



BandTec

DIGITAL SCHOOL

SQL – Structured Query Language

- Desenvolvido no início dos anos 1970, pelo Departamento de pesquisas da IBM
 - Interface para o sistema de banco de dados relacional System R
 - Inicialmente chamava-se SEQUEL (Structured English QUERy Language)
 - A partir de 1977, passou a ser chamada de SQL
 - Query = consulta, em inglês

SQL – Structured Query Language

- Em 1986, o Instituto Nacional Americano de Padrões (ANSI) juntamente com a ISO (International Standards Organization) publicam padrão de linguagem SQL-86 ou SQL-1
 - Linguagem padrão adotada para Bancos de Dados Relacionais
 - Os vários SGBDs relacionais passaram a utilizar SQL
- A SQL-86 passou por revisões:
 - SQL-92 ou SQL-2, em 1992
 - SQL-99 ou SQL-3, em 1999
 - SQL-2003, em 2003

SQL – Categorias de instruções

- DDL (relacionado à linguagem SQL)
 - Data Definition Language – grupo de instruções do SQL para criar tabelas, alterar a estrutura das tabelas ou eliminar tabelas.
 - Instruções CREATE, ALTER, DROP
- DML (relacionado à linguagem SQL)
 - Data Manipulation Language – grupo de instruções do SQL para criar manipular as tabelas, ou seja, para inserir dados, atualizar os dados, excluir dados, consultar dados
 - Instruções INSERT, UPDATE, DELETE, SELECT

Comandos SQL – MySQL e SQL Server

- No MySQL Server, todos os comandos devem ser finalizados por ponto e vírgula (;)
- No SQL Server, da Microsoft, não é obrigatório o ponto e vírgula no final do comando

Criando um banco de dados (Schema)

- Comando:
`CREATE DATABASE nome-do-banco;`
- No MySQL Workbench, os bancos de dados aparecem na parte esquerda da tela, na janela Schemas
- As tabelas são criadas dentro de um banco de dados. Para isso, devemos selecionar o banco de dados que queremos utilizar

Selecionando um banco de dados (Schema)

- Comando:
`USE nome-do-banco;`
- No MySQL Workbench, é possível selecionar o banco apenas dando um duplo clique com o mouse em cima do nome do banco na janela SCHEMAS

Criando uma tabela

- Comando:

```
CREATE TABLE nome-da-tabela (  
    nome-campo1 tipo-campo1,  
    nome-campo2 tipo-campo2,  
    .....  
    nome-campoN tipo-campoN  
);
```

- A tabela será criada dentro do banco de dados selecionado, com os campos definidos no comando.
- No comando acima, é possível acrescentar restrições aos campos, como PRIMARY KEY, etc.

Criando uma tabela (exemplo)

- Exemplo feito em aula:

```
CREATE TABLE Aluno (  
    ra INT PRIMARY KEY,  
    nome VARCHAR(40),  
    bairro VARCHAR(40)  
);
```

- O comando acima criará a tabela Aluno, com 3 campos: ra, nome e bairro.
- O campo ra terá um valor numérico e inteiro e esse campo será a chave primária da tabela.
- Os valores dos campos nome e bairro serão caracteres.

Campos com valores caracteres

- Campo com valores caracteres como o nome e o bairro podem ser definidos com o tipo CHAR ou VARCHAR
- Diferença entre CHAR e VARCHAR:
 - Quando se define que o nome é CHAR(10), então todos os campos desse tipo terão 10 caracteres, mesmo que o nome inserido tenha menos do que 10 caracteres
 - Ex: 'Bruno' será armazenado como 'Bruno '
 - O campo nome terá 5 espaços em branco para completar 10 caracteres
 - Quando se define que o nome é VARCHAR(10), então todos os campos terão no máximo 10 caracteres. Se o nome inserido tiver menos do que 10 caracteres, o campo conterá apenas os caracteres inseridos
 - Ex: 'Bruno' será armazenado como 'Bruno'

Visualizando ou listando os dados da tabela

- Comando:

