Linguagem SQL

A **linguagem SQL – Structure Query Language**, é usada em banco de dados relacionais e envolve as operações de definição do banco de dados como manipulação dos dados em si.

Desenvolvida nos anos 70 e revisada várias vezes depois, o SQL tornou-se a linguagem padrão no mercado de banco de dados. Embora o SQL tenha se tornado um padrão para os bancos de dados, ele sofreu alterações específicas e dependendo do banco poderá ter recursos específicos

- A linguagem SQL é dividida em subconjuntos de acordo com as operações que queremos efetuar sobre um banco de dados.

- Linguagem de definição de dados (ou DDL, de Data Definition Language) é um conjunto de comandos dentro da SQL usada para a definição das estruturas de dados. Entre os principais comandos DDL estão CREATE (Criar), DROP (deletar) e ALTER (alterar).

- Linguagem de manipulação de dados (ou DML, de Data Manipulation Language) é o grupo de comandos dentro da linguagem SQL utilizado para a recuperação, inclusão, exclusão e alteração de informações em bancos de dados. Os principais comandos DML são SELECT (Seleção de Dados), INSERT (Inserção de Dados), UPDATE (Atualização de Dados) e DELETE (Exclusão de Dados).

- Linguagem de controle de dados (ou DCL, de Data Control Language) é o grupo de comandos que permitem ao administrador de banco de dados controlar o acesso aos dados deste banco.

INTRODUÇÃO COMANDOS INICIAIS SQL – RESUMO

Criação de Banco de dados:

Sintaxe:
CREATE DATABASE nome_bancodedados

Ex.: Criar um banco de dados chamado Desktop
Create Database Desktop;

Exclusão de Banco de Dados:

Sintaxe: DROP DATABASE nome_bancodedados

Ex.: Excluir um banco de dados chamado Desktop
Drop Database Desktop;

Conexão Banco de Dados

Antes que pensemos em inserir tabelas (e registros), precisamos informar qual bando de dados (database) iremos utilizar:

Sintaxe:
Use nome_bancodedados ;

Ex.: Estabelecer conexão com o banco de dados chamado Desktop

```
Use Desktop;
```

Comentários nos scripts

É importante comentar os scripts. Pode-se utilizar as formas:

Sintaxe:
-- comenta o restante da linha
/* inicia o bloco, podendo incluir várias linhas

*/ Finaliza o bloco

Criação de tabelas

Para criar as tabelas e as respectivas colunas, é necessário conhecer os tipos de dados que podem ser utilizados**(verificar no final do documento).**

Sintaxe:

```
Criação de tabela Simples:
 CREATE TABLE  nome_da_tabela
(nome_do_campo tipo_de_dados** tamanho do campo especificações*,
nome_do_campo tipo_de_dados** tamanho do campo especificações*,
nome_do_campo tipo_de_dados** tamanho do campo especificações*,
nome_do_campo tipo_de_dados**tamanho do campo)


Criação de tabela com Chaves Estrangeiras:

 CREATE TABLE  nome_da_tabela
(nome_do_campo tipo_de_dados** tamanho do campo especificações*,
nome_do_campo tipo_de_dados** tamanho do campo especificações*,
nome_do_campo tipo_de_dados** tamanho do campo especificações*,
nome_do_campo tipo_de_dados**tamanho do campo,
Chave_Estrangeira (nome_da_coluna_desta_tabela) referencia
      Nome_da_outra_tabela(Nome_da_coluna))
```

*Especificações mais utilizadas:
Not null – campo não nullo
primary key – campo chave primária
auto_increment – campo autonumeração
unique – campo comvalor único.

** Tipos de Dados mais utilizados
Dados numéricos

INT — número inteiro de tamanho comum;
DECIMAL — número decimal, de ponto fixo;
BIT — um campo de um bit.

Tipos de dados em strings

CHAR — uma cadeia de caracteres (string), de tamanho fixo e não-binária;
VARCHAR — uma string de tamanho variável e não-binária;
TEXT — uma string não-binária e pequena;

Tipos de dados de Valores Temporais

DATE — o valor referente a uma data no formato 'CCYY-MM-DD'. Por exemplo 1985-11-25 (ano-mês-dia). O 'CC' se refere aos dois dígitos do século (Century, em inglês);
TIME — um valor horário no formato 'hh:mm:ss' (hora:minutos:segundos);
DATETIME : data e hora juntos.

