MySQL suporta um certo números de tipos de campos que podem ser agrupaos em três categorias: tipos numéricos, tipos de data e hora, e tipos string (caracteres).

Os tipos de campos suportados pelo MySQL estão listados abaixo: As seguintes letras são usadas como código nas descrições:

- `M`

  Indica o tamanho máximo do display. O tamanho máximo oficial do display é 255.

- `D`

  Aplica aos tipos de ponto flutuante e indica o número de digitos após o ponto decimal. O maior valor possível é 30, mas não pode ser maior que `M`-2.

Colchetes (‘`[`’ and ‘`]`’) indicam partes de tipos específicos que são opicionais

Note que se você especificar `ZEROFILL` para um campo MySQL automaticamente irá adicionar o atributo `UNSIGNED` ao campo.

Aviso: você deve estar ciente de que quando fizer uma subtração entre valores inteiros, onde um deles é do tipo `UNSIGNED`, o resultado será sem sinal!

`TINYINT[(M)] [UNSIGNED] [ZEROFILL]`

Um inteiro muito pequeno. A faixa deste inteiro com sinal é de `-128` até `127`. A faixa sem sinal é de `0` até `255`.

- `BIT`, `BOOL`, `BOOLEAN`

  Estes são sinônimos para `TINYINT(1)`.

  O sinônimo `BOOLEAN` foi adicionado na versão 4.1.0.

  Um tipo boolean verdadeiro será introduzido de acordo com o SQL-99.

- `SMALLINT[(M)] [UNSIGNED] [ZEROFILL]`

  Um inteiro pequeno. A faixa do inteiro com sinal é de `-32768` até `32767`. A faixa sem sinal é de `0` a `65535`.

- `MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]`

Um inteiro de tamanho médio. A faica com sinal é de `-8388608` a `8388607`. A faixa sem sinal é de `0` to `16777215`.

- `INT[(M)] [UNSIGNED] [ZEROFILL]`

  Um inteiro de tamanho normal. A faixa com sinal é de `-2147483648` a `2147483647`. A faixa sem sinal é de `0` a `4294967295`.

- `INTEGER[(M)] [UNSIGNED] [ZEROFILL]`

  Este é um sinônimo para `INT`.

- `BIGINT[(M)] [UNSIGNED] [ZEROFILL]`

  Um inteiro grande. A faixa com sinal é de `-9223372036854775808` a `9223372036854775807`. A faixa sem sinal é de `0` a `18446744073709551615`.

  Existem algumas coisas sobre campos `BIGINT` sobre as quias você deve estar ciente:

  - Todas as operações aritiméticas são feitas usando valores `BIGINT` ou `DOUBLE` com sinal, não devemos utilizar inteiros sem sinal maiores que `9223372036854775807` (63 bits) exceto com funções ded bit! Se você fizer isto, alguns dos últimos digitos no resultado podem estar errados por causa de erros de arredondamento na conversão de `BIGINT` para `DOUBLE`.

    O MySQL 4.0 pode tratar `BIGINT` nos seguintes casos:

    - Usar inteiros para armazenar grandes valores sem sinais em uma coluna `BIGINT`.
    - Em `MIN(big_int_column)` e `MAX(big_int_column)`.
    - Quando usar operadores (`+`, `-`, `*`, etc.) onde ambos os operandos são inteiros.
  - Você pode armazenar valores inteiro exatos em um campo `BIGINT` aramzenando-os como string, como ocorre nestes casos não haverá nenhuma representação intermediaria dupla.
  - '`-`', '`+`', e '`*`' serão utilizados em cálculos aritiméticos `BIGINT` quando ambos os argumentos forem valores do tipo `INTEGER`! Isto significa que se você multilicar dois inteiros grandes (ou obter resultados de funções que retornam inteiros) você pode obter resultados inesperados quando o resultado for maior que `9223372036854775807`.

- `FLOAT(precisão) [UNSIGNED] [ZEROFILL]`

  Um número de ponto flutuante. Não pode ser sem sinal. `precisão` pode ser `<=24` para um número de ponto flutuante de precisão simples e entre 25 e 53 para um número de ponto flutuante de dupla-precisão. Estes tipos são como os tipos `FLOAT` e `DOUBLE` descritos logo abaixo. `FLOAT(X)` tem o mesma faixa que os tipos correspondentes `FLOAT` e `DOUBLE`, mas o tamanho do display e número de casas decimais é indefinido.

  Na versão 3.23 do MySQL, este é um verdadeiro valor de ponto flutuante. Em versões anteriores , `FLOAT(precisão)` sempre tem 2 casas decimais.