`SELECT * FROM nome-da-tabela;`



* significa "todas as colunas"

- O comando acima exibe todos os dados de uma tabela
- Quando a tabela acabou de ser criada e ainda não tem dados, o comando exibirá apenas os títulos das colunas

Inserindo dados na tabela

- Comando:
`INSERT INTO` nome-da-tabela
`VALUES` (dado-campo1, dado-campo2,..... dado-campoN);
- O comando acima vai inserir os dados dentro dos parênteses na tabela, preenchendo um novo registro (nova “linha”) na tabela
- A ordem que os dados devem aparecer dentro do parênteses deve corresponder à ordem da criação dos campos no comando CREATE TABLE

Inserindo dados na tabela (exemplo)

- Exemplo:

`INSERT INTO` Aluno

`VALUES` (51000, 'Maria' , 'Paraíso');

- O comando acima vai inserir os dados correspondentes a um aluno, de ra= 51000, nome = 'Maria', bairro = 'Paraíso'
- Repare que para o valor int, não é preciso aspas
- E que para o valor do tipo varchar, é preciso aspas
 - Tanto aspas simples quanto aspas duplas são aceitas, mas `acostume-se a utilizar aspas simples`

Inserindo dados de mais de uma linha na tabela

- Exemplo:

INSERT INTO Aluno

VALUES (51001, 'José' , 'Tatuapé'), (51002, 'Claudio', 'Cambuci'),
(51003, 'Ana', 'Saúde'), (51004, 'Marina', 'Jabaquara') ;

- Pode-se inserir de uma só vez os dados de vários alunos, como no exemplo acima (inserção de 4 alunos)
- É preciso tomar cuidado para colocar entre parênteses os dados de cada aluno, na ordem em que foi criada a tabela Aluno, separados por vírgula.
- Cada conjunto de parênteses deve ser separado por vírgula.

Visualizando os dados apenas de algumas colunas

- Exemplo 1:

`SELECT` nome `FROM` Aluno;

O comando acima exibe apenas a coluna nome da tabela Aluno.

- Exemplo 2:

`SELECT` nome, bairro `FROM` Aluno;

O comando acima exibe apenas as colunas nome e bairro da tabela Aluno.

- Exemplo 3:

`SELECT` bairro, ra `FROM` Aluno;

O comando acima exibe apenas as colunas bairro e ra da tabela Aluno, nessa ordem.

Visualizando os dados apenas de algumas linhas

- Comando:

`SELECT * FROM nome-da-tabela WHERE condição;`



pode ser *
ou o(s) nome(s)
da(s) coluna(s)
desejada(s)



utiliza-se o **WHERE**
para apresentar a condição
para “filtrar” as linhas desejadas

- As linhas que satisfazem a condição (colocada após o WHERE) são exibidas pelo comando.

Visualizando os dados apenas de algumas linhas

- Exemplo 1:

`SELECT * FROM Aluno WHERE ra = 51002;`

O comando acima exibe os dados do aluno de RA 51002

- Exemplo 2:

`SELECT * FROM Aluno WHERE ra >= 51002;`

O comando acima exibe os dados dos alunos de RA >= 51002

- Exemplo 3:

`SELECT * FROM Aluno WHERE ra <> 51002;`

O comando acima exibe os dados dos alunos de RA diferente de 51002

Obs.: o MySQL e o SQL Server aceitam também != como sinal de "diferente", mas o padrão é <>

Visualizando os dados apenas de algumas linhas

- Exemplo 4:

`SELECT * FROM Aluno WHERE nome LIKE 'M%';`

O comando acima exibe os dados dos alunos cujo nome começa com M.

Quando se especifica um padrão como 'M%', utiliza-se o LIKE. O sinal de % representa zero ou mais caracteres.

Dessa forma, esse comando procurará os nomes que tenham a primeira letra M e depois pode vir uma quantidade qualquer de caracteres, e não importa quais caracteres.

