SUMÁRIO

1. SOBRE CONSTRAINTS

A restrição no MySQL é usada para especificar a regra que permite ou restringe quais valores / dados serão armazenados na tabela. Eles fornecem um método adequado para garantir a precisão e integridade dos dados dentro da tabela. Também ajuda a limitar o tipo de dados que serão inseridos na tabela.

Desta forma, as constraints servem para restringir a entrada de dados em uma coluna, tendo estas, ainda, obviamente, uma limitação imposta pelo próprio tipo primitivo aceito. A exemplo, se uma coluna é do tipo “INT”, significa que apenas pode receber e armazenar dados numéricos inteiros, e ainda, se possuir a constraint “AUTO_INCREMENT”, significa que o número que armazena será sempre igual a unidade anterior mais 1, ou seja, será auto incrementado pelo próprio sistema.

Ademais, cada coluna pode ter, naturalmente, mais de uma constraint, ou seja, o quanto afunilamos a entrada de dados para um determinado campo depende apenas da compatibilidade entre as limitações impostas paralelamente.

1.1. COMO DEFINIR CONSTRAINTS

As constraints podem, e devem ser definidas para cada campo no momento da criação da tabela, com CREATE, ou, ainda, podem, se necessário, ser configuradas após a criação dos campos, com o comando de alteração ALTER.

2. PRINCIPAIS TIPOS DE CONSTRAINTS

2.1. NOT NULL

A constraint NOT NULL impõe a uma coluna a NÃO aceitar valores NULL, Ou seja, obriga um campo a sempre possuir um valor.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

Fonte: https://www.w3schools.com/sql/sql_notnull.asp

2.2. UNIQUE

A restrição UNIQUE identifica de forma única cada registro em uma tabela de um banco de dados, garantindo, assim como a constraint PRIMARY KEY, a unicidade em uma coluna ou conjunto de colunas. **A exemplo**, um campo de CPF pode ser UNIQUE, porque não podem existir dos registros com mesmo valor nesse campo.

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

Figura 1: https://www.w3schools.com/sql/sql_unique.asp

2.3. PRIMARY KEY

A chave primária, ou primary key, é o conceito mais básico relacionado à organização em um banco de dados. Toda tabela possuirá uma, e somente uma, chave primária. Essa chave é utilizada como identificador único da tabela. **A exemplo**, o código de um usuário, como identificador único e mais relevante de uma tabela de usuários, pode ser uma PK.

Vale uma comparação com a constraint UNIQUE. Tanto esta, quanto a PK garantem unicidade, e impossibilidade de repetição de valor do campo em outros registros da mesma tabela, mas a PK é um elemento de importância maior, que é utilizado para criar relacionamentos com tabelas externas – é um pilar do banco relacional.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

Fonte: https://www.w3schools.com/sql/sql_primarykey.asp

2.4. FOREIGN KEY

Como dito anteriormente, a PK é um elemento de identificação único utilizado para criar relacionamentos entre tabelas. Pois bem: a Foreign Key, ou chave estrangeira, é o campo de uma tabela que recebe a chave primária de outra. **A exemplo**, se uma tabela X possui um

campo cujo valor é dado do campo PK de outra tabela, dizemos que as duas possuem uma relação. Além disto, dizemos que a tabela que recebe o valor da chave primária, e que possui, portanto, um campo para chaves estrangeiras, é a “entidade dominante”.

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

Fonte: https://www.w3schools.com/sql/sql_foreignkey.asp

Na imagem acima, para deixar claro, o campo “PersonID int” da tabela “Orders”, foi criado para servir como chave estrangeira, isto é, como campo que irá receber a PK da tabela Persons, cujo nome também é PersonID. Entendendo isto, ainda podemos afirmar que a tabela Ordens é claramente a entidade dominante da relação.

2.5. AUTO INCREMENT

O incremento automático permite que um número único seja gerado automaticamente quando um novo registro é inserido em uma tabela. Geralmente é utilizado no campo de chave primária que gostaríamos de criar automaticamente toda vez que um novo registro for inserido.

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

Fonte: https://www.w3schools.com/sql/sql_autoincrement.asp

2.6. DEFAULT

A restrição DEFAULT é usada para inserir um valor padrão especificado em uma coluna, que será adicionado caso nenhum outro valor seja especificado para o campo do registro em um comando INSERT.

```sql
CREATE TABLE Orders (
    ID int NOT NULL,
    OrderNumber int NOT NULL,
    OrderDate date DEFAULT GETDATE()
);
```

Fonte: https://www.w3schools.com/sql/sql_default.asp

2.7. CHECK

A restrição CHECK é usada para limitar o intervalo de valores que pode ser colocado em uma coluna.

```sql
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

Fonte: https://www.w3schools.com/sql/sql_check.asp

FONTES

https://www.w3schools.com/sql/sql_check.asp

http://www.bosontreinamentos.com.br/mysql/mysql-constraints-restricoes-primary-key-fk-default-etc-06/