Tipos de Dados SQL

Textuais

- Char [(n)] / Varchar (n)
- Long Varchar - Não impõe limites ao número de caracteres
- Text – igual a Long Varchar mas admite NULL

Booleano

- Bit / Boolean / Bool / Tinyint(1)
  - Valores: 0 e 1.

Datas e tempos

- Date
- Time
- Datetime e Timestamp
  (MS SQL Server: Datetime e Datetimeoffset)

➔https://dev.mysql.com/doc/refman/8.0/en/data-types.html

➔https://docs.microsoft.com/en-us/sql/t-sql/data-types

➔https://www.w3resource.com/sql/data-type.php

Numéricos

- Integer / Int
  - 4 bytes
  - -2,147,483,647 » + 2,147,483,647
- Smallint
  - 2 bytes.
  - -32,767 » +32,767 ou
    0 » 65535 (Smallint Unsigned)
- Tinyint
  - 1 byte.
  - 0 » 255.
- Decimal (M[, p])
  - Vírgula fixa:
    M = Qtd máxima de dígitos;
    p = Qtd de dígitos decimais
  - Alguns SGBD usam os sinónimos: Numeric, Real.
- Float / Double [(p)]
  - Vírgula flutuante: os valores são armazenados em notação científica, com mantissa e expoente. P.ex: 1200 é 1.2E3.
  - Só relevante para grandes números que possam ser arredondados a partir de *p* dígitos.
  - *p* = qtd máximo de dígitos da mantissa.
  - Ocupação de memória: Float: 4 bytes; Double: 8 bytes.

https://mariadb.com/kb/en/data-types/

Char vs. *Varchar*

Diferem no armazenamento:

CHAR (n) – São sempre armazenados *n* caracteres; se o valor armazenado tem menos de *n* caracteres, o restante é preenchido com espaços em branco, os quais são retirados sempre que o valor é lido.

VARCHAR (n) – São armazenados apenas os caracteres necessários, aos quais acrescerá 1 byte para registar o tamanho da *string*, ou 2 bytes se *n* > 255.

Value	**CHAR (4)**	Storage required	**VARCHAR (4)**	Storage required
`''`	`'  '`	4 bytes	`''`	1 byte
`'ab'`	`'ab '`	4 bytes	`'ab'`	3 bytes
`'abcd'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes
`'abcdefgh'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes

Ao ser lido, é mostrado com espaços. Para retirar: usar função Rtrim (*valor*).

Extraído de: http://dev.mysql.com/doc/refman/8.0/en/char.html

Text e *Long Varchar*

Armazenam até 2.147.483.647 bytes de caracteres

Não podem ser usados em:

- Cláusulas ORDER BY, GROUP BY e UNION;
- Na cláusula WHERE, excepto com a *keyword* LIKE (mas pouco eficiente);
- *Joins* e *subqueries*;
- Índices,

  Excepto se o índice for de tipo *full-text*, sendo nesse caso indexadas as palavras individualmente, e não todo o valor *text* em questão. Este tipo de índices não é rentabilizado em pesquisas do tipo 'coluna = valor' nem 'coluna LIKE valor'.
- Em parâmetros de *stored procedures*;

Datetime vs. *Timestamp* / T-SQL: *Datetimeoffset*

Datetime inclui informação de datas e horas, com precisão até aos microssegundos.

Microssegundos são relevantes para aplicações tais como leilões online.

Timestamp / ***Datetimeoffset*** (SQL Server) guarda a mesma informação que *Datetime* e também o fuso horário (*time zone*).

É o tipo de dados a usar para BDs com operação transnacional.

https://docs.microsoft.com/en-us/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql

➔https://dev.mysql.com/doc/refman/8.0/en/datetime.html

➔https://dev.mysql.com/doc/refman/8.0/en/date-and-time-literals.html

No Transact-SQL (T-SQL), a mais difundida extensão ao SQL, o tipo de dados com fuso horário designa-se ***Datetimeoffset***, e não *Timestamp*.

➔https://docs.microsoft.com/en-us/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql

Date, Time, Datetime / Timestamp

Exemplo de aplicação:

- Atributo *Filme.Estreia* – Date

  Quanto à data de estreia de um filme, apenas interessa registar a sua data. Não interessa a hora, a qual, inclusive, será diferente de cinema para cinema.

- Atributo *Sessão.Hora* – Time

  Neste atributo pretende-se registar a que horas existe cada sessão, em cada sala de cinema. A dita sessão existe todos os dias; logo, as datas concretas não são relevantes.

- Atributo *Reserva.DiaHora* – Datetime ou Timestamp

  Cada reserva precisa de ser rotulada não apenas com a hora nem apenas com o dia, mas sim para ambos.

