERE** <condicao>] ;
 - O * indica que queremos obter a informação de todos os atributos.
- **Ex: SELECT * FROM** Categoria
 - Liste todos os atributos de todos os registros da tabela “Categoria”

Exemplos do comando **SELECT**

- **SELECT * FROM** Categoria
 - Liste todos os atributos de todos os registros da tabela “Categoria”

<u>id</u>	nome_categoria	descricao
1	Limpeza	Produtos de Limpeza
2	Doces	Produtos comestíveis doces
3	Cosmeticos	Produtos cosméticos

Comando **SELECT** e expressões aritméticas

- A consulta **SELECT** pode conter expressões aritméticas (+ - / *) envolvendo constantes ou atributos. **Exemplos:**
 - **SELECT** num_emprestimo, nome_agencia, quantia * 100
FROM Emprestimo
 - Mostre os números dos empréstimos, nome da agencia e quantia multiplicada por 100
 - **SELECT** nome_produto, qtd_vendida * preco_venda
FROM Vendas
 - Mostre o nome do produto e o total vendido de cada produto (em reais)

Comando **SELECT** e múltiplas tabelas

- Podemos ter consultas usando mais de uma tabela de uma vez. **Exemplo:**
 - SELECT** Aluno.nome, Professor.nome **FROM** Aluno, Professor

Aluno	
nome	turma
Edvan	632
Maria	631
Carlos	632

Professor	
nome	disciplina
André	BD
Geisa	Filosofia
Adson	Web



nome	nome
Edvan	André
Edvan	Geisa
Edvan	Adson
Maria	André
Maria	Geisa
Maria	Adson
Carlos	André
Carlos	Geisa
Carlos	Adson

Como não temos **WHERE**, essa consulta SQL vai mostrar todas as combinações entre Aluno.nome e Professor.nome (**produto cartesiano**)

Nomes Qualificados de Atributos

- **SELECT** Aluno.nome, Professor.nome **FROM** Aluno, Professor
 - Observe que temos atributos com os mesmos nomes em tabelas diferentes
 - **Ex:** Em Aluno temos um atributo nome e em Professor também
- Logo, precisamos especificar o nome dos atributos no **SELECT** de forma **qualificada**
 - **Sintaxe:** nome_tabela.nome_atributo
 - **Ex:** Aluno.nome
 - **Ex:** Professor.cpf

Exemplos do comando **SELECT**

- **SELECT** nome_categoria, descricao **FROM** Categoria **WHERE** nome_categoria = 'Limpeza'

nome_categoria	descricao
Limpeza	Produtos de Limpeza

- **SELECT** descricao **FROM** Categoria **WHERE** nome_categoria = 'Limpeza' **OR** id = 3

descricao
Produtos de Limpeza
Produtos cosméticos

A consulta **SELECT** mostra todos os registros, mesmo os duplicados

Comando **SELECT DISTINCT**

- Consulta registros únicos (distintos) em uma tabela
- **Sintaxe:**
 - **SELECT DISTINCT** <atributo1>[, <atributo2>, ...] **FROM** <nome_tabela> [**WHERE** <condicao>] ;
 - **SELECT DISTINCT * FROM** <nome_tabela> [**WHERE** <condicao>] ;
 - O * indica que queremos obter a informação de todos os atributos.
- **Ex: SELECT DISTINCT * FROM** Categoria
 - Liste todos os atributos de todos os registros únicos da tabela “Categoria”

Comando **AS**

- Renomeia atributos ou tabelas
 - **Porque?**
 - Quando dois atributos tem mesmo nome em um **SELECT** eles aparecem duplicados no resultado
 - **Ex: SELECT** emprestimo.nome_cliente, tomador.nome_cliente **FROM** emprestimo, tomador ;
 - Se usarmos uma expressão aritmética no **SELECT**, o atributo resultante não possuirá um nome
 - **Ex: SELECT** qtd_vendida * preco **FROM** vendas ;

Comando **AS**

- Renomeia atributos ou tabelas
 - **Porque?**
 - Podemos querer mudar o nome do atributo na tabela de resultado, por alguma razão
 - **Ex: SELECT** qtd_vendida **AS** qtd **FROM** vendas ;
- **Sintaxe:**
 - **SELECT** <nome_antigo> **AS** <nome_novo> **FROM** <tabela>;
 - **SELECT** <atributos> **FROM** <tabela_antigo> **AS** <tabela_novo> ;

Exemplo do comando **AS**

- **SELECT** A.nome, P.nome
FROM Aluno **AS** A, Professor **AS** P;
 - Mostre o nome dos alunos e professores
 - Use *A* e *P* como **variáveis de tupla** que representam as tabelas *Aluno* e *Professor* respectivamente