  Note que o uso de `FLOAT` pode trazer alguns problemas inesperados como nos cálculos já que em MySQL todos são feitos com dupla-precisão.

  Esta sintaxe é fornecida para comptibilidade com ODBC.

- `FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]`

  Um número de ponto flutuante pequeno (precisão simples). Os valores permitidos estão entre `-3.402823466E+38` e `-1.175494351E-38`, `0` e entre `1.175494351E-38` e `3.402823466E+38`. Se `UNSIGNED` for especificado, valores negativos não são permitidos O `M` é a largura do display e o `D` é o número de casas decimais. `FLOAT` sem um argumento ou `FLOAT(X)` onde `X` <=24 tende a um número de ponto flutuante de precisão simples.

- `DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]`

  Um número de ponto flutuante de tamanho normal (dupla-precisão). Valores permitidos estão entre `-1.7976931348623157E+308` e `-2.2250738585072014E-308`, `0` e entre `2.2250738585072014E-308` e `1.7976931348623157E+308`. Se `UNSIGNED` for especificado, valores negativos não são permitidos. O `M` é a largura do display e o `D` é número de casa decimais. `DOUBLE` sem argumento ou `FLOAT(X)` onde 25 <= X <= 53 são números de ponto flutuante de dupla-precisão.

- `DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL],REAL[(M,D)] [UNSIGNED] [ZEROFILL]`

  Estes são sinônimos para `DOUBLE`.

- `DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]`

  Um número de ponto flutuante não empacotado. Se comporta como um campo `CHAR`: ``não empacotado" significa que o número é armazenado como uma string,

usando um caracter para cada digito do valor. O ponto decimal e, para números negativos, o sinal de menos ('-'), não são contados em M (mas é reservado espaço para isto). Se D for 0, os valores não terão ponto decimal ou parte fracionária. A faixa máxima do valor DECIMAL é a mesma do DOUBLE, mas a faixa atual para um campo DECIMAL dado pode ser limitado pela escolha de M e D. Se UNSIGNED é especificado, valores negativos não são permitidos.

Se D não for definido será considerado como 0. Se M não for definido é considerado como 10.

Note que antes da versão 3.23 do MySQL o argumento M deve incluir o espaço necessário para o sinal é o ponto decimal.

- `DEC[(M[,D])] [UNSIGNED] [ZEROFILL], NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL], FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]`

  Este é um sinônimo para DECIMAL.

  O alias FIXED foi adicionado na versão 4.1.0 para compatibilidade com outros servidores.

- `DATE`

  Uma data. A faixa suportada é entre '1000-01-01' e '9999-12-31'. MySQL mostra valores DATE no formato 'AAAA-MM-DD', mas permite a você a atribuir valores a campos DATE utilizando tanto strings quanto números. DATETIME

  Um combinação de hora e data. A faixa suportada é entre '1000-01-01 00:00:00' e '9999-12-31 23:59:59'. MySQL mostra valores DATETIME no formato 'AAAA-MM-DD HH:MM:SS', mas permite a você que atribuir valores a campos DATETIME utilizado strings ou números

- `TIMESTAMP[(M)]`

  Um timestamp. A faixa é entre '1970-01-01 00:00:00' e algum momento no ano 2037.

  No MySQL 4.0 ou anteriores, os valores TIMESTAMP são exibidos nos formatos YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, ou YYMMDD, dependendo se M é 14 (ou não definido), 12, 8 ou 6, mas permite a você atribuir valores ao campo TIMESTAMP usando strings ou números.

  Um campo TIMESTAMP é util para gravar a data e a hora em uma operação de INSERT or UPDATE porque é automaticamente definido a data e a hora da operação

mais recente se você próprio não especificar um valor. Você também pode definir a data e a hora atual atribuindo ao campo um valor `NULL`.

Desde o MySQL 4.1, `TIMESTAMP` é retornado com um string com o formato `'YYYY-MM-DD HH:MM:SS'`. Se você deseja tê-lo como um número você deve adcionar +0 a coluna timestamp. Teimestamp de tamanhos diferentes não são supoortados. Desde a versão 4.0.12, a opção `--new` pode ser usada para fazer o servidor se comportar como na versào 4.1.

Um `TIMESTAMP` sempre é armazenado em 4 bytes. O argumento `M` só afeta como a coluna `TIMESTAMP` é exibida.

Note que colunas do tipo `TIMESTAMP(M)` columns onde M é 8 ou 14 são apresentadas como números enquanto as outras colunas `TIMESTAMP(M)` são strings. Isto é apenas para assegurar que podemos eliminar e restaurar com segurança tabelas com estes tipos!

