Banco de Dados I

Prof. Esp. Wesley Neves

12

AULA 12 – BANCO DE DADOS

Conteúdo

12.Restrições de Integridade

Regras de Integridade do Modelo Relacional

Integridade de Domínio - Restringe o conjunto de valores que podem ser gravados em uma coluna de uma tabela. Desta forma, somente os valores que pertencem ao domínio podem ser gravados na coluna da tabela. Outros valores não são permitidos e a atualização é desfeita pelo gerenciador de banco de dados.

Integridade de Identidade - A chave primária não pode conter valores nulos. Como toda informação em um banco de dados relacionam precisa ter uma identidade exclusiva, a chave primária deve ser obrigatoriamente preenchida.

Integridade Referencial - Se uma determinada tabela A possui uma chave estrangeira que estabelece relacionamento com uma tabela B, então o valor da chave estrangeira da tabela A deve ser igual a um valor de chave primária na tabela B. Esta regra garante que as referências de uma tabela para outra tabela sejam válidas, Como nem sempre o relacionamento entre tabelas é obrigatório uma chave estrangeira pode possuir valor nulo.

Tipo de Dados - Banco de Dados

Você sabia que escolhas ruins de tipo de dados podem ter um impacto significativo no design e no desempenho do banco de dados? **Desenvolvedores e DBAs podem melhorar o desempenho do banco de dados entendendo os tipos de dados suportados pelo SQL Server** e as implicações de escolher diferentes tipos. Uma prática recomendada é “dimensionar corretamente” os tipos de dados fazendo perguntas de negócios e determinando os tipos de dados mais adequados às necessidades da organização e do aplicativo.

Tipo de Dados - Banco de Dados

O dimensionamento correto pode resultar em uma enorme economia de armazenamento, o que pode levar a um desempenho mais rápido do banco de dados

Tipo de Dados - Banco de Dados

Um TIPO DE DADO nada mais é que algo do mundo real que pode ser representado computacionalmente.

O tipo de dados especifica o que atributo pode manter, como por exemplos, dados inteiros, dados de caracteres, dados monetários, data e hora, cadeias de caracteres binárias, etc.

Cada tipo de dados pode variar de acordo com SGBD (Sistema
Gerenciador de Banco de Dados).

Tipo de Dados - Banco de Dados

A. Cada SGBD tem caracteristicas próprias para seus tipos de dados
B. Os tipos de dados variam de acordo com seu valor (valor máximo aceitável) e tamanho (tamanho máximo aceitável)
C. **Eles podem ser de tamanho fixo ou variável**

Tipos de dados de largura váriavel

#1. No SQL Server, os tipos de dados de comprimento fixo e variável têm requisitos de armazenamento diferentes

- Os tipos de dados de largura variável sempre têm dois bytes extras de sobrecarga.
- Para colunas de largura variável, o tamanho de cada valor varia de registro para registro

Variable Width Data Types & Storage Requirements			
VARCHAR	n + 2 bytes	NVARCHAR	(n * 2) + 2 bytes

Tipos de dados de largura váriavel

#2. Ao dimensionar corretamente os tipos de dados, uma prática recomendada é analisar se um tipo de dados é o contêiner apropriado para o valor que será armazenado

Tipo de Dados - Banco de Dados

Categorias de tipo de dados

Os tipos de dados em SQL Server são organizados nas seguintes categorias:

Numéricos exatos

Numéricos aproximados

Data e hora

Cadeias de caracteres

Cadeias de caracteres Unicode

Cadeia de caracteres binária

Outros tipos de dados

Tipo de Dados - Banco de Dados

Numéricos exatos

tinyint

smallint

int

bigint

Tipos numéricos:

- **TINYINT**: Armazena valores numéricos inteiros, variando de 0 a 256

- **SMALLINT**: Armazena valores numéricos inteiros, variando de -32.768 a 32.767

- **INT**: Armazena valores numéricos inteiros, variando de -2.147.483.648 a 2.147.483.647

- **BIGINT**: Armazena valores numéricos inteiros, variando de -9.223.372.036.854.775.808 a -9.223.372.036.854.775.807