Comando **ORDER BY**

- Ordenar um conjunto de resultados em ordem crescente (*padrão do comando ORDER BY*) ou decrescente
- **Sintaxe:**
 - **SELECT** <atributos> **FROM** <nome_tabela> [**WHERE** <cond>] [**ORDER BY** <atributo>] [**DESC**];
- A ordem dos comandos SQL importa
 - **ORDER BY** deve vir sempre após o **WHERE**
 - **DESC** indica que queremos os resultados classificados em ordem decrescente

Comando ORDER BY

- Ex: **SELECT** nome **FROM** Funcionario **ORDER BY** nome;
 - Liste os registros de “Funcionario”, ordenados de forma crescente pelo “nome”

	MariaDB	MariaDB.1	MariaDB.2	MariaDB.3
1 SELECT * FROM Funcionarios ORDER BY nome;				
!	id	nome	cargo	
4		Carla	Atendimento ao cliente	
1		Felipe	Analista de desenvolvimento	
2		Jose	Analista senior	
3		Maria	Analista de qualidade de software	

Funções de Agregação SQL

- Processa um conjunto de valores contidos em um atributo de uma tabela e retorna um único valor como resultado
 - **Ex:** COUNT, MAX, MIN, AVG, SUM
- **Sintaxe:**
 - **SELECT** <nome_funcao(atributo)>
FROM <tabela>

WHERE nao permite o uso de
funcoes de agregação

Valores nulos são excluídos
antes da função de agregação
ser executada

Funções de agregação só
podem ser usadas com os
comandos **SELECT** ou **HAVING**

Exemplos de Funções de Agregação SQL

- **SELECT MAX(preco) FROM Produto**
 - Encontre o maior preço dentre todos os produtos da loja
- **SELECT MIN(preco) FROM Produto**
 - Encontre o menor preço dentre todos os produtos da loja
- **SELECT AVG(preco) FROM Produto**
 - Encontre o média dos preços de todos os produtos da loja
- **SELECT SUM(preco * qtd) FROM Produto**
 - Encontre o valor total dos produtos em estoque na loja

Exemplos de Funções de Agregação SQL

- **SELECT COUNT(preço) FROM Produto WHERE nome = 'arroz'**
 - Conte os registros cujo atributo preço não é **NULL**
 - Isto é, conte quantos produtos tem o nome = 'arroz'
 - Se preço for **NULL**, o registro não será contado
 - Isto é produtos sem preço serão ignorados na contagem
- **SELECT COUNT(*) FROM Produto WHERE nome = 'arroz'**
 - Conta quantos registros tem o atributo nome = 'arroz'
 - A contagem considera TODOS OS REGISTROS, mesmos aqueles com atributos **NULL**

Comando **GROUP BY**

- Agrupa registros em grupos de valores, fornecendo um resultado para cada grupo
 - Registros são agrupados por atributos com valores em comum
- **Sintaxe:**
 - **SELECT** <atributos>
FROM <nome_tabela>
[WHERE <condição>
GROUP BY <atributo>
- **Ex: SELECT AVG(salario) FROM Funcionario GROUP BY num_dep;**

Exemplos do comando **GROUP BY**

- **SELECT Dnr, COUNT(*), AVG(Salario) FROM Funcionario GROUP BY Dnr**
 - Agrupe os funcionários por departamento (**Dnr**) e calcule a quantidade de funcionários (**COUNT**) e média dos salários (**AVG**)

Pnome	Minicial	Unome	Cpf	...	Salario	Cpf_supervisor	Dnr		Dnr	Count (*)	Avg (Salario)
João	B	Silva	12345678966		30.000	33344555587	5	}	5	4	33,250
Fernando	T	Wong	33344555587		40.000	88866555576	5		4	3	31.000
Ronaldo	K	Lima	66688444476		38.000	33344555587	5		1	1	55.000
Joice	A	Leite	45345345376	...	25.000	33344555587	5				
Alice	J	Zelaya	99988777767		25.000	98765432168	4	}			
Jennifer	S	Souza	98765432168		43.000	88866555576	4				
André	V	Pereira	98798798733		25.000	98765432168	4				
Jorge	E	Brito	88866555576		55.000	NULL	1	}			

Comando **HAVING**

- Comando **HAVING** filtra os resultados que serão submetidos a agregação (só pode ser utilizado quando o GROUP BY também é utilizado)
 - Os filtros do **HAVING** executam após a função de agregação ter sido executada
- **Sintaxe:**
 - **SELECT** <lista_atributos>
FROM <tabela>
GROUP BY <atributo>
HAVING <condição_que_PODE_conter_funções_de_agregação>

