



Banco de Dados I
Helio Rangel

A Linguagem SQL nos Bancos Relacionais

SQL

Considerações gerais:

- A Linguagem SQL **não** é sensível ao caso, isto é, não importa se os nomes das entidade, atributos, comando etc. estão grafados em maiúsculas ou minúsculas. É uma boa prática escrever os comandos em letras maiúsculas.

Exemplo: SELECT, INSERT, UPDATE, FROM, DELETE, WHERE etc.

- Em SQL o texto é escrito entre aspas simples, portanto, caso no texto que for inserido exista uma aspas simples, duplique a aspa simples para que não retorne um erro sintático na query.

Exemplo Problema: UPDATE TAB_Aluno set nome='Jose D'Silva' WHERE cod_aluno='541234'.

Exemplo Solução: UPDATE TAB_Aluno set nome='Jose D"Silva' WHERE cod_aluno='541234'

- Cuidado quando sua query for inserir ou atualizar números. Para o banco, a vírgula decimal (em português usamos vírgula e não ponto) não é considerada separador decimal e sim separador de milhares. Isso pode confundir e trazer resultados errados em suas consultas. Portanto utilize apenas um ponto para separador decimal e retire as vírgulas do número antes de submeter (rodar) a query;

Exemplo Problema: UPDATE TAB_Curso set cargaHoraria = '12,5' Where codigoCurso=1000;

Exemplo Solução: UPDATE TAB_Curso set cargaHoraria = '12.5' Where codigoCurso=1000;

A Linguagem SQL nos Bancos Relacionais

SQL

Considerações gerais:

- Trabalhar com datas também pode ser um problema. O formato da data pode estar configurado para o formato brasileiro, ou não. Uma forma de ficar independente do formato de datas do banco é utilizar o formato: 'YYYY-MM-DD'. Lembre-se de escrever as datas sempre entre aspas simples;

Exemplo: `SELECT nome FROM TAB_Aluno WHERE data_nascimento = '2001-01-23'`

Naturalmente, se você estiver certo que o banco esta configurado para data BR, você pode usar a data da forma que estamos acostumados e ignorar essa dica:

Exemplo: `SELECT nome FROM TAB_Aluno WHERE data_nascimento = '23/01/2001'`

- Mais um alerta sobre datas: Se o tipo de data for datetime (tipo que armazena data e hora), isso quer dizer que o banco irá armazenar as datas no formato ano-mês-dia-hora-minuto-segundo, ou seja, se eu pegar a data do servidor neste exato momento (que estou digitando isso aqui) vou obter: 2022-01-15 16:22:15.

O que você espera de resultado com esta query se o campo data_nota for um campo datetime?

`SELECT nota_Fiscal FROM TAB_Notas WHERE data_nota = '2022-01-15'`



A Linguagem SQL nos Bancos Relacionais

SQL

O SELECT

É a instrução que permite a consulta direta aos dados que estão armazenados no banco.

Sintaxe: `SELECT atributo1 [,atributo2, ... , atributoN] FROM entidade [WHERE chave=valor]`

Assim as seguintes queries são válidas como SELECT:

`SELECT Codigo_Aluno FROM TAB_Aluno`
Retorna todos os códigos dos alunos cadastrados

`SELECT Nome_Aluno FROM TAB_Aluno WHERE Codigo_Aluno=541234;`
Retorna o nome do aluno cujo código é '541234'

`SELECT * FROM TAB_Aluno`
Retorna todas as linhas e todas as colunas da tabela.

