SQL – Structure Query Language

Embora a Linguagem de Consulta Estruturada (SQL) tenha sido desenvolvida pela IBM, o SQL está atualmente sob a regulamentação do American National Standarts Institute (ANSI). Como com todos os "padrões", existem variações do SQL - "aprimoramentos proprietários," para serem usados com o Microsoft SQL Server, Oracle e outros. Portanto, conhecer o conjunto de comandos SQL padrão permitirá a você o conhecimento fundamental sobre SQL que você poderá utilizar com múltiplos sistemas de Banco de dados. Aqui estão alguns comandos padrão:
O profissional responsável na criação e manipulação e segurança do Banco de Dados é denominado **Desenvolvedor DBA**.

Principais grupos de comandos **DDL** e **DML**:

- DDL (Data Definition Language):
  - Principais comandos: **create, alter e drop**

- DML (Data Manipulation Language):
  - Principais comandos: **select, insert, update e delete**

. CREATE - Cria uma nova tabela;
. ALTER - Modifica a definição (estrutura, tipo de dados e outros) de uma tabela existente;
. DROP - Remove definitivamente elementos como campos e tabelas;

. INSERT - **Adiciona** um registro(**tupla**) a uma tabela(**entidade**);
. UPDATE - **Modifica** dados em um registro existente;
. SELECT - Executa uma **consulta** em uma tabela, incluindo funções matemáticas, comparação de campo, combinação de padrão e outros;
. DELETE - **Remove** definitivamente os elementos de uma tabela.

. CREATE

O comando CREATE cria uma tabela em um Banco de dados selecionado. A sintaxe básica é a seguinte:

CREATE TABLE [nome da tabela] [(nome_do_campo tipo_de_dado, ...] [opções]
Nota: Alguns tipos de dados comuns são **numeric**,**int**, double, **char**, **varchar**, **date**, time, datetime, text e blob. Veja a documentação do seu Banco de dados para obter uma lista de tipos de dados suportados e quaisquer restrições de tamanho e dados.

Para criar a tabela INFO_AMIGOS, usada na coluna passada, a declaração SQL poderia ser semelhante a essa:
(*) Para criar um BANCO DE DADOS utiliza-se o comando abaixo:
Create database [nome do banco];
p.ex: create database escola;

```
CREATE TABLE INFO_AMIGOS (
ID_AMIGO char(5) not null,
NOME_COMPLETO varchar(100),
DATA_ANIVERSARIO date,
COR_FAVORITA varchar(25),
UNIQUE ID_AMIGO (ID_AMIGO)
);
```

Xamp....phpmyadmin
Este exemplo de código usa vários espaços em branco. Espaços em branco são irrelevantes em declarações SQL, mas certamente fazem seu código ficar mais fácil de ler (e mais fácil de editar depois!). A declaração anterior também poderia ser escrita assim:

```
CREATE TABLE INFO_AMIGOS (ID_AMIGO char(5) not null, NOME_COMPLETO varchar(100), DATA_ANIVERSARIO date, COR_FAVORITA varchar(25), UNIQUE ID_AMIGO (ID_AMIGO));
```

O comando CREATE define os atributos dos campos em sua tabela; ele fornece cada tipo e tamanho. Neste exemplo, o campo ID_AMIGO é definido como "not null", indicando que quando um registro for adicionado, se o campo estiver vazio, estará simplesmente vazio, ou terá o valor default como "0" para campos numéricos. Um valor de NULL indica algo completamente diferente de "vazio", e é uma distinção muito importante a lembrar. O campo ID_AMIGO também é identificado como único, adicionando UNIQUE ID_AMIGO (ID_AMIGO) à declaração SQL, indicando que ID_AMIGO possui sempre dados únicos: Embora possa haver múltiplas entradas para João Roberto, haverá apenas um ID_AMIGO 1,2 e assim por diante.

. ALTER

O comando ALTER lhe dá a oportunidade de modificar elementos em uma tabela em particular. Por exemplo, você pode usar ALTER para adicionar campos ou mudar tipos de campos. Por exemplo, se você deseja mudar o tamanho do campo NOME_COMPLETO da tabela INFO_AMIGOS de 100 para 150, pode usar este código:

```
ALTER TABLE INFO_AMIGOS
CHANGE NOME_COMPLETO
NOME_COMPLETO VARCHAR (150);
```

Para adicionar uma coluna a uma tabela, você pode utilizar a seguinte declaração, que adiciona um campo chamado TAMANHO_CHAPEU:

```
ALTER TABLE INFO_AMIGOS
ADD TAMANHO_CHAPEU char(5);
```

Utilizar o comando ALTER alivia a necessidade de deletar toda uma tabela e recriá-la, somente porque você especificou um nome de campo incorretamente ou cometeu outros erros menores.