- `TIME`

  Uma hora. A faixa é entre `'-838:59:59'` e `'838:59:59'`. MySQL mostra valores `TIME` no formato `'HH:MM:SS'`, mas permite a você atribuir valores para as colunas `TIME` usando strings ou números.

- `YEAR[(2|4)]`

  Um ano no formato de 2 ou 4 digitos (padrão são 4 digitos). Os valores permitidos estão entre `1901` e `2155`, `0000` no formato de 4 digitos, e 1970-2069 se você estiver usando o formato de 2 digitos (70-69). MySQL mostra valores `YEAR` no formato `YYYY`, mas permie atribuir valores aos campos do tipo `YEAR` usando strings ou números. (O tipo `YEAR` é novo na versão 3.22 do MySL).

  `[NATIONAL] CHAR(M) [BINARY | ASCII | UNICODE]`

  Uma string de tamanho fixo que é sempre preenchida a direita com espaços até o tamanho especificado quando armazenado. A faixa de `M` é de 1 a 255 caracteres. Espaços extras são removidos quando o valor é recuperado. Valores `CHAR` são ordenados e comparados no modo caso insensitivo de acordo com o conjunto de caracteres padrão, a menos que a palavra chave `BINARY` seja utilizada.

  A partir da versão 4.1.0, se o valor `M` especificado é maio que 255, o tipo de coluna é convertido para `TEXT`. Este é um recurso de compatibilidade.

NATIONAL CHAR (ou em sua forma reduzida NCHAR) é o modo SQL-99 de definir que um campo CHAR deve usar o conjunto CHARACTER padrão. Este é o padrão no MySQL.

CHAR é uma simplificação para CHARACTER.

A partir da versão 4.1.0, o atributo ASCII pode ser especificado o que atribui o conjunto de caracteres latin1 a coluna CHAR.

A partir da versão 4.1.1, o atributo UNICODE pode ser especificado o que atribui o conjunto de caracteres ucs2 a coluna CHAR.

O MySQL lhe permite criar um campo do tipo CHAR(0).Isto é muito útil quando você precisa de comptibilidade com aplicativos antigos que dependem da existência de uma coluna, mas que, na verdade, não utiliza um valor. Isto também é muito bom quando você precisa de uma coluna que só pode receber 2 valores. Um CHAR(0), que não é definido como um NOT NULL, só irá ocupar um bit e pode assumir 2 valores: NULL or "".

- BIT, BOOL, CHAR

  This is a synonym for CHAR(1).

- [NATIONAL] VARCHAR(M) [BINARY]

  Uma string de tamanho variável. **NOTA:** Espaços extras são removidos quando o caracter é armazenado (o que difere da especificação ANSI SQL). A faixa de M é de 1 a 255 characters. Valores VARCHAR são ordenados e comparados no modo caso insensitivo a menos que a palavra chave BINARY seja utilizada.

  A partir da versão 4.1.0, se o valor M especificado é maio que 255, o tipo de coluna é convertido para TEXT. Este é um recurso de compatibilidade.

  VARCHAR é uma simplificação para CHARACTER VARYING.

- TINYBLOB, TINYTEXT

  Um campo BLOB ou TEXT com tamanho máximo de 255 (2^8 - 1) caracteres..

- BLOB, TEXT

  Um campo BLOB ou TEXT com tamanho máximo de 65535 (2^16 - 1) caracteres.

- `MEDIUMBLOB`, `MEDIUMTEXT`

  Um campo `BLOB` ou `TEXT` com tamanho máximo de 16777215 (2^24 - 1) caracteres.

- `LONGBLOB`, `LONGTEXT`

  Um campo `BLOB` ou `TEXT` com tamanho máximo de 4294967295 ou 4G (2^32 - 1) caracteres.

  Até a versão 3.23 o protocolo cliente/servidor e tabelas MyISAM tinham um limite de 16M por pacote de transmissão/registro de tabela, a partir da versão 4.x o tamanho máximo permitido das colunas `LONGTEXT` ou `LONGBLOB` depende do tamanho máximo configurado para o pacote no protocolo cliente/servidor e da memória disponível.

- `ENUM('valor1','valor2',...)`

  Uma enumeração. Um objeto string que só pode ter um valor, selecionado da lista de valores `'valor1'`, `'valor2'`, `...`, `NULL` ou valor especial de erro `""`. Um `ENUM` pode ter um máximo de 65535 valores diferentes.

- `SET('valor1','valor2',...)`

  Um conjunto. Um objeto string que pode ter zero ou mais valores, cada um deve ser selecionado da lista de valores `'valor1'`, `'valor2'`, `...`. Um `SET` pode ter até 64 membros.

Fonte:

http://dev.mysql.com/