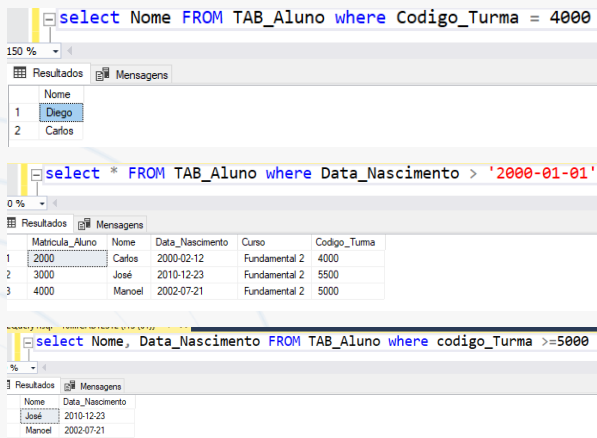
Este tipo de consulta não é recomendado porque normalmente não precisamos o retorno de toda a tabela. O Banco levará mais tempo e ocupará mais memória do que se a consulta estivesse melhor filtrada. Por isso se recomenda que a query retorne apenas as linhas e colunas que são uteis no momento da consulta.



A Linguagem SQL nos Bancos Relacionais

O SELECT – Exemplos práticos

Nota: Todas as consultas serão sobre a TAB_Aluno ilustrada a direita



The screenshot shows a SQL query editor with three queries and their results:

```
select Nome FROM TAB_Aluno where Codigo_Turma = 4000
```

Nome
1 Diego
2 Carlos

```
select * FROM TAB_Aluno where Data_Nascimento > '2000-01-01'
```

Matricula_Aluno	Nome	Data_Nascimento	Curso	Codigo_Turma
1 2000	Carlos	2000-02-12	Fundamental 2	4000
2 3000	José	2010-12-23	Fundamental 2	5500
3 4000	Manoel	2002-07-21	Fundamental 2	5000

```
select Nome, Data_Nascimento FROM TAB_Aluno where codigo_Turma >=5000
```

Nome	Data_Nascimento
José	2010-12-23
Manoel	2002-07-21

TAB_Aluno

	Matricula_Aluno	Nome	Data_Nascimento	Curso	Codigo_Turma
1	1000	Diego	1995-01-07	Fundamental 2	4000
2	2000	Carlos	2000-02-12	Fundamental 2	4000
3	3000	José	2010-12-23	Fundamental 2	5500
4	4000	Manoel	2002-07-21	Fundamental 2	5000



A Linguagem SQL nos Bancos Relacionais

O UPDATE

É a instrução que permite atualizar/modificar os dados que estão armazenados no banco. É necessário definir que tabela, que atributo e que linhas devem ser modificadas.

Sintaxe: UPDATE entidade SET campo1=valor[,campo2=valor, ..., campoN=valor] [WHERE chave=valor]

Assim as seguintes queries são válidas como UPDATE:

UPDATE TAB_Aluno set nome='Diego Augusto' WHERE Matricula_Aluno='1000'
Troca o nome do aluno código de matricula 1000

UPDATE TAB_Aluno set nome_curso='fundamental 1' WHERE Codigo_Turma=4000;
Troca os dois alunos, Diego e Carlos, de Fundamenta2 para Fundamental1

UPDATE TAB_Aluno set Codigo_Turma = '5000'
Troca TODAS os codigos_turma de todos os alunos para 5000

Ops!!! Update sem where... Deu ruim!



A Linguagem SQL nos Bancos Relacionais

O UPDATE – Exemplos práticos

```
UPDATE TAB_Aluno set nome='Diego Augusto' WHERE Matricula_Aluno='1000'
```

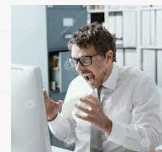
Matricula_Aluno	Nome	Data_Nascimento	Curso	Codigo_Turma
1000	Diego Augusto	1995-01-07	Fundamental 2	4000
2000	Carlos	2000-02-12	Fundamental 2	4000
3000	José	2010-12-23	Fundamental 2	5500
4000	Manoel	2002-07-21	Fundamental 2	5000

```
UPDATE TAB_Aluno set curso='fundamental 1' WHERE Codigo_Turma=4000;
```

Matricula_Aluno	Nome	Data_Nascimento	Curso	Codigo_Turma
1000	Diego Augusto	1995-01-07	Fundamental 1	4000
2000	Carlos	2000-02-12	Fundamental 1	4000
3000	José	2010-12-23	Fundamental 2	5500
4000	Manoel	2002-07-21	Fundamental 2	5000

```
UPDATE TAB_Aluno set Codigo_Turma = '5000'
```

Este último exemplo na realidade é uma armadilha tipo pesadelo se for realizado em uma base de dados de produção. Com um update deste (update sem *where*), podemos trocar uma coluna inteira de uma tabela por um mesmo valor (no caso, todos os códigos de turma ficarão iguais a 5000). As vezes pode ser isso mesmo que queremos, se estiver inicializando uma coluna por exemplo, mas o mais provável é que executamos este comando por engano, simplesmente esquecendo de definir que linhas deveriam ser afetadas. Se não tiver backup atualizado... Já era.