. DROP

O comando DROP é um pouco perigoso se você não estiver prestando atenção no que está fazendo, porque irá deletar toda a sua tabela em um piscar de olhos. A sintaxe é muito simples:

DROP TABLE [nome da tabela];

Se você deseja deletar a tabela INFO_AMIGOS e confiar em sua memória para lembrar o aniversário dos seus amigos, o comando pode ser:

DROP TABLE INFO_AMIGOS**;**

Você pode utilizar o comando DROP juntamente com o comando ALTER para deletar campos específicos (e todos os dados contidos neles). Para deletar o campo TAMANHO_CHAPEU da tabela INFO_AMIGOS, use esta declaração:

ALTER TABLE INFO_AMIGOS
DROP TAMANHO_CHAPEU;

O campo TAMANHO_CHAPEU não existirá mais, no entanto todos os outros campos da sua tabela permanecerão intactos.

. **INSERT**

Você usa o comando INSERT para popular (ou inserir novas tuplas ou registros) em suas tabelas com um registro por vez. A sintaxe básica para o comando INSERT é:

INSERT INTO [nome da tabela] ([nome do campo1], [nome do campo2], ...) **VALUES** ('[valor do campo 1]', '[valor do campo 2]', ...);

Para adicionar um registro(tupla) à tabela INFO_AMIGOS, onde os campos são ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO e COR_FAVORITA, o comando seria:

Ex: **INSERT INTO** INFO_AMIGOS **(**ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA**)** **VALUES** **(**'1', 'João Roberto', '03-10-1970', 'Azul'**);**

Para adicionar seus outros amigos, utilize declarações INSERT adicionais:

INSERT INTO INFO_AMIGOS (ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA) **VALUES** ('2', 'Elphidium', '15-10-1978', 'Preto');

INSERT INTO INFO_AMIGOS (ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA) **VALUES** ('3', 'Leonardo', '15-10-1996', 'Azul');

INSERT INTO INFO_AMIGOS (ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA) **VALUES** ('4', 'Márcio', '15-10-1992', 'Amarelo');

INSERT INTO INFO_AMIGOS (ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA) **VALUES** ('5', 'Maria', '15-10-1991', 'Amarelo');

INSERT INTO INFO_AMIGOS (ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA) VALUES ('3', 'Vladisauskis', '02-10-1982', 'Verde');

INSERT INTO INFO_AMIGOS (ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA) VALUES ('4', 'João Roberto', '03-11-1970', 'Azul');

Ao inserir registros, certifique-se de separar suas strings com aspas simples ou aspas duplas. Se você utilizar aspas simples em torno de suas strings e dados que você for adicionar contiverem apóstrofes, evite erros, anulando o apóstrofe com (\') dentro da declaração INSERT. Igualmente, se utilizar aspas duplas em torno das strings e desejar incluir aspas duplas como parte dos dados, anule-as (\") dentro da declaração INSERT.

Por exemplo, se você usa aspas simples em torno de suas strings e deseja inserir um registro para um amigo chamado Mark O'Hara, pode usar a seguinte declaração:

INSERT INTO INFO_AMIGOS (ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO, COR_FAVORITA) VALUES ('5', "Mark O'Hara", "12-12-1968", "Laranja");

. UPDATE

O comando UPDATE modifica parte de um registro sem substituir todo o registro. Aqui está a sintaxe básica do comando UPDATE:

UPDATE [nome da tabela]
SET [nome do campo] = '[novo valor]'
WHERE [alguma expressão];

Por exemplo, suponhamos que você tenha uma data de aniversário incorreta na tabela INFO_AMIGOS: João Roberto, com um ID_AMIGO: 1, nasceu em 3 de novembro e não 3 de outubro. Em vez de deletar o registro e inserir um novo, basta usar o comando UPDATE para mudar os dados no campo DATA_ANIVERSARIO:

```
UPDATE INFO_AMIGOS
SET DATA_ANIVERSARIO = '03-11-1970'
WHERE ID_AMIGO = '1';
```

Se você utilizar um comando UPDATE sem especificar uma expressão WHERE, estará atualizando todos os registros. Por exemplo, para mudar a cor favorita de todos para vermelho, use esta declaração UPDATE:

```
UPDATE INFO_AMIGOS
SET COR_FAVORITA = 'vermelho';
```

UPDATE pode ser considerado um comando muito poderoso do SQL. Por exemplo, você pode realizar funções de strings e funções matemáticas em registros existentes e utilizar o comando UPDATE para modificar seus valores.

