

Bases de Dados

Módulo 6: Modelo Relacional – Restrições (Constraints)

Prof. André Bruno de Oliveira

Data 14/12/2023 19:54

Tópicos

- **Modelo Relacional**
 - **Restrições (Constraints)**
 - **Chaves**
 - Chave Primária
 - Chave Candidata
 - **Valor NULL e Restrição NOT NULL**
 - **Integridade Referencial**
 - **Relacionamentos**
 - **Chaves Estrangeiras**
 - **Restrição CHECK**
 - **Implementando Restrições na SQL:**
 - PRIMARY KEY
 - NOT NULL
 - FOREIGN KEY
 - UNIQUE
 - CHECK

Restrições (*Constraints*)

- Em um BD relacional normalmente, existirá um conjunto de restrições (*constraints*) “regulando” os valores que os atributos das relações poderão assumir.
 - Essas restrições são derivadas das regras do **minimundo** que o BD representa.
 - Exemplos (BD do Censo Escolar sobre Escola):
 - Uma Escola é identificado unicamente por co_entidade.
 - Não é permitido cadastrar mais de uma Escola conteúdo no atributo, o valor no_entidade.
 - As escolas devem pertencer a um conjunto predeterminado de Regiões, Unidades da Federação, Município, Localização e Dependência.

Restrições (*Constraints*)

- As principal categoria de *constraint* é chamada de *constraint* **explícitas** ou **de esquema**.
 - Este tipo de restrição pode ser definida diretamente nos esquemas através da DDL (Data Definition Language).
 - São 4 subtipos principais:
 - Restrições de Chave
 - Restrição NOT NULL
 - Integridade Referencial
 - Restrição CHECK

Restrições (*Constraints*)

- Chave
- Uma relação é um conjunto de tuplas.
 - Elementos de um conjunto são, por definição, distintos.
 - Portanto, todas as tuplas de uma relação R devem ser distintas.
 - Ou seja: não devem existir duas tuplas em R com a mesma combinação de valores para **todos** os seus atributos.
 - Ex.: na relação *Escola* não podem existir duas tuplas (duas escolas) com o mesmo conjunto {co_entidade, no_entidade, cod_regiao, cod_uf, cod_municipio, tp_dependencia, tipo_localizacao, qt_mat_bas_fem, qt_mabas_masc, qt_mat_bas, qt_doc_bas}

Restrições (*Constraints*)

Chave

- Na prática, para uma relação R , quase sempre existirá algum subconjunto de atributos que jamais possuirá duas tuplas com a mesma combinação de valores.
 - Exemplo:
 - A relação Escola tem o subconjunto{**cod_entidade**} que representa a chave.
 - Isto porque, como projetistas do BD, considerando o minimundo que estamos modelando, achamos que uma chave composta por **no_entidade** + **cod_regiao**+**cod_uf**+**cod_municipio** seja muito complexa, assim usamos o código do INEP, **co_entidade**.

Escolas

	co_entidad	no_entidade	cod_reg	cod_uf	cod_municipio	tp_dependencia	tp_localizacao	qt_mat_bas_fem	qt_mat_bas_masc	qt_mat_bas	qt_doc_bas
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1	11	1100205	3	1	70	58	128	7
2	11003391	EMEF MANOEL MACIEL NUNES	1	11	1100205	3	2	21	22	43	2
3	11009888	EEEFM LAURINDO RABELO	NULL	NULL	NULL	2	NULL	NULL	NULL	NULL	NULL
4	11017112	EMEIEF MARACATIARA	NULL	NULL	NULL	3	NULL	NULL	NULL	NULL	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL	11	NULL	NULL	NULL	NULL	NULL	155	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL	11	NULL	NULL	NULL	NULL	NULL	28	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA ...	3	35	3548401	3	1	54	50	103	13
8	110500201	EMEF CECILIA MEIRELES	NULL	11	NULL	NULL	NULL	NULL	NULL	155	NULL

Restrições (*Constraints*)

• Chave

- O nome da escola mantido no atributo **no_entidade** em conjunto com os atributos **cod_regiao**, **cod_uf**, **cod_municipio** até que traz esta ideia de chave única. Só que há linhas com valores nulos para **cod_regiao**, **cod_uf**, **cod_municipio**.
- O nome da escola é muito extenso e podem ocorrer abreviações ou enganos na hora da coleta que resultem em nomes homônimos. Assim, neste caso é desejável que a chave primaria seja formada por um único atributo. No mundo real, este tipo de situação costuma ser contornado com relatórios de validação. Isso vai depender de cada cenário do minimundo.

Escolas

	co_entidad	no_entidade	cod_reg	cod_uf	cod_municipio	tp_dependencia	tp_localizacao	qt_mat_bas_fem	qt_mat_bas_masc	qt_mat_bas	qt_doc_bas
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1	11	1100205	3	1	70	58	128	7
2	11003391	EMEF MANOEL MACIEL NUNES	1	11	1100205	3	2	21	22	43	2
3	11009888	EEEFM LAURINDO RABELO	NULL	NULL	NULL	2	NULL	NULL	NULL	NULL	NULL
4	11017112	EMEIEF MARACATIARA	NULL	NULL	NULL	3	NULL	NULL	NULL	NULL	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL	11	NULL	NULL	NULL	NULL	NULL	155	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL	11	NULL	NULL	NULL	NULL	NULL	28	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA ...	3	35	3548401	3	1	54	50	103	13
8	110500201	EMEF CECILIA MEIRELES	NULL	11	NULL	NULL	NULL	NULL	NULL	155	NULL

Chave

- **DEFINIÇÃO**

Uma chave de uma relação R é *um conjunto de atributos* que especifica uma restrição de unicidade: jamais duas tuplas de R poderão ter o mesmo valor para a chave.