A Linguagem SQL nos Bancos Relacionais

O INSERT

Esta instrução permite inserir novos dados no banco de dados

Sintaxe: INSERT INTO entidade (nomeCampo1[,nomeCampo2,...,nomeCampoN) VALUES (valor1[,valor2, ..., valor)

Exemplos:

```
INSERT INTO TAB_Aluno(matricula_aluno,nome, data_nascimento, curso, codigo_turma) VALUES ('5000', 'Ricardo', '1991-05-01', 'Direito', '8000');
```

```
INSERT INTO TAB_Turma (codigo_turma, Nome_Turma) VALUES (8000, 'Direito');
```

```
INSERT INTO TAB_Aluno VALUES ('6000', 'Indeciso', '2000-01-01', 'Direito', NULL)
```

Neste último INSERT é uma forma alternativa, note que foi suprimida a lista de campos. Isso porque existe um conteúdo para cada um dos atributos da tabela. Neste caso a query coloca os campos na tabela na ordem que eles aparecem.

TAB_Turma

	Codigo_Turma	Nome_Turma
1	4000	Quinta Série A
2	5000	Quinta Série B
3	5500	Sexta série A
4	6000	Sexta série B
5	8000	Direito

TAB_Aluno

Matricula_Aluno	Nome	Data_Nascimento	Curso	Codigo_Turma
1000	Diego Augusto	1995-01-07	Fundamental 1	4000
2000	Carlos	2000-02-12	Fundamental 1	4000
3000	José	2010-12-23	Fundamental 2	5500
4000	Manoel	2002-07-21	Fundamental 2	5000
5000	Ricardo	1991-05-01	Direito	8000
6000	Indeciso	2000-01-01	Direito	NULL



A Linguagem SQL nos Bancos Relacionais

Fazendo consulta de mais de uma tabela

É bem simples fazer consultas de tabelas distintas ligadas por suas chaves primárias e estrangeiras. Precisamos identificar quem é quem em cada uma das tabelas, igualar os campos PK da tabela 1 = FK da tabela 2 e está feito

Exemplo:

```
SELECT Matricula_Aluno, Nome, Data_Nascimento, TAB_Turma.Codigo_Turma, nome_Turma
FROM TAB_Aluno, TAB_Turma where TAB_Turma.Codigo_Turma = TAB_Aluno.Codigo_Turma
```

Matricula_Aluno	Nome	Data_Nascimento	Codigo_Turma	nome_Turma
1000	Diego Augusto	1995-01-07	4000	Quinta Série A
2000	Carlos	2000-02-12	4000	Quinta Série A
3000	José	2010-12-23	5500	Sexta série A
4000	Manoel	2002-07-21	5000	Quinta Série B
5000	Ricardo	1991-05-01	8000	Direito

Podemos filtrar melhor nossa query, mostrando apenas os alunos da quinta série A.

```
SELECT Matricula_Aluno, Nome, Data_Nascimento, TAB_Turma.Codigo_Turma, nome_Turma
FROM TAB_Aluno, TAB_Turma where TAB_Turma.Codigo_Turma = TAB_Aluno.Codigo_Turma
AND TAB_Turma.codigo_Turma=4000
```

Matricula_Aluno	Nome	Data_Nascimento	Codigo_Turma	nome_Turma
1000	Diego Augusto	1995-01-07	4000	Quinta Série A
2000	Carlos	2000-02-12	4000	Quinta Série A



A Linguagem SQL nos Bancos Relacionais

Fazendo consulta de mais de uma tabela

Nota importante: Quando relacionamos tabelas que o nome do atributo na tabela 1 é idêntico ao nome do atributo da tabela 2 (Codigo_Turma por exemplo), e isso é bastante comum, colocamos o nome da tabela seguido de um ponto, seguido do nome do atributo. Isto é necessário para que o SGBD saiba qual dos atributos está sendo referenciado.