Enum / Enumerado

```
Create table Produto_de_Vestuario (
    …
    tamanho ENUM ('XS', 'S', 'M', 'L', 'XL', 'XXL')
    …)
```

Domínios / Data Types (criados)

Novos tipos de dados podem ser definidos

CREATE { DOMAIN | TYPE } [AS] <*nome-domínio*> <*datatype*>;

- CREATE TYPE dm_Morada VARCHAR(100);
- CREATE TYPE dm_Dinheiro Numeric (9,2);
- CREATE TYPE dm_Preço Numeric (5,2);

Indicado para quando vários atributos partilham a mesma definição de tipo de dados

- Exºs:
  - Domínio Morada para as colunas:
    - Cliente.morada, Sucursal.morada, Encomenda.morada_entrega;
  - Domínio Tamanho_de_Vestuario para as colunas:
    - Produto_vestuário.Tamanho, Cliente.Tamanho_vestuario
- Objectivo: facilitar a manutenção da base de dados

Criação de Tabelas

Comando para criar uma tabela:

CREATE TABLE *<nome-da-tabela>* (
<definição-das-colunas>,
<restrições-de-integridade>)

Exemplo:

```
CREATE Datatype dm_morada VARCHAR(100);

CREATE TABLE Cliente (
    cod_cliente  INTEGER  NOT NULL,
    bi           INTEGER  NOT NULL,
    nome         VARCHAR(100),
    morada           dm_morada,
    CONSTRAINT   prim_key PRIMARY KEY (cod_cliente),
    CONSTRAINT   cand_key UNIQUE (bi));
```

Definição das colunas

Restrições de integridade

Chave alternativa

Criação de Tabelas II

Exemplos:

```
CREATE TABLE Factura (
num_factura    INTEGER        NOT NULL,
data      DATE      NOT NULL,
valor          DECIMAL(10,2)  NOT NULL,
cod_cliente    INTEGER        NOT NULL,
CONSTRAINT     prim_key  PRIMARY KEY (num_factura),
CONSTRAINT     for_key_cliente
     FOREIGN KEY (cod_cliente)
     REFERENCES Cliente (codigo)
     ON UPDATE CASCADE
     ON DELETE RESTRICT);
```

Chave estrangeira

```
CREATE TABLE Produto (
cod_produto  INTEGER   NOT NULL,
tipo      CHAR(2)  DEFAULT 'MP'  CHECK (tipo IN ('MP','PA')),
Designação    VARCHAR(100),
CONSTRAINT     prim_key PRIMARY KEY (cod_produto));
```

Valor por omissão

Restrição

Criação de Tabelas III

Exemplo:

```
CREATE TABLE Item (
     num_factura    INTEGER   NOT NULL,
     num_item       INTEGER   NOT NULL,
     quantidade     INTEGER   CHECK (quantidade > 0) NOT NULL,
     valor          DECIMAL (8,2)  NOT NULL,
     cod_produto    INTEGER        NOT NULL,
     CONSTRAINT     pk_item   PRIMARY KEY (num_factura, num_item),
     CONSTRAINT     fk_item_to_factura
          FOREIGN KEY (num_factura)
          REFERENCES Factura (num_factura)
          ON UPDATE CASCADE
          ON DELETE CASCADE,
     CONSTRAINT     fk_item_to_produto
          FOREIGN KEY (cod_produto)
          REFERENCES Produto (codigo)
          ON UPDATE CASCADE
          ON DELETE RESTRICT);
```

Restrição

Check constraints

As validações possíveis por via de *Check* são em geral poucas e muito simples. Na maioria dos sistemas:

- Só podem ser realizadas validações que envolvam atributos do próprio registo, valores constantes e funções de sistema determinísticas.
- Não são possíveis validações que incluam:
  - Funções de sistema não determinísticas. Por exemplo:
    - *Check data_nascimento* < *current_date()* não é possível.
  - *Stored functions* ou *procedures*.
- As validações que exijam este tipo de elementos têm que ser realizadas através de *triggers*.

De sistema para sistema há grande variação nas possibilidades para *Check*.

Alteração e remoção de tabelas

Comando para alterar uma tabela:

ALTER TABLE <*nome-da-tabela*>
<*alterações*>

Exemplo:

```
ALTER TABLE cliente
    ADD [COLUMN] telefone VARCHAR (10),
    DROP [COLUMN] bi,
    CHANGE [COLUMN] Numero INT IDENTITY;
```

Comando para apagar uma tabela:

DROP TABLE <*nome-da-tabela*>

Criação e remoção de índices

Comando para criar um índice numa tabela:

CREATE [UNIQUE] INDEX <*nome-do-índice*> **ON** <*nome-da-tabela*>
(<*coluna*> [ASC|DESC], ...)

Exemplo:

```
create unique index idx_pkey on Medicamentos_em_receita (
    Codigo,
    ID_Receita )
```

Comando para apagar um índice:

DROP INDEX <*nome-do-índice*>

Notas finais

É conveniente ter um ficheiro com a definição completa da base de dados. Esse ficheiro pode ser executado sempre que seja necessário reconstruir a base de dados.