- Ou seja: O valor da chave identifica unicamente cada tupla em uma relação.
 - Exemplos (minimundo sobre escola):
 - A chave da relação Uf é {cod_uf}
 - A chave da relação Municipio é {cod_municipio}
 - A chave da relação Regiao é {cod_regiao}

Chave Primária e Chave Candidata

- Algumas vezes, uma relação pode ter mais de uma chave.
 - Exemplo: na relação Carro, tanto a “placa” como o “chassi” podem identificar unicamente uma tupla.

Carro

	placa	chassi	fabricante	modelo	ano
1	ABC2A33	9BWCA11J0Y4000001	Volkswagen	Gol	2017
2	BBB1J25	9BHDA05XT05000319	Hyundai	HB20	2021
3	LSL9J89	9BDHE21J6A4450243	Fiat	Cronos	2019
4	KPM8E80	9BDZZZ30ZSP050001	Fiat	Mobi	2020
5	LXW1J30	9BWC68E92AP000001	Volkswagen	Fox	2021

- Nesse caso, cada uma das chaves é chamada de chave candidata.
- Uma das chaves candidatas deve ser escolhida como chave primária(*primary key* – PK). Ela será utilizada para identificar as tuplas em uma relação.
 - Como convenção, a PK aparece sublinhada no esquema da relação. Neste exemplo, escolhemos a “placa” como PK:
 - *Carro*(placa, chassi, fabricante, modelo, ano)

Chave Primária e Chave Candidata

- Quando há mais de uma possibilidade de escolha de chave, devemos seguir o seguinte princípio:
 - Escolher a chave primária mais simples é formada pelo menor número de atributos possíveis.

Exemplos:

- Entre duas chaves candidatas, uma composta por um único atributo e a outra composta por dois atributos, **escolha a primeira opção.**
- Se uma chave candidata é um atributo do tipo número inteiro e a outra um atributo alfanumérico, **escolha o número inteiro.**

Chave Primária e Chave Candidata

- Na relação *Imóveis* temos duas chaves candidatas:
 - inscrição: número de inscrição do imóvel na Prefeitura.
 - endereço + cep: esses dois atributos em conjunto nunca possuirão valores iguais em duas tuplas.
- Seguindo a regra apresentada no slide anterior (escolher a chave mais simples), determinamos {inscrição} como chave primária.

Relação *Imoveis*

<u>inscricao</u>	endereco	cep	valor_ipatu
123456	Rua Alpha, 44	11111-000	120,00
999999	Rua Alpha, 47	11111-000	120,00
345890	Rua Beta, 23, Bl. 2, Apto 1801	22222-333	245,00
698123	Rua Ômega, 1290	55555-999	202,00

Chave Primária e Chave Candidata

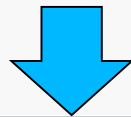
- Na linguagem SQL:
 - Uma restrição do tipo PRIMARY KEY seria definida para inscrição.
 - Uma restrição do tipo UNIQUE poderia ser definida para endereço + cep.
 - Mais detalhes a respeito da restrição UNIQUE serão apresentados ainda nesta aula.

Relação *Imoveis*

<u>inscricao</u>	endereco	cep	valor_ipatu
123456	Rua Alpha, 44	11111-000	120,00
999999	Rua Alpha, 47	11111-000	120,00
345890	Rua Beta, 23, Bl. 2, Apto 1801	22222-333	245,00
698123	Rua Ômega, 1290	55555-999	202,00

O Valor NULL

- O conceito de valor nulo (NULL) é utilizado em SGBD relacionais para representar a ausência de informação sobre um determinado campo.
- Se um campo de uma linha contém valor nulo, isto indica que seu conteúdo é desconhecido, inexistente, ou não-aplicável.
- No exemplo abaixo o valor da “cod_regiao” das escolas ‘11009888’ e ‘11017112’ é NULL (3ª e 4ª linha). Da mesma forma, outras colunas apresentam valor nulo.



Relação *Escola*

	co_entidad	no_entidade	cod_reg	cod_uf	cod_municipio	tp_dependencia	tp_localizacao	qt_mat_bas_fem	qt_mat_bas_masc	qt_mat_bas	qt_doc_bas
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1	11	1100205	3	1	70	58	128	7
2	11003391	EMEF MANOEL MACIEL NUNES	1	11	1100205	3	2	21	22	43	2
3	11009888	EEEFM LAURINDO RABELO	NULL	NULL	NULL	2	NULL	NULL	NULL	NULL	NULL
4	11017112	EMEIEF MARACATIARA	NULL	NULL	NULL	3	NULL	NULL	NULL	NULL	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL	11	NULL	NULL	NULL	NULL	NULL	155	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL	11	NULL	NULL	NULL	NULL	NULL	28	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA ...	3	35	3548401	3	1	54	50	103	13
8	110500201	EMEF CECILIA MEIRELES	NULL	11	NULL	NULL	NULL	NULL	NULL	155	NULL

Restrição NOT NULL

- A restrição NOT NULL especifica se valores nulos (NULL) poderão ou não ser permitidos.
 - Exemplo:
 - Se o minimundo do Censo Escolar indicar que cada tupla de escola deva obrigatoriamente ter um atributo **observação** com conteúdo especificado, então o atributo “**observacao**” precisará receber uma restrição do tipo NOT NULL.
 - De maneira oposta, se o preenchimento do campo observação do *Escola* for opcional, então o atributo “observacao” não deverá ser NOT NULL.

Relacionamentos

- Em um BD relacional tipicamente **existirão muitas relações**
- O próprio conceito de BD relacional sugere que os dados sejam armazenados em tabelas relacionadas de modo que as redundâncias de dados sejam minimizadas.
- Uma vez que o BD contém muitas relações, torna-se preciso manter atributos com valores em comum em relações diferentes, pois só assim será possível vincular os dados das mesmas.
- A seguir há as instâncias de duas relações: *Escola* e *Regiao*.