Exemplos do comando **HAVING**

- **SELECT MAX(preco) FROM Produto GROUP BY id_categoria HAVING MAX(preco) < 10 ;**
 - Encontre o preço máximo de cada categoria de produto, desde que esse preço seja < 10 reais
- **SELECT COUNT(id_consumidor), pais FROM Consumidor GROUP BY pais HAVING COUNT(id_consumidor) > 5 ;**
 - Conte todos os consumidores nascidos em cada país e mostre a contagem apenas quando existir mais de 5 consumidores

Comando **DISTINCT**

- Se quisermos remover valores duplicados antes de usar a função agregada, precisamos do comando **DISTINCT**
- **Sintaxe:**
 - nome_função_agregação(**DISTINCT** <atributo>)
- **Ex:**
 - **SELECT** nome_agencia, **COUNT**(**DISTINCT** nome_cliente)
FROM conta
GROUP BY nome_agencia
 - Conte o número de clientes de acordo com o nome da agência

Consultas com valores NULOS e UNKNOWN

- Valores nulos são representados pela palavra chave **NULL**
- Devemos evitar valores nulos sempre que possível pois:
 - Operações aritméticas (+ - / *) envolvendo valores **NULL** tem como resultado um valor **NULL**
 - **Ex: SELECT** quantia * (1+tx_juros) **FROM** emprestimo;
 - Se quantia for **NULL**, o resultado da operação também será
 - Quando comparamos valores **NULL** no SQL, o resultado da comparação é um valor **UNKNOWN**
 - **Ex: SELECT** quantia > 0 **FROM** emprestimo;

Comando IS

- Usado para testar se um valor é **NULL** ou **UNKNOWN**
- **Sintaxe:**
 - **SELECT** <atributos>
FROM <nome_tabela>
WHERE <nome_atributo> **IS [NOT] NULL;**
 - **SELECT** <atributos>
FROM <nome_tabela>
WHERE <nome_atributo> **IS [NOT] UNKNOWN;**

Exemplo de comando IS

- **SELECT** Pnome, Unome **FROM** Funcionario **WHERE** Cpf_supervisor **IS NULL**
 - Pegue o nome dos funcionários que não possuem supervisores
- **SELECT** cliente **FROM** Emprestimo **WHERE** (quantia > 0) **IS NOT UNKNOWN**
 - Pegue o nome dos clientes com empréstimos com quantia > 0
- **SELECT** email **FROM** Aluno **WHERE** email **IS NOT NULL**
 - Pegue o email dos alunos que possuem email institucional cadastrado

Operações com Strings

- Strings são representados na linguagem SQL como textos em aspas simples
 - **Ex:** 'valença'
- Para representar um apóstrofo dentro de um texto, usamos dois apóstrofos simples no texto
 - **Ex:** 'caixa d"água'

Comando LIKE

- Compara strings
- **Sintaxe:**

Note que caracteres maiúsculos e minúsculos são importantes para comparações entre strings.

Ex: 'Amanda' não é igual a 'amanda'

- **SELECT** <atributo> **FROM** <tabela> **WHERE** <atributo_string> **[NOT] LIKE** <condição_string> ;
- **Ex:**
 - **SELECT** nome **FROM** funcionario **WHERE** nome **LIKE** 'Amanda' ;
 - Encontre todos os funcionários com nome 'Amanda'

Porque usar o comando **LIKE** ao invés do **=** ?

- O comparador de igualdade (**=**) não permite o uso de caracteres especiais para pesquisar textos, como o **%** e o **_**
- Isto é, para o operador de comparação **=** esses caracteres não tem nenhum significado especial (são caracteres comuns)
 - **Ex: SELECT * FROM cidade WHERE nome = 'valen%'**
 - Busca pela string 'valen' seguida do caractere percentual (%)
 - **Ex: SELECT * FROM cidade WHERE nome LIKE 'valen%'**
 - Busca por strings que começam com 'valen'

Caracteres Especiais em Operações com Strings

- Os caracteres % e _ tem significado especial
 - '%' : corresponde a qualquer substring
 - **Ex:** 'valen%' localiza qualquer string começando com 'valen'
 - **Ex:** '%len%' localiza qualquer string que contenha 'len'
 - '_' : corresponde a um caractere
 - **Ex:** 'valen_a' localiza qualquer string começando com 'valen' e terminando com 'a'
 - **Ex:** '_alenç_' localiza qualquer string que possui um caractere seguido de 'alenç' seguido de outro caractere qualquer