Tipo de Dados - Banco de Dados

Numéricos exatos

tinyint

smallint

int

bigint

```
1  DECLARE @variavel_TINYINT TINYINT =255;
2  DECLARE @variavel_SMALLINT SMALLINT =255;
3  DECLARE @variavel_INT INT =255;
4  DECLARE @variavel_BIGINT BIGINT =255;
5
6  SELECT 'TINYINT' [Data Type] , DATALENGTH(@variavel_TINYINT) AS [Quantidade Bytes TINYINT]
7  UNION
8  SELECT 'SMALLINT' [Data Type] , DATALENGTH(@variavel_SMALLINT) AS [Quantidade Bytes SMALLINT]
9  UNION
10 SELECT 'INT' [Data Type] , DATALENGTH(@variavel_INT) AS [Quantidade Bytes INT]
11 UNION
12 SELECT 'BIGINT' [Data Type] , DATALENGTH(@variavel_BIGINT) AS [Quantidade Bytes BIGINT]
13
```

133 %

Results | Messages

	Data Type	Quantidade Bytes TINYINT
1	BIGINT	8
2	INT	4
3	SMALLINT	2
4	TINYINT	1

Tipo de Dados - Banco de Dados

Numéricos exatos

- **SMALLMONEY**: Valores numéricos decimais variando de -214,748.3648 a 214,748.3647

- **MONEY**: Valores numéricos decimais variando de -922,337,203,685,477.5808 a +922,337,203,685,477.5807

smallmoney

money

Tipos de dados que representam valores monetários ou de moeda.

Os tipos de dados **money** e **smallmoney** têm a precisão de dez milésimos das unidades monetárias que eles representam. Para o Informatica, os tipos de dados **money** e **smallmoney** têm a precisão de um centésimo das unidades monetárias que eles representam.

Use um ponto para separar unidades monetárias parciais, como centavos, de unidades monetárias inteiras. Por exemplo, 2.15 especifica 2 dólares e 15 centavos.

Tipo de Dados - Banco de Dados

Numéricos exatos

smallmoney

money

Na tabela a seguir, você pode ver a diferença entre smallmoney e tipo de dados money no SQL Server:

Tipo de dado:	Valor mínimo:	Valor máximo:	Armazenamento:
smallmoney	-214,748.3648	214,748.3647	4 bytes
money	-922,337,203,685,477.5808	922,337,203,685,477.5807	8 bytes

Tipo de Dados - Banco de Dados

Numéricos exatos

smallmoney

money

Dinheiro vs Decimal

O Smallmoney parece equivalente ao decimal(10,4) e o Money é equivalente ao decimal(19,4). Mas elas não são as mesmas

O dinheiro é armazenado como inteiro

A principal diferença é que o dinheiro é armazenado como números inteiros. O Smallmoney é armazenado como int e o Money é armazenado como BigInt. Eles são armazenados sem casas decimais. As casas decimais são adicionadas somente quando você consulta e visualiza os dados.

Por exemplo

O número 1234,00 é armazenado como 12340000, 5555,5555 é armazenado como 55555555

O decimal é armazenado com um ponto decimal e dígitos decimais

Tipo de Dados - Banco de Dados

Numéricos exatos

smallmoney

money

As casas decimais do dinheiro são fixas.

O tipo de dados Money usa os 4 dígitos de fração fixos. Em Decimal, você pode definir o número de dígitos fracionários.

Perda de precisão

O dinheiro sofre com a perda de precisão, pois é tratado como um número inteiro em operações matemáticas. Especialmente, quando o usamos em operações envolvendo multiplicações e divisões.

O dinheiro armazena até 4 casas decimais. Se o resultado (mesmo para resultados intermediários) da multiplicação ou divisão resultar em um valor com mais de 4 casas decimais, será arredondado.

Tipo de Dados - Banco de Dados

Numéricos exa

smallmoney

money

```
1  DECLARE @variavel_SMALLMONEY SMALLMONEY = 2.55;
2  DECLARE @variavel_MONEY MONEY = 2.55;
3
4
5  SELECT 'SMALLMONEY' [Data Type],
6         @variavel_SMALLMONEY AS Valor,
7         DATALENGTH(@variavel_SMALLMONEY) AS [Quantidade Bytes SMALLMONEY]
8  UNION
9  SELECT 'MONEY' [Data Type],
10        @variavel_MONEY AS Valor,
11        DATALENGTH(@variavel_MONEY) AS [Quantidade Bytes MONEY];
12
13
```

33 %

Results | Messages

	Data Type	Valor	Quantidade Bytes SMALLMONEY
1	SMALLMONEY	2,55	4
2	MONEY	2,55	8

Tipo de Dados - Banco de Dados

Numéricos exatos

smallmoney

money

```
1  DECLARE @num1 SMALLMONEY;
2  SET @num1 = 1234.5678;
3  SELECT (@num1 / 10) * 10;
4
5
6
7  DECLARE @num2 DECIMAL(10,4)
8  SET @num2 = 1234.5678
9  SELECT (@num2 /10)* 10
```

.33 %

Results | Messages

	(No column name)
1	1234,567

	(No column name)
1	1234.5678000

Tipo de Dados - Banco de Dados

Numéricos exatos

smallmoney

money

As operações de inteiros são mais rápidas

As operações matemáticas como adições e subtrações são significativamente mais rápidas porque o dinheiro é tratado como números inteiros.

Dinheiro requer menos armazenamento

O dinheiro requer menos armazenamento em comparação com os decimais. O Smallmoney requer 4 bytes, enquanto o decimal equivalente(10,4) precisa de pelo menos 9 bytes. O dinheiro leva 8 bytes, enquanto o decimal(19,4) precisa de 9 bytes.

Como você pode ver, o uso do Smallmoney economiza muito espaço.

Tipo de Dados - Banco de Dados

decimal[(*p*[, *s*])] e **numeric**[(*p*[, *s*])]

Números de precisão e escala fixos. Quando a precisão máxima for usada, os valores válidos serão de - 10^38 +1 a 10^38 - 1. Os sinônimos ISO para **decimal** são **dez** e **dec**(*p*, *s*) . **numeric** é funcionalmente equivalente a **decimal**.

p (precisão)

O número total máximo de dígitos decimais a ser armazenados. Esse número inclui o que está à direita e à esquerda do ponto decimal. A precisão deve ser um valor de 1 até a precisão máxima de 38. A precisão padrão é 18.

Precisão	Bytes de armazenamento
1 - 9	5
10–19	9
20–28	13
29-38	17

Tipo de Dados - Banco de Dados

SQL

```
CREATE TABLE dbo.MyTable
(
  MyDecimalColumn DECIMAL(5,2)
,MyNumericColumn NUMERIC(10,5)

);

GO
INSERT INTO dbo.MyTable VALUES (123, 12345.12);
GO
SELECT MyDecimalColumn, MyNumericColumn
FROM dbo.MyTable;
```

SQL

```
MyDecimalColumn                          MyNumericColumn
---------------------------------------- ----------------
123.00                                   12345.12000

(1 row(s) affected)
```

Tipo de Dados - Banco de Dados

Numéricos exatos

bit

Tipo BIT:

– **BIT**: Armazena bits ou seja somente poderá conter os valores lógicos 0 ou 1.

Tipo de Dados - Banco de Dados

Numéricos aproximados

float

real

Sintaxe

float [(*n*)] em que *n* é o número de bits usado para armazenar a mantissa do número **float** na notação científica e, portanto, exige a precisão e o tamanho do armazenamento. Se *n* for especificado, ele precisará ser um valor entre **1** e **53**. O valor padrão de *n* é **53**.

Valor de *n*	Precisão	Tamanho de armazenamento
1-24	7 dígitos	4 bytes
25-53	15 dígitos	8 bytes

Tipos de dados numéricos aproximados para uso com dados numéricos de ponto flutuante. Os dados de ponto flutuante são aproximados; portanto, nem todos os valores no intervalo de tipo de dados podem ser representados de maneira exata. O sinônimo ISO de **real** é **float(24)** .