Escola

	co_entidade	no_entidade	cod_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3

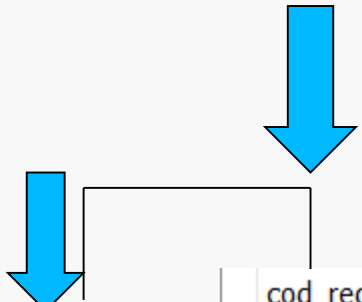
Regiao

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Relacionamentos

Observando a figura abaixo, responda a seguinte pergunta:

Em que Região a Escola ‘EEEFM LAURINDO RABELO’ está localizada?



	co_entidade	no_entidade	cod_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Relacionamentos

- No entanto, para conseguir determinar essa informação, você provavelmente precisou executar os seguintes passos.
- Inicialmente, procurou na relação *Escola* pelo conteúdo de `no_entidade` cujo o nome é 'EMEF MANOEL MACIEL NUNES', identificando então que código de região desta escola é '1'.
- Depois você foi até a relação *Região* para procurar pelo nome da região cujo código é 1, identificando assim que se tratava da região NORTE.

Escola

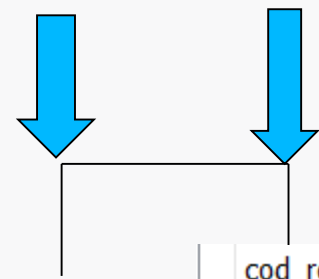
	co_entidade	no_entidade	cod_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3

Regiao

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Um instante

- Não foi muito simples identificar a escola desejada fazendo uma busca por nome em cada linha da coluna **no_entidade**. Imagine isso numa tabela com milhares de escolas. É muito mais simples usar o atributo chave **co_entidade**, pois ele é do tipo numérico com um tamanho menor diferente do atributo **no_entidade** do tipo **alfanumérico com um tamanho bem maior**.



	co_entidade	no_entidade	cod_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Relacionamentos

IMPORTANTE:

- Os atributos que armazenam os valores **não precisam ser do mesmo nome**. Por exemplo, na relação *Escola* o nome do atributo `cod_regiao` poderia ser `co_regiao` e diferente do nome contido na relação *Regiao*. O uso de nomes iguais para os campos de mesmo significado facilita a compreensão do esquema.
- Para o caso prático, não haverá impedimentos se os nomes forem diferentes, o que importa é que eles representam o mesmo conceito (códigos de regiões do Brasil).

Escola

	co_entidade	no_entidade	co_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DO...	3



Regiao

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Chave Estrangeira

Exemplo - Relações *Carro* e *FabricanteCarro*.

- Na relação *Carro*, o atributo “fabricante” designa a empresa fabricante do carro.
- Para implementar a integridade referencial, devemos determinar que o atributo “fabricante” seja uma chave estrangeira (**foreignkey–FK**) de *Carro* referenciando o atributo “nome” da relação *FabricanteCarro*.
- Isto significa que o valor de “fabricante” em qualquer tupla de *Carro* precisa “estar contido” no conjunto de valores do atributo “nome” pertencente à relação *FabricanteCarro* (o atributo “nome” é chave primária em *FabricanteCarro*).

<i>Carro</i>					
	placa	chassi	fabricante	modelo	ano
1	ABC2A33	9BWCA11J0Y4000001	Volkswagen	Gol	2017
2	BBB1J25	9BHDA05XT05000319	Hyundai	HB20	2021
3	LSL9J89	9BDHE21J6A4450243	Fiat	Cronos	2019
4	KPM8E80	9BDZZZ30ZSP050001	Fiat	Mobi	2020
5	LXW1J30	9BWC68E92AP000001	Volkswagen	Fox	2021

<i>FabricanteCarro</i>	
	nome
1	Chevrolet
2	Hyundai
3	Volkswagen
4	Fiat

Chave Estrangeira

- Para implementar a integridade referencial entre a relação *Escola* e *Regiao* devemos determinar que o atributo **cod_regiao** de *Escola* seja chave estrangeira da chave primária **cod_regiao** na relação *Regiao*. Como todo conjunto tem como subconjunto o valor nulo, a associação referencial entre *Escola* e *Regiao* através da chave **cod_regiao** aceita nulo (vazio) como valor válido para o domínio de **cod_regiao** na relação *Escola*.

Escola

	co_entidade	no_entidade	cod_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3

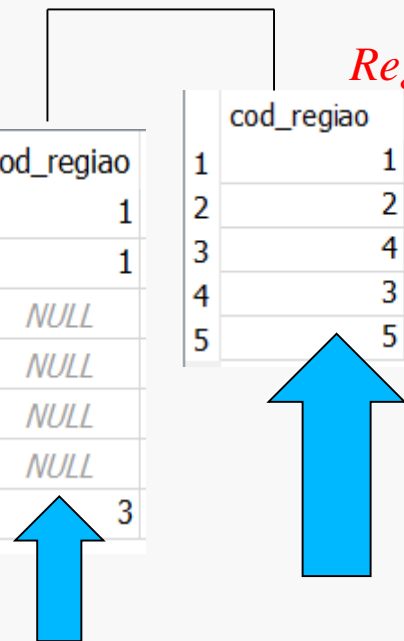
Regiao

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Chave Estrangeira

- Como você já deve ter percebido a chave estrangeira (**FK**) tem duas propriedades muito importantes.
 - **1ª Propriedade.**
 - A chave estrangeira em uma relação **A** deve referenciar uma chave primária na relação **B**.