Exemplos do Comando **LIKE** com Caracteres Especiais

- **Ex: SELECT** nome **FROM** funcionario **WHERE** nome **NOT LIKE** 'Amanda%';
 - Encontre todos os funcionários cujo nome não começa com 'Amanda'
- **Ex: SELECT** nome **FROM** funcionario **WHERE** nome **LIKE** '_manda';
 - Encontre todos os funcionários cujo nome começa com qualquer caracter e termina com 'manda'

Exemplos do Comando **LIKE** sem Caracteres Especiais

- **Ex: SELECT** nome **FROM** funcionario **WHERE** nome **LIKE** 'Amanda' ;
 - Encontre todos os funcionários cujo nome é exatamente 'Amanda'
 - Se o funcionário possuir um nome maior como 'Amanda Silva', não iremos encontrar esse registro

Usando caracteres especiais como texto simples

- Como pesquisar um texto que contenha os caracteres % ou _ ?
 - Usamos o caractere **escape** (\), também chamado de barra invertida, antes do caractere especial que queremos usar
 - **Ex: SELECT * FROM produto WHERE descricao LIKE '70\% cacau'**
 - Encontre os nomes dos produtos cuja descrição é o texto '70% cacau'
 - **Ex: SELECT nome_turma FROM turma WHERE nome_turma LIKE '631_ensino_medio'**
 - Encontre a turma com nome igual a '631_ensino_medio'

Comando **UPPER** e **LOWER**

- Coloca os caracteres de uma string como maiúsculos (**UPPER**) ou minúsculos (**LOWER**)
- **Sintaxe:**
 - **SELECT UPPER(<atributo>) FROM <tabela>;**
 - **SELECT LOWER(<atributo>) FROM <tabela>;**
- **Ex: SELECT UPPER(nome) FROM funcionario ;**
 - Encontre o nome de todos os funcionários e mostre em MAIÚSCULO
- **Ex: SELECT LOWER(nome) FROM funcionario ;**

Operação Booleana - AND

- **A AND B**
 - True **AND** Unknown = Unknown
 - False **AND** Unknown = False
 - Unknown **AND** Unknown = Unknown
 - True **AND** False = False
 - False **AND** False = False
 - True **AND** True = True

A operação **AND** é
comutativa

Ex: True **AND** False
=
False **AND** True

Operação Booleana - OR

- **A OR B**
 - True **OR** Unknown = True
 - False **OR** Unknown = Unknown
 - Unknown **OR** Unknown = Unknown
 - True **OR** False = True
 - False **OR** False = False
 - True **OR** True = True