Podemos também criar apelidos para as tabelas de forma que os relacionamentos e os filtros (filtros=where) fiquem mais enxutos, menos poluídos. Vamos repetir esta última query, criando apelidos para as tabelas:

```
SELECT Matricula_Aluno, Nome, Data_Nascimento, T.Codigo_Turma, nome_Turma
FROM TAB_Aluno as A, TAB_Turma as T where T.Codigo_Turma = A.Codigo_Turma
AND T.codigo_Turma=4000
```

A tabela TAB_Aluno ganhou o apelido de A e a tabela TAB_Turma ganhou o apelido de T. O resultado é exatamente o mesmo

Matricula_Aluno	Nome	Data_Nascimento	Codigo_Turma	nome_Turma
1000	Diego Augusto	1995-01-07	4000	Quinta Série A
2000	Carlos	2000-02-12	4000	Quinta Série A

Alguns bancos o 'as' entre o nome da tabela e o apelido é opcional:

Em vez de `FROM TAB_Aluno as A,` pode ser `FROM TAB_Aluno A`



A Linguagem SQL nos Bancos Relacionais

Fazendo consulta de mais de uma tabela utilizando JOIN

Nos exemplos anteriores relacionamos diretamente as chaves da tabela TAB_Aluno com a TAB_Curso igualando TAB_Aluno.Codigo_Turma com TAB_Curso.Codigo_Turma. Essa forma de relacionar funciona bem mas se for o caso de mostrar algum aluno que ainda não está matriculado em nenhum curso, esta query não vai trazer porque ela exige que exista um valor de Codigo_Turma na coluna de turma, e não é o caso.

```
SELECT Matricula_Aluno, Nome, Data_Nascimento, T.Codigo_Turma, nome_Turma FROM  
TAB_Aluno as A INNER JOIN  
TAB_Turma as T ON A.Codigo_Turma = T.Codigo_Turma
```

Retorna todos os alunos exceto o que não se esta em nenhuma turma. Isso porque usamos INNER JOIN que equivale ao relacionamento direto utilizando os campos chave. Vamos tentar novamente utilizando o LEFT JOIN

Matricula_Aluno	Nome	Data_Nascimento	Codigo_Turma	nome_Turma	curso
1000	Diego Augusto	1995-01-07	4000	Quinta Série A	fundamental 1
2000	Carlos	2000-02-12	4000	Quinta Série A	fundamental 1
3000	José	2010-12-23	5500	Sexta série A	Fundamental 2
4000	Manoel	2002-07-21	5000	Quinta Série B	Fundamental 2
5000	Ricardo	1991-05-01	8000	Direito	Direito



A Linguagem SQL nos Bancos Relacionais

Fazendo consulta de mais de uma tabela utilizando LEFT JOIN

```
SELECT Matricula_Aluno, Nome, Data_Nascimento, T.Codigo_Turma, nome_Turma, curso FROM  
TAB_Aluno as A LEFT JOIN  
TAB_Turma as T ON A.Codigo_Turma = T.Codigo_Turma
```

Agora a consulta retorna todos os alunos inclusive o aluno que não esta inscrito em nenhuma turma .

Matricula_Aluno	Nome	Data_Nascimento	Codigo_Turma	nome_Turma	curso
1000	Diego Augusto	1995-01-07	4000	Quinta Série A	fundamental 1
2000	Carlos	2000-02-12	4000	Quinta Série A	fundamental 1
3000	José	2010-12-23	5500	Sexta série A	Fundamental 2
4000	Manoel	2002-07-21	5000	Quinta Série B	Fundamental 2
5000	Ricardo	1991-05-01	8000	Direito	Direito
6000	Indeciso	2000-01-01	NULL	NULL	NULL