<i>Escola</i>			<i>Regiao</i>	
	co_entidade	no_entidade	cod_regiao	
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1	1 NORTE
2	11003391	EMEF MANOEL MACIEL NUNES	1	2 NORDESTE
3	11009888	EEEFM LAURINDO RABELO	NULL	4 SUL
4	11017112	EMEIEF MARACATIARA	NULL	3 SUDESTE
5	11050021	EMEF CECILIA D'ALEMBERT	NULL	5 CENTRO OESTE
6	11050527	ESCOLA CASTELINHO DO SABER	NULL	
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3	



Chave Estrangeira

- **2ª Propriedade** : A chave estrangeira em *A* pode aceitar apenas:
 - Valores que estejam armazenados na chave primária de *B*;
 - O valor NULL pode ser aceito desde que na definição da chave estrangeira (cod_regiao de *Escola*) não inclua alguma restrição sobre o valor nulo (NOT NULL).

	co_entidade	no_entidade	cod_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Chave Estrangeira

- Considerando a instância de *Escola* e *Regiao* apresentadas abaixo, o atributo “cod_regiao” em *Escola* aceitará apenas o seguinte conjunto de valores:
 - Os valores armazenados no atributo “cod_regiao” de *Regiao*: {1,2,3, 4 e 5}. Se houver a tentativa de inserir uma região com o valor 6 no atributo cod_regiao de *Escola*, o SGBD rejeitará (haverá tentativa de violação da integridade referencial).
 - O valor NULL (supondo que “cod_regiao” de *Escola* não foi declarado como NOT NULL).

Escola

	co_entidade	no_entidade	cod_regiao
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1
2	11003391	EMEF MANOEL MACIEL NUNES	1
3	11009888	EEEFM LAURINDO RABELO	NULL
4	11017112	EMEIEF MARACATIARA	NULL
5	11050021	EMEF CECILIA D'ALEMBERT	NULL
6	11050527	ESCOLA CASTELINHO DO SABER	NULL
7	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3

Regiao

	cod_regiao	nome_regiao
1	1	NORTE
2	2	NORDESTE
3	4	SUL
4	3	SUDESTE
5	5	CENTRO OESTE

Restrição CHECK

- A restrição CHECK define uma regra que um valor de atributo deve satisfazer em uma relação.
- Este tipo de restrição é normalmente utilizado para **determinar o domínio** de um atributo.
- **Exemplo:** Na relação Escola, uma restrição do tipo CHECK pode ser definida sobre o atributo “qt_mat_bas” para garantir que este atributo aceite apenas valores entre 0 de 1000.

Constraints em SQL

- O slides a seguir, incluem a **sintaxe básica** para a definição de tabelas com constraints em SQL.
- Todos os exemplos podem ser executados no SQLite.
- Ou no emulador de banco de dados <https://sqliteonline.com/> para os BD Sqlite, MariaDB, PostgreSQL, Microsoft SQL Server.



Constraints em SQL

NOT NULL e Chave Primária – Sintaxe na SQL:

```
CREATE TABLE FabricanteCarro (  
nome VARCHAR(20) NOT NULL,  
PRIMARY KEY (nome));
```

Na definição acima:

- A tabela *FabricanteCarro* é criada.
- Ela contém um único campo chamado “nome” com as seguintes restrições:
 - NOT NULL
 - Chave Primária
- **ATENÇÃO:** A restrição de um atributo chave não aceitar nulo (NOT NULL) geralmente faz parte de uma validação que o SGBD inclui automaticamente.

Constraints em SQL

Chave Estrangeira –Sintaxe na SQL:

```
CREATE TABLE Carro (  
  placa CHAR(7) NOT NULL,  
  chassi CHAR(17) NOT NULL,  
  fabricante VARCHAR(20),  
  modelo VARCHAR(30) NOT NULL,  
  ano INT,  
  PRIMARY KEY (placa),  
  FOREIGN KEY (fabricante) REFERENCES FabricanteCarro(nome)  
);
```

- Na definição acima:
 - A tabela *Carro* é criada.
 - Os campos “placa”, “chassi” e “modelo” são definidos como NOT NULL.
 - O campo “placa” é definido como chave primária.
 - É definida uma **chave estrangeira** que vincula o campo “fabricante” de Carro com o campo “nome” de Empresa contido na tabela *Fabricante*.
 - O valor de “fabricante” da tabela Carro deve ser ou um dos valores constantes em “nome” da tabela Fabricante ou NULL.

Constraints em SQL

- **Restrição UNIQUE**

- Outro tipo de restrição que pode ser definida em um comando CREATE TABLE é a restrição UNIQUE.
- Ela é utilizada para indicar ao BD que determinado campo ou conjunto de campo(s) representa(m) uma chave candidata que não foi escolhida como PK da tabela.
- Ou seja: a restrição UNIQUE impede dois registros com o mesmo valor na(s) coluna(s) indicada(s).

Constraints em SQL

- **Restrição UNIQUE –Sintaxe na SQL:**

```
CREATE TABLE Imoveis (  
    inscricao INT NOT NULL,  
    endereco VARCHAR(100) NOT NULL,  
    cep CHAR(8) NOT NULL,  
    valor NUMERIC,  
    PRIMARY KEY (inscricao),  
    UNIQUE (endereco, cep)  
);
```

- A tabela *Imoveis* no SQL acima é criado de modo que:
 - O campo “inscricao” é definido como chave primária (não podem existir dois imóveis com a mesma inscrição).
 - Além disso, uma restrição do tipo UNIQUE é definida para os campos “endereco” e “cep” (chave candidata).
 - Com isto, o SGBD também não aceitará dois imóveis que possuam o mesmo endereço e cep.

Constraints em SQL

- **Constraint CHECK (Exemplo 1) –Sintaxe na SQL:**

```
CREATE TABLE Cliente (  
  id INT NOT NULL,  
  nome VARCHAR(60) NOT NULL,  
  ano_nasc INT NOT NULL,  
  possui_filhos CHAR(1) NOT NULL,  
  PRIMARY KEY (id),  
  CHECK (possui_filhos IN ('S', 'N'))  
);
```

- No SQL de definição acima:
 - A tabela *Cliente* é criada, com os campos “id” (PK), “nome”, “ano_nasc” e “possui_filhos” definidos como NOT NULL.
 - É definida uma restrição do tipo CHECK sobre a coluna “possui_filhos” com o objetivo de garantir que apenas dois valores sejam permitidos: ‘S’ (de sim) ou ‘N’ (de não).
 - Uma *constraint* CHECK implementa uma **restrição de domínio**.
 - Neste exemplo, estamos determinando que o domínio do atributo “possui_filhos” é o conjunto {‘S’, ‘N’}.
 - Note que {‘S’, ‘N’} é um subconjunto do tipo básico CHAR.

Constraints em SQL

- **Constraint CHECK (Exemplo 2) – Sintaxe na SQL:**

```
CREATE TABLE Empregado (  
  matricula CHAR(9) NOT NULL,  
  nome VARCHAR(60) NOT NULL,  
  salario NUMERIC NOT NULL,  
  PRIMARY KEY (matricula),  
  CHECK (salario >= 1000)  
);
```

- No SQL de definição acima:
 - A tabela *Empregado* é criada, com os campos “matricula” (PK), “nome” e “salario”, todos NOT NULL.
 - É definida uma restrição do tipo **CHECK** sobre a coluna “salario” com o objetivo de garantir que uma linha seja aceita apenas se o salário for igual ou superior a R\$ 1000,00.
 - Neste exemplo, estamos determinando que o domínio do atributo “salario” são os números reais iguais ou maiores que 1000.

Constraints em SQL

- **Constraint CHECK – observações**

No modelo relacional, as *constraints* do tipo check não são consideradas tão importantes quanto as PK e FK.

Na prática, costumam ser utilizadas com moderação e, na maioria das vezes, são definidas em campos simples, cujo domínio seja pequeno.

- Por exemplo:
 - Atributos binários, como $\{0, 1\}$ ou $\{'S', 'N'\}$.
 - Atributos que guardam informações variáveis categóricas com poucas categorias, como $\{'A', 'B', 'C', 'D'\}$

CHECK X FK

- Quando o domínio de um atributo necessitar de uma restrição maior o mais usual é adotar a FK.
- Exemplo: incluir uma validação de Unidade Federativa com 27 códigos para a tabela de *Escola*.
- O CHECK é mais indicado para restrições com domínios simples e que raramente devem mudar. A FK (chave estrangeira) é mais utilizada porque permite alterações em seu domínio que não alterem a definição do esquema e além disso inclui o recurso relacionar tabelas, o que permite exibir, por exemplo, uma descrição do domínio.

Escola

co_entidade	no_entidade	cod_uf
11001100	EMEF GEDOCY RUAS WOLFF	11
11002158	EMEF SAO FRANCISCO DE ASSIS	11
11003391	EMEF MANOEL MACIEL NUNES	11
11003901	EMEF ALUIZIO FERREIRA	11
11004428	EEEFM BURITI	11
11005360	EEEFM PAULO FREIRE	11
11006722	APAE DE ARIQUEMES	11

Uf

	cod_uf	nome_uf	sigla_uf
1	11	RONDÔNIA	RO
2	12	ACRE	AC
3	13	AMAZONAS	AM
4	14	RORAIMA	RR
5	15	PARÁ	PA
6	16	AMAPÁ	AP
	⋮	⋮	⋮
25	51	MATO GROSSO	MT
26	52	GOIÁS	GO
27	53	DISTRITO FEDERAL	DF

Exercícios de fixação

a) Qual chave você sugere para as relações abaixo?

--- *Relação de Alunos de uma escola que tem até o nono ano.*

Aluno (nome, cpf, matrícula, idade, sexo, endereço)

-- *Relação de estabelecimentos comerciais*

Estabelecimento (nome_estabelecimento, nome_proprietário, idade, endereço, cnpj)

-- *Relação de livros publicados*

Livros (titulo, nome_autor, ano, ibsn, editora)

-- *Relação de funcionários de uma empresa*

Funcionario (nome, cargo, escolaridade, salário, data_nascimento, nome_da_mãe)

Exercícios propostos

- 1) Com o uso da ferramenta <https://sqliteonline.com/> utilize o MS SQL crie a tabela *FabricanteCarro*, faça a inserção da lista de fabricantes.

--Criar tabela FabricanteCarro

```
CREATE TABLE FabricanteCarro (  
nome VARCHAR(20) NOT NULL,  
Modelo VARCHAR(20),  
PRIMARY KEY (nome));
```

--Inserir fabricantes

```
INSERT INTO FabricanteCarro (nome, modelo) values ('Volkswagem', 'Gol');
```

```
INSERT INTO FabricanteCarro (nome , modelo) values ('Hyoundai', 'HB20');
```

```
INSERT INTO FabricanteCarro (nome , modelo) values ('Fiat', 'Cronos');
```

```
INSERT INTO FabricanteCarro (nome , modelo) values ('Chevrolet', 'Fox');
```

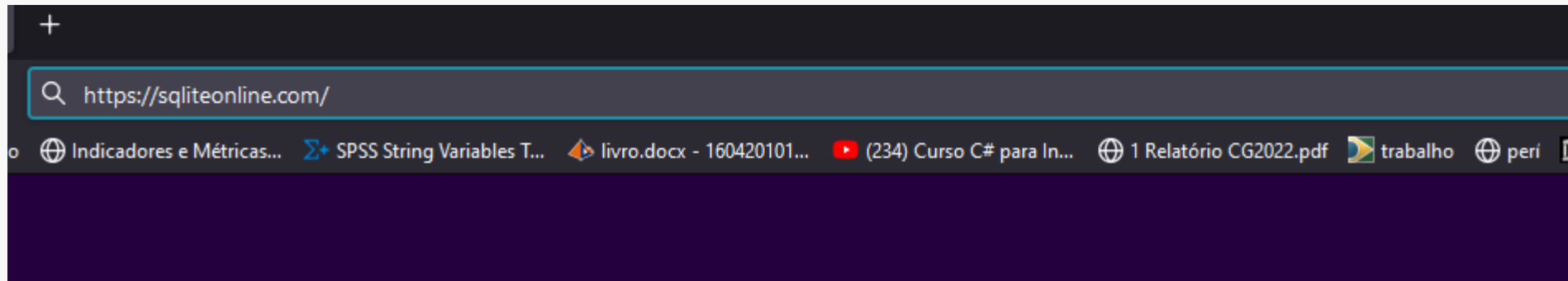
```
INSERT INTO FabricanteCarro (nome , modelo) values ('', 'Mobi');
```

```
INSERT INTO FabricanteCarro (nome , modelo) values ('Fiat', 'Cronos');
```

```
INSERT INTO FabricanteCarro (nome , modelo) values (null, 'Cronos');
```

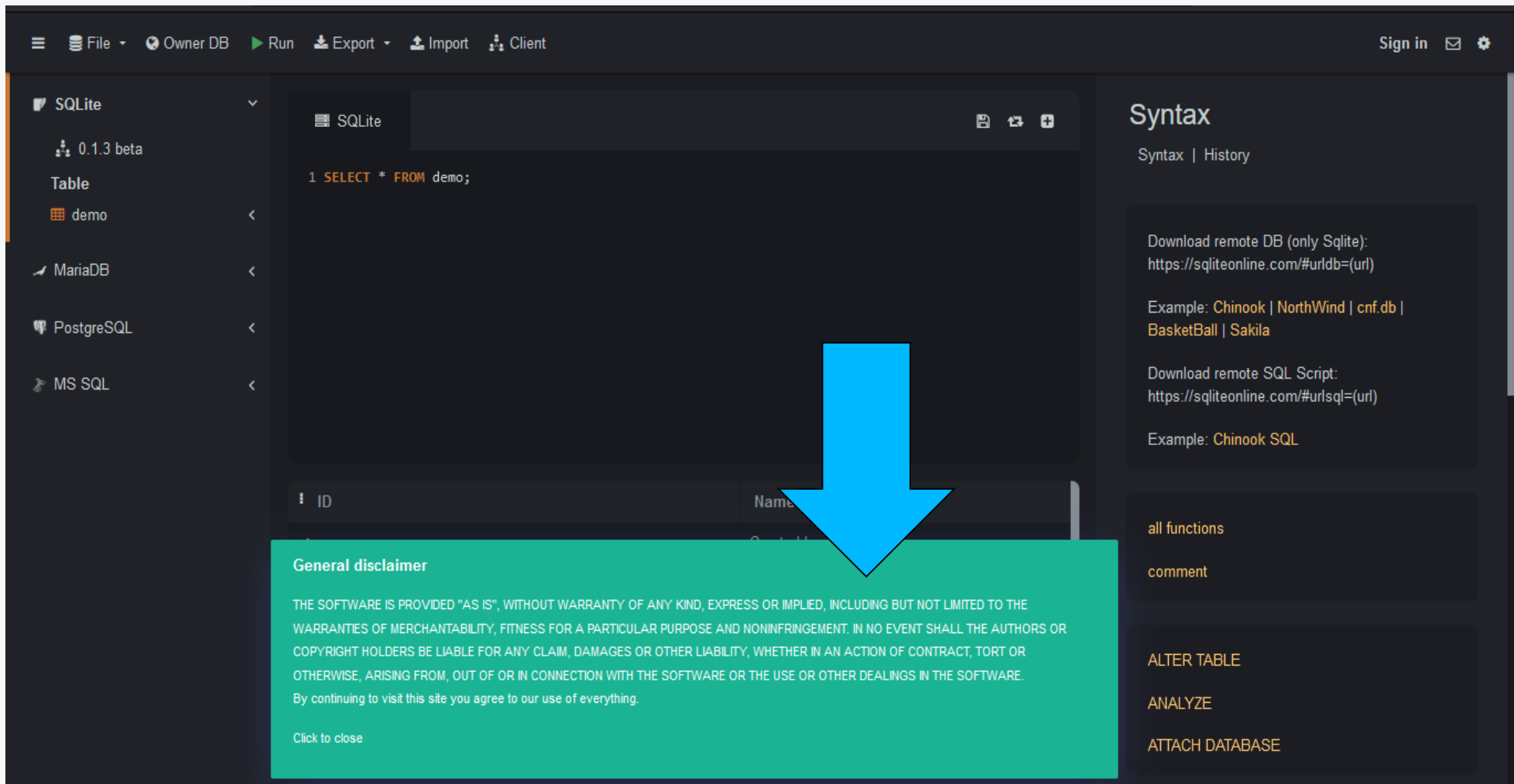
Emulador <https://sqliteonline.com/>

Abra seu navegador de Internet Chrome ou FireFox e na linha de endereço digite <https://sqliteonline.com/> e confirme com a tecla Enter.



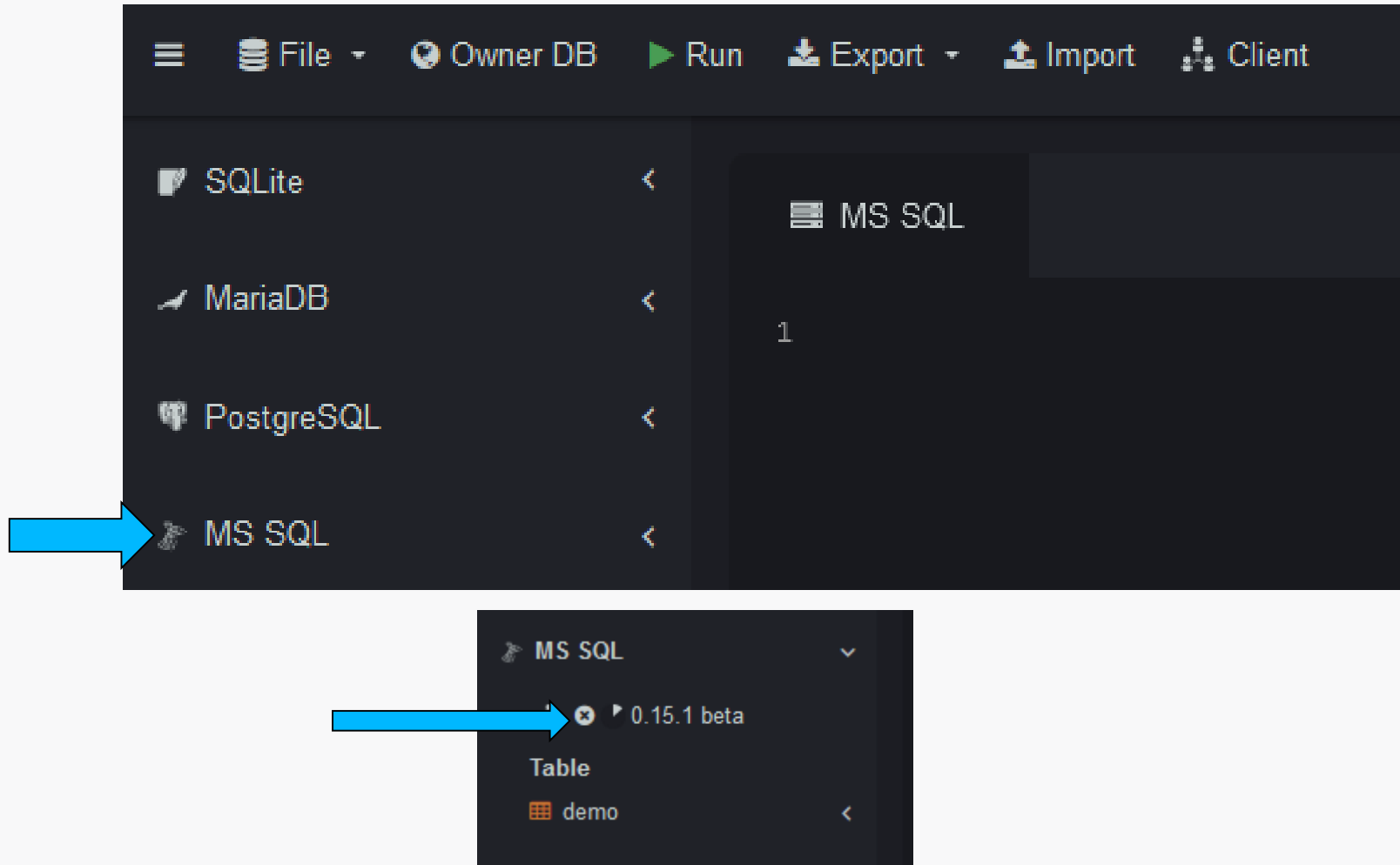
Emulador <https://sqliteonline.com/>

- A tela seguinte de abertura deve aparecer.
- Clique nesta imagem indicada abaixo para fechar.



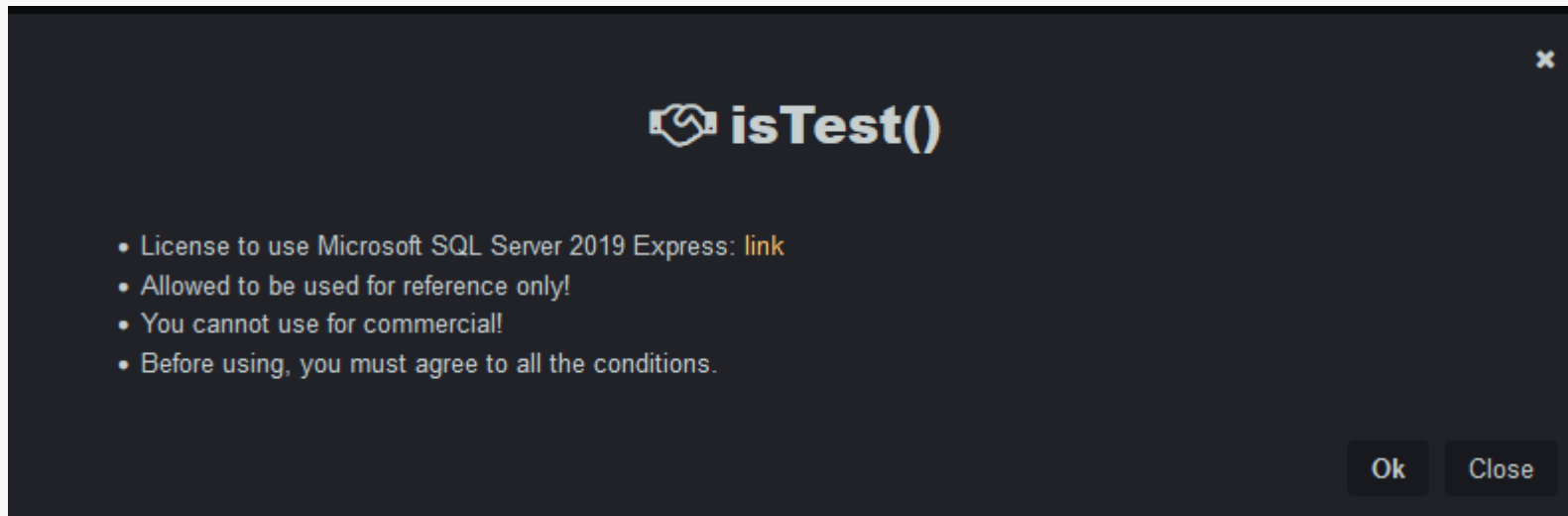
Emulador <https://sqliteonline.com/>

Selecione o banco de dados MS SQL (Microsoft SQL Server) e clique no ícone para abrir a conexão com o banco de dados.