A operação **OR** é
comutativa

Ex: True **OR** False
=
False **OR** True

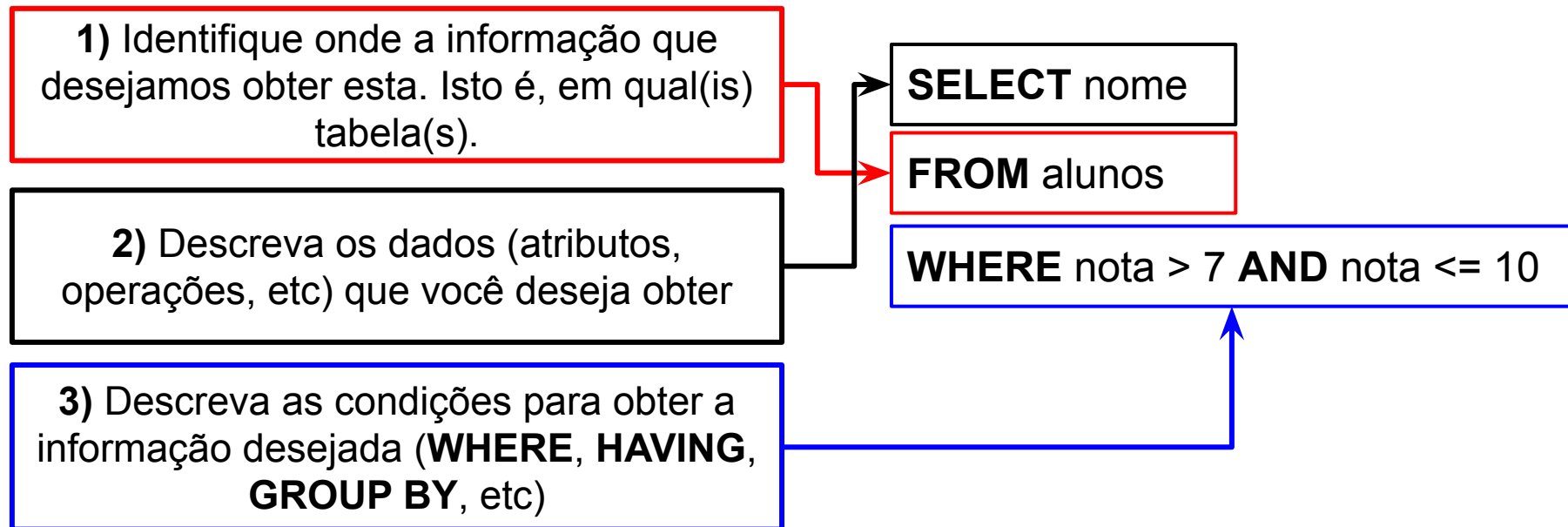
Operação Booleana - NOT

- **NOT A**
 - **NOT** True = False
 - **NOT** False = True
 - **NOT** Unknown = Unknown

Operações de Comparação

- O SQL suporta várias operações de comparação, como:
 - `>`, `>=`, `<`, `<=`, `=`, `<>`
- O resultado de cada operação de comparação pode ser um dos valores booleanos: **TRUE**, **FALSE**, **UNKNOWN**
- **Ex: SELECT** tx_juros **FROM** emprestimo **WHERE** quantia > 5000 **AND** quantia < 10000 ;
 - Mostre a taxa de juros dos empréstimos cujas quantias estejam entre 5.000,00 e 10.000,00 reais

Regra 123 : Como criar consultas SQL ?



Exemplo de Banco de Dados

- Seja um banco de dados definido pelas tabelas: **Aluno**(cpf, nome, data_nasc, nome_disciplina, nota_disciplina, faltas_disciplina, status_disciplina), **TurmaAluno**(cod_turma, *cpf_aluno*)
 - Assuma que os atributos sublinhados são as chaves primárias e os atributos em *italico* são as chaves estrangeiras

Exemplo de Consulta (SELECT / FROM / WHERE)

- Seja um banco de dados definido pelas tabelas: **Aluno**(cpf, nome, data_nasc, nome_disciplina, nota_disciplina, faltas_disciplina, status_disciplina), **TurmaAluno**(cod_turma, *cpf_aluno*)
- Crie uma consulta para encontrar o nome e a nota de cada aluno da disciplina 'Banco de Dados' (considere que o nome da disciplina pode estar escrito em minúsculo, maiúsculo ou uma mistura dos dois)

2) **SELECT** nome, nota

1) **FROM** Aluno

3) **WHERE LOWER(nome_disciplina) LIKE 'banco de dados'**

Exemplo de Consulta (SELECT / FROM / GROUP BY)

- Seja um banco de dados definido pelas tabelas: **Aluno**(cpf, nome, data_nasc, nome_disciplina, nota_disciplina, faltas_disciplina, status_disciplina), **TurmaAluno**(cod_turma, *cpf_aluno*)
- Encontre a média das notas dos alunos por disciplina

2) **SELECT** nome_disciplina, **AVG**(nota)

1) **FROM** Aluno

3) **GROUP BY** nome_disciplina

Exemplo de Consulta

(SELECT / FROM / GROUP BY / HAVING)

- Seja um banco de dados definido pelas tabelas: **Aluno**(cpf, nome, data_nasc, nome_disciplina, nota_disciplina, faltas_disciplina, status_disciplina), **TurmaAluno**(cod_turma, *cpf_aluno*)
- Conte quantos alunos foram aprovados em cada disciplina do curso técnico em computação (assuma a carga horária de 72h para cada disciplina)

2) **SELECT** nome_disciplina, **COUNT**(*)

1) **FROM** Aluno

3) **GROUP BY** nome_disciplina
HAVING nota > 6.0 **AND** faltas_disciplina <= 72*0.75

Revisão - Ordem de de escrita dos comandos SQL

Ordem dos comandos SQL

SELECT lista_atributos

FROM tabela

[**WHERE** condições_do_select]

[**GROUP BY** lista_atributos]

[**HAVING** condições_do_group_by]

[**ORDER BY** lista_atributos]

O **WHERE** define as condições que devem ser satisfeitas ANTES do **SELECT** ser executado

O **HAVING** define as condições que devem ser satisfeitas ANTES do **GROUP BY** ser executado e APÓS o **WHERE**

Revisão - Ordem de escrita dos comandos SQL

Ordem dos comandos SQL

SELECT lista_atributos

FROM tabela

[WHERE condições_do_select]

[GROUP BY lista_atributos]

[HAVING condições_do_group_by]

[ORDER BY lista_atributos]

Como memorizar a ordem?

Sandy é

Fascinada por

Wesley safadão porque

George clooney

Hesitou

Ontem

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.