Exemplo 1: Criar tabela Produto com os campos, Código(tipo inteiro, não nulo, chave primária e autonumeração), nome, preço e quantidade)

```
 CREATE TABLE Produto
(Codigo int not null primary key auto_increment,
Nome varchar(50),
Preço decimal(10,2),
quantidade int)
```

Exemplo 2: Utilizando chaves estrangeiras

Criar tabela Categoria

Código da categoria(tipo inteiro, não nulo, chave primária e autonumeração) e nome

Criar tabela Produto

Código do produto (tipo inteiro, não nulo, chave primária e autonumeração), nome, preço, quantidade e categoria*(referência a categoria))

```
 CREATE TABLE Categoria
(CodigoCategoria int not null primary key auto_increment,
Nome varchar(50));
```

```
 CREATE TABLE Produto
(Codigo int not null primary key auto_increment,
Nome varchar(50),
Preço decimal(10,2),
quantidade int,
codigocategoria int,
Foreign key (codigocategoria) references Categoria(codigocategoria)
);
```

Tipos de dados numéricos no MySQL

Resumo dos tipos de dados possíveis no MySQL:

TINYINT — número inteiro muito pequeno (tiny);

SMALLINT — número inteiro pequeno;

MEDIUMINT — número inteiro de tamanho médio;

INT — número inteiro de tamanho comum;

BIGINT — número inteiro de tamanho grande;

DECIMAL — número decimal, de ponto fixo;

FLOAT — número de ponto flutuante de precisão simples (32 bits);

DOUBLE — número de ponto flutuante de precisão dupla (64 bits);

BIT — um campo de um bit.

Tipos de dados em strings

Strings são cadeias de caracteres. No MySQL, uma string pode ter qualquer conteúdo, desde texto simples a dados binários – tais como imagens e arquivos. Cadeias de caracteres podem ser comparadas e ser objeto de buscas.

CHAR — uma cadeia de caracteres (string), de tamanho fixo e não-binária;

VARCHAR — uma string de tamanho variável e não-binária;

BINARY — uma string binária de tamanho fixo;

VARBINARY — uma string binária de tamanho variável;

BLOB — um BLOB (Binary Large OBject – OBjeto Grande Binário) pequeno;

TINYBLOB — um BLOB muito pequeno;

MEDIUMBLOB — um BLOB de tamanho médio;

LONGBLOB — um BLOB grande;

TINYTEXT — uma string não-binária e de tamanho bem reduzido;

TEXT — uma string não-binária e pequena;

MEDIUMTEXT — uma string de tamanho comum e não-binária;

LONGTEXT — uma string não-binária de tamanho grande;

Armazenamento de data e hora

Há várias opções para armazenar dados relacionados a data e hora. Se você quiser apenas armazenar o ano referente a um evento, pode usar o tipo YEAR. O tipo TIMESTAMP pode ser usado para acompanhar as mudanças ocorridas em um campo de uma tabela:

DATE — o valor referente a uma data no formato 'CCYY-MM-DD'. Por exemplo 1985-11-25 (ano-mês-dia). O 'CC' se refere aos dois dígitos do século (Century, em inglês);

TIME — um valor horário no formato 'hh:mm:ss' (hora:minutos:segundos);

TIMESTAMP — timestamp é uma sequência de caracteres ou informação codificada que identifica uma marca temporal ou um dado momento em que um evento ocorreu. No MySQL, ele tem o formato 'CCYY-MM-DD hh:mm:ss' – neste caso, seguem a padronização ISO 8601;

YEAR — armazena um ano no formato 'CCYY' ou 'YY';

Dados espaciais

Observações:

O atributo AUTO_INCREMENT é bastante utilizado quando se precisa criar um campo na tabela para ser chave primária quando não temos muitas alternativas em outros atributos para que seja gerado automaticamente um identificador único para cada registro que será cadastrado na tabela. Para que um campo AUTO_INCREMENT seja criado sem problemas de compilação, a coluna criada deverá seguir determinadas diretrizes, tais como:

Obrigatoriamente o tipo de dado deve ser numérico e inteiro, ou seja, a coluna pode ter tipos SMALLINT, MEDIUMINT, INT ou BIGINT;

A coluna deve ser indexada (INDEX, UNIQUE INDEX ou PRIMARY KEY) e ter a restrição NOT NULL;

Uma tabela somente poderá ter uma única coluna com este atributo;

Quando atribuímos de forma explícita um valor, caso esse valor não seja repetido, o SGBD cadastra na coluna aquele que foi enviado e continua a seqüência a partir deste. Para deixarmos que o SGBD tome conta da seqüência dos números dessa coluna, podemos omitir a coluna da lista de colunas no comando INSERT.

O MySQL 5.0 ainda suporta os tipos de dados de ponto-flutuante,FLOAT e DOUBLE que requerem quatro e oito bytes respectivamente. Apesar de serem mais eficientes em performace na recuperação por utilizarem um tipo numérico nativo do processamento computacional, estes podem apresentar erros de arredondamento. Não seria uma boa idéia utilizar tais tipos de dados para armazenar dados financeiros, como preços e valores de serviços em notação monetária.

Para isto, o MySQL 5.0 disponibiliza o tipo de ponto-fixo DECIMAL, que fornece mais segurança quanto ao armazenamento de preços por exemplo por não apresentar problemas com arredondamento, mas por outro lado são mais lentos na recuperação e cálculos. NUMERIC é o mesmo que DECIMAL e funciona da mesma maneira no MySQL 5.0.

OBS.: Já que para valores monetários, é melhor utilizarmos o tipo de dado DECIMAL, podemos definir os campos como DECIMAL(10,2), o que nos dará 10 digitos de precisão – posições antes da vírgula – e 2 posições de escala – posições após a vírgula. Para inserirmos dados num campo decimal, a vígula deverá ser substituída pelo ponto (‘. ‘).