Tipo de Dados - Banco de Dados

Numéricos aproximados

float

real

Tipo de dados	Intervalo	Armazenamento
float	- 1,79E+308 a -2,23E-308, 0 e 2,23E-308 a 1,79E+308	Depende do valor de *n*
real	- 3,40E + 38 a -1,18E - 38, 0 e 1,18E - 38 a 3,40E + 38	4 bytes

Tipo de Dados - Banco de Dados

Data e hora

date

datetimeoffset

datetime2

smalldatetime

datetime

time

- *Date – 0001-01-01 a 9999-12-31*
- *DateTimeOffset – Data com hora, minutos e segundos com reconhecimento de fuso horário*
- *DateTime2 – Data com hora e minutos expressa de 0 a 24*
- *Smalldate – Data com hora expressa de 0 a 24*
- *Datetime – Data com hora, minutos e segundos expressa de 0 a 24*
- *Time – Hora, minutos e segundos expressas de 0 a 24*

Tipo de Dados - Banco de Dados

Data e hora

date

datetimeoffset

datetime2

smalldatetime

datetime

time

Tipo de dados	Saída
time	12:35:29. 1234567
date	2007-05-08
smalldatetime	2007-05-08 12:35:00
datetime	2007-05-08 12:35:29.123
datetime2	2007-05-08 12:35:29.1234567
datetimeoffset	2007-05-08 12:35:29.1234567 +12:15

Tipo de Dados - Banco de Dados

Valor	Tipo	Quantidade Bytes
2022-04-14 00:00:00.0000000 +00:00	DATE	3
2022-04-14 18:51:41.3600000 +00:00	DATETIME	8
2022-04-14 18:51:41.3600000 +00:00	DATETIME2(2)	6
2022-04-14 18:51:41.3600000 +00:00	DATETIMEOFFSET	10
2022-04-14 18:52:00.0000000 +00:00	SMALLDATETIME	4

```
DECLARE @variavel DATETIME = GETDATE();
SELECT @variavel AS Valor,
       'DATETIME' AS [Tipo],
       DATALENGTH(@variavel) AS [Quantidade Bytes]
UNION
SELECT CAST(@variavel AS DATE) AS Valor,
       'DATE' AS [Tipo],
       DATALENGTH(CAST(@variavel AS DATE)) AS [Quantidade Bytes]
UNION
SELECT CAST(@variavel AS DATETIME2(2)) AS Valor,
       'DATETIME2(2)' AS [Tipo],
       DATALENGTH(CAST(@variavel AS DATETIME2(2))) AS [Quantidade Bytes]
UNION
SELECT CAST(@variavel AS SMALLDATETIME) AS Valor,
       'SMALLDATETIME' AS [Tipo],
       DATALENGTH(CAST(@variavel AS SMALLDATETIME)) AS [Quantidade Bytes]
UNION
SELECT CAST(@variavel AS DATETIMEOFFSET) AS Valor,
       'DATETIMEOFFSET' AS [Tipo],
       DATALENGTH(CAST(@variavel AS DATETIMEOFFSET)) AS [Quantidade Bytes];
```

Tipo de Dados - Banco de Dados

Cadeia de caracteres

binária

binary

varbinary

- Binário – Cadeia de caracteres específico para armazenamento binário. Varia de 1 a 8000
- VarBinário – Cadeia de caracteres específico para armazenar binário. Varia até 2^31 -1 (2GBbytes.

imagem

Tipo de Dados - Banco de Dados

Cadeias de caracteres

char

varchar

text

- Char – Cadeia de caracteres não Unicode com o valor fixo (de 1 a 8000)
- Varchar – Cadeia de caracteres não Unicode com valor variável (de 1 a 8000). Podemos usar o tamanho MAX que representa 2^31-1 bytes (2GBytes)
- Texto – Dados não Unicode de comprimento variável para armazenar dados grandes ou binários

Tipo de Dados - Banco de Dados

Cadeias de caracteres Unicode

- nChar – Cadeia de caracteres Unicode com valor fixo (de 1 a 8000)
- nVarchar – Cadeia de caracteres Unicode com valor variável (de 1 a 8000). Podemos usar o tamanho MAX que representa 2^31 -1 bytes (2GBytes)
- nTexto – Dados Unicode de comprimento variável para armazenar dados grandes ou binários

nchar

nvarchar

ntext

Unicode – Fornece um único conjunto de caracteres contendo os idiomas do mundo.

Tipo de Dados - Banco de Dados

Outros tipos de dados

cursor rowversion hierarchyid

uniqueidentifier sql_variant

xml Tipos de geometria espacial Tipos de geografia espacial

table

Tipos de Restrições de Integridade

Os tipos de restrições de integridade que estudaremos neste artigo são os seguintes:

- Integridade Referencial