Visualizando os dados apenas de algumas linhas

- Exemplo 5:

```
SELECT * FROM Aluno WHERE nome LIKE '%a';
```

O comando acima exibe os dados dos alunos cujo nome começa termina com a.

O sinal de % representa zero ou mais caracteres.

Dessa forma, esse comando procurará os nomes que tenham uma quantidade qualquer de caracteres, e não importa quais caracteres, desde que no final tenha a letra a

Visualizando os dados apenas de algumas linhas

- Exemplo 6:

```
SELECT * FROM Aluno WHERE nome LIKE '_n%';
```

O comando acima exibe os dados dos alunos cujo nome tenha a letra n como segunda letra.

O sinal de _ representa apenas um caractere.

Dessa forma, esse comando procurará os nomes que tenham um caractere qualquer, seguido da letra n e depois do n pode vir uma quantidade qualquer de caracteres, e não importa quais caracteres

Alterando o valor de algum dado já inserido

- Pode-se alterar o valor de mais de uma coluna numa mesma linha em um só comando.
- Exemplo:

```
UPDATE Aluno SET nome = 'Ana Maria', bairro = 'Tatuapé'  
WHERE ra = 51003;
```

O comando acima altera o valor do nome e do bairro do aluno de RA 51003.

- **ATENÇÃO:** Se não for colocado a cláusula WHERE, todas as linhas da tabela serão afetadas por este comando.

(o MySQL tem uma proteção contra isso, mas ela pode ser desabilitada, e outros SGBDs não têm essa proteção)

Excluindo uma ou mais linhas da tabela

- Comando:

DELETE FROM nome-da-tabela **WHERE** condição;

- As linhas que satisfazem a condição (colocada após o WHERE) serão excluídas da tabela
- Exemplo: **DELETE FROM** Aluno **WHERE** ra = 51003;

O comando acima exclui da tabela a linha referente ao aluno de RA 51003.

- **ATENÇÃO:** Se não for colocado a cláusula WHERE, todas as linhas da tabela serão afetadas por este comando.

(o MySQL tem uma proteção contra isso, mas ela pode ser desabilitada, e outros SGBDs não têm essa proteção)

MySQL – Comandos UPDATE e DELETE

- O MySQL tem uma proteção que não permite que o UPDATE e o DELETE FROM sejam executados sem que se coloque a cláusula WHERE.
- O MySQL também obriga que se utilize na condição do WHERE a coluna que é a chave primária da tabela. Isso vale para os comandos UPDATE e DELETE FROM
- Isso já não ocorre no SQL Server (que será usado no Azure)
- No MySQL, essa proteção pode ser desabilitada, caso o usuário queira.
- Por isso, é bom tomar cuidado em não esquecer de colocar WHERE nos comandos UPDATE e DELETE FROM.

Alterando o valor de algum dado já inserido

- Pode-se alterar o valor de mais de uma coluna numa mesma linha em um só comando.
- Exemplo:

```
UPDATE Aluno SET nome = 'Ana Maria', bairro = 'Tatuapé'  
WHERE ra = 51003;
```

O comando acima altera o valor do nome e do bairro do aluno de RA 51003.

- **ATENÇÃO:**

Se não for colocado a cláusula WHERE, todas as linhas da tabela serão afetadas por este comando.

(No MySQL, há uma proteção que não deixa isso acontecer

Alter table

Acrescentar a coluna:

```
alter table <tabela> add <campo> tipo;
```

```
alter table <tabela> add column <campo> tipo;
```

Alterar o tamanho da coluna:

```
alter table <tabela> modify <campo> Tipo ;
```

Mostrar a estrutura da tabela:

```
desc <tabela>;
```

```
describe <tabela>
```

Alter Table

Excluir a coluna da tabela:

```
alter table <tabela> drop <campo>;
```

```
alter table <tabela> drop column <campo>;
```

Exibir os dados da tabela ordenados crescente:

```
select * from <tabela> order by <campo>
```

Exibir os dados da tabela ordenados decrescente:

```
select * from <tabela> order by <campo1> desc, <campo n> desc;
```