Valores possíveis para cada Tipo de Campo

Tipos Numéricos				
Tipo	**Uso**		**Tamanho**	
		Atributo	**MIN**	**MAX**
TINYINT	Um inteiro muito pequeno	Signed:	-128	127
		Unsigned	0	255
SMALLINT	Um inteiro pequeno	Signed:	–32768	32767
		Unsigned	0	65535
MEDIUMINT	Um inteiro de tamanho mediano	Signed:	–8388608	8388607
		Unsigned	0	16777215
INT or INTEGER	Um inteiro de tamanho normal	Signed:	–2147483648	2147483647
		Unsigned	0	4294967295
BIGINT	Um inteiro de temanho grande	Signed:	–9223372036854775808	9223372036854775807
		Unsigned	0	18446744073709551615
FLOAT	Um pequeno número de	Signed	–3.402823466E+38	–1.175494351E-38, 0
			1.175494351E-38	3.402823466E+38

	ponto flutuante (precisão simples)	**Não pode ser unsigned**	-	
		OBS	Se o número de decimais não for especificado ou for <= 24 será de precisão simples	
DOUBLE, DOUBLE PRECISION, REAL	Um número de ponto flutuante de tamanho normal (precisão dupla)	Signed	-1.7976931348623157E+308	-2.2250738585072014E-308, 0
			2.2250738585072014E-308	1.7976931348623157E+308
		Não pode ser unsigned	-	
		OBS	Se o número de decimais não for especificado ou for 25 <= Decimals <= 53 será de precisão dupla	
DECIMAL, NUMERIC	Um número de ponto flutuante descompactado.	Signed	Se comporta como um campo CHAR: “descompactado” significa que o número é armazenado como uma string, usando um caractere para cada dígito do valor. O ponto decimal e, para números negativos, o sinal ‘-’ não é contado. Se o decimal for 0, os valores não terão ponto decimal ou parte fracionária.	O alcance máximo de valores decimais é o mesmo que para o DOUBLE, mas a faixa atual para um campo DECIMAL dado pode ser limitado pela escolha de comprimento e decimais.
		Não pode ser unsigned	-	
		OBS	Se Decimais é deixado de fora ele é definido como 0. Se o comprimento é deixado de fora ele é definido como 10. Note que no MySQL 3,22 o comprimento inclui o sinal eo ponto decimal	
Campos de Datas				
		Formato	**MIN**	**MAX**
DATE	Data		‘1000-01-01’	‘9999-12-31’
		OBS	Formato: ‘YYYY-MM-DD’	
DATETIME	Data e horário		‘1000-01-01 00:00:00’	‘9999-12-31 23:59:59’
		OBS	Formato: ‘YYYY-MM-DD HH:MM:SS’	
TIMESTAMP	Timestamp		‘1970-01-01 00:00:00’	aproximadamente 2037

		OBS	Formato: YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD ou YYMMDD, dependendo se M é 14 (ausente), 12, 8 ou 6, podendo ser strings ou números. **Este tipo é recomendável para instruções de INSERT ou UPDATE pois é automaticamente marcado com os valores da operação mais recente quando não informado.**	
TIME	A time		‘-838:59:59’	‘838:59:59’
		OBS	formato: ‘HH:MM:SS’, podem ser strings ou números	
YEAR	Anos com 2 ou 4 digitos. O padrão é 4 digitos	4 digitos	1901	2155 e 0000
		2 digitos	1970	2069
		OBS	Formato: YYYY podem ser strings ou números.	
Campos Texto				
			MIN	**MAX**
CHAR	String de tamanho fixo. Sempre é completada com espaços a direita até o tamanho definido		1	255 caracteres
		OBS	Espaços excessivos são removidos quando o valor é trazido.Os valores são ordenados e comparados ignorando caixas altas e baixas de acordo com a codificação padrão, a menos que seja fornecido uma chave binária.	
VARCHAR	String de tamanho variável		1	255 caracteres
		OBS	Os valores são ordenados e comparados ignorando caixas altas e baixas de acordo com a codificação padrão, a menos que seja fornecido uma chave binária.Nota: Espaços execessivos são removidos quando o valor é inserido.	
TINYTEXT			0	255 (2^8 – 1) caracteres
TEXT			0	65535 (2^16 – 1) caracteres
MEDIUMTEXT			0	16777215 (2^24 – 1) caracteres
LONGTEXT			0	4294967295 (2^32 – 1) caracteres
Dados Binários				
TINYBLOB			0	255 (2^8 – 1) caracteres
BLOB			0	65535 (2^16 – 1)

				caracteres
MEDIUMBLOB			0	16777215 (2^24 – 1) caracteres
LONGBLOB			0	4294967295 (2^32 – 1) caracteres
Listas				
			MIN	**MAX**
ENUM	Enumeração		String que pode conter apenas um valor ou zero	65535 valores distintos.
SET	Lista		String que pode conter zero ou mais valores	64 itens
fonte: http://help.scibit.com/Mascon/masconMySQL_Field_Types.html e http://dev.mysql.com/doc/refman/5.0/en/data-type-overview.html				