. SELECT

Quando você estiver criando um Banco de dados direcionado a site da Web, o comando SELECT provavelmente será muito utilizado. Ele faz com que determinados registros em seu Banco de dados sejam escolhidos, baseados no critério que você definir. Aqui está a sintaxe básica para o comando SELECT:

```
SELECT [nome do campo]
FROM [nome da tabela]
WHERE [alguma expressão]
ORDER BY [nomes do campo];
```

Para selecionar todos os registros em uma tabela, como a tabela INFO_AMIGOS, use esta declaração:

```
SELECT * FROM INFO_AMIGOS;
```

Para selecionar somente os dados no campo NOME_COMPLETO da tabela INFO_AMIGOS, use isto:

```
SELECT NOME_COMPLETO
FROM INFO_AMIGOS;
```

Para selecionar todos os registros em uma tabela e retorná-los em uma ordem particular use a expressão para ORDER BY. Por exemplo, para ver ID_AMIGO, NOME_COMPLETO e DATA_ANIVERSARIO em cada registro da tabela INFO_AMIGOS, ordenando-os do amigo mais novo para o mais velho, use o seguinte:

```
SELECT ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO
FROM INFO_AMIGOS
ORDER BY DATA_ANIVERSARIO DESC;
```

DESC significa "descendente". Para visualizar o mais velho para mais novo, use ASC, de "ascendente":

```
SELECT ID_AMIGO, NOME_COMPLETO, DATA_ANIVERSARIO
FROM INFO_AMIGOS
ORDER BY DATA_ANIVERSARIO ASC;
```

Ao preparar cláusulas ORDER BY, a ordem default é ASC (ascendente).

Você também pode executar funções matemáticas e funções de string dentro de declarações SQL, usando desta forma SELECT. Por exemplo, para localizar rapidamente o número de amigos em sua tabela INFO_AMGISO, use:

```
SELECT COUNT(ID_AMIGO) FROM INFO_AMIGOS;
```

O resultado desta declaração é 5. Você pode também utilizar a função COUNT() em qualquer outro campo para contar o número destas entradas na tabela.

Eu poderia escrever volumes de dados em muitas variações de comandos SELECT, mas, felizmente, outros já o fizeram. Adicionalmente, se estiver usando o Banco de dados MySQL, é bom saber que o manual do MySQL contém uma bela referência ao SQL, não deixe de visitar http://www.mysql.org.

. DELETE

O comando DELETE não é tão engraçado quando o SELECT, mas é útil. Assim como o comando DROP, usar DELETE sem prestar atenção no que você está fazendo pode trazer conseqüências terríveis em um ambiente de produção. Quando você usa DROP em uma tabela ou usa o DELETE para deletar um registro, isto é sempre. A sintaxe básica do comando DELETE é a seguinte:

```
DELETE FROM [nome da tabela]
WHERE [alguma expressão];
```

Por exemplo, para deletar entradas para João Roberto da sua tabela INFO_AMIGOS, você pode usar:

```
DELETE FROM INFO_AMIGOS
WHERE NOME_COMPLETO = 'João Roberto';
```

Mas espere um minuto! Se você executar este comando, irá deletar "ambos" os registros para João Roberto. Então, use o identificador juntamente com o nome:

```
DELETE FROM INFO_AMIGOS
WHERE NOME_COMPLETO = 'João Roberto' AND DATA_ANIVERSARIO = '03-11-1970';
```

Ou, se você sabe da importância dos identificadores únicos, pode utilizar:

```
DELETE FROM INFO_AMIGOS
WHERE ID_AMIGO = '1';
```

Se você usar um comando DELETE sem especificar uma expressão WHERE, irá deletar todos os registros. Por exemplo, este simples código SQL deleta todos os registros na tabela INFO_AMIGOS:

```
DELETE FROM INFO_AMIGOS;
```

Então lembre-se: Se você não deseja deletar todos os registros, mas somente alguns campos de uma tabela, deve usar isto:

```
ALTER TABLE [nome da tabela]
DROP [nome do campo];
```

Faça sempre backup dos seus dados se for utilizar os comandos DELETE e DROP, caso haja alguma perda por engano.