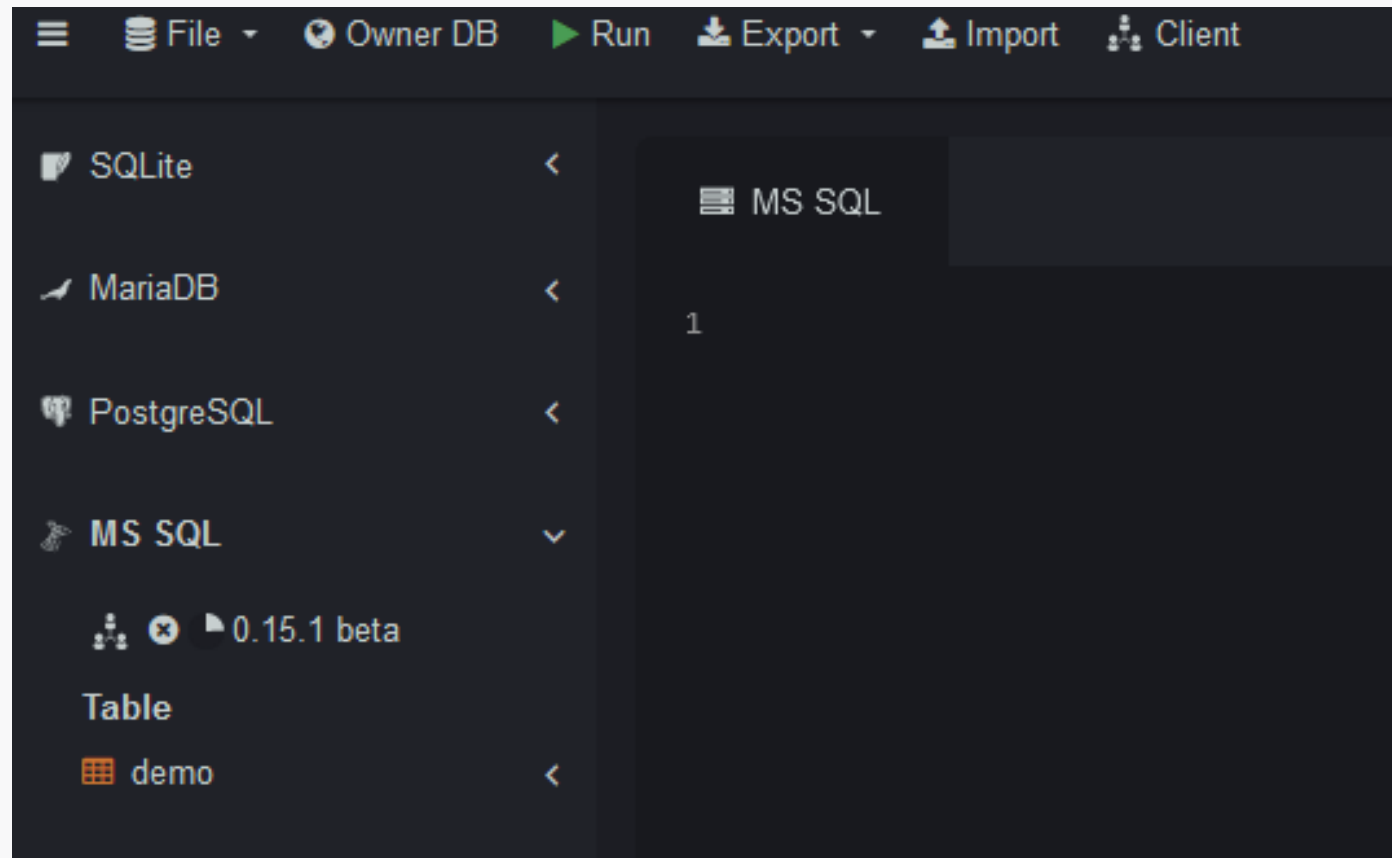
Emulador <https://sqliteonline.com/>

Clique em OK para confirmar a abertura da conexão.



Emulador <https://sqliteonline.com/>

- Após a conexão ser estabelecida a janela para edição *queries* (estará pronta para uso). O esquema padrão (DEFAULT) possui uma tabela de demonstração.



Emulador <https://sqliteonline.com/>

Escreva uma *query* para listar o conteúdo da tabela de demonstração para fazer um teste: *SELECT * FROM Demo*. Três colunas são exibidas.

MS SQL

1 **SELECT** * **FROM** DEMO

id	name	hint
1	test	1
2	server	mssql
3	limit time out	3h
4	limit db	70mb
5	limit count name: table, proc, fun, ..	300
6	limit overrun	auto-drop DB

Exercícios propostos

2) Use o mesmo SQL do exercício anterior para o BD PostgreSQL.

O que você percebeu ?

Exercícios propostos

- 3) Entre no site do SQLite (<https://www.sqlite.org/>) e veja como se faz para criar um índice para o atributo **uf** da tabela *Escola*.

Um índice serve para acelerar o tempo de acesso às linhas de uma tabela. Quando há uma tabela com milhões de linhas, o índice ajuda a economizar recursos e reduz o tempo processamento.

É fundamental estudar, executar
e testar os exemplos das aulas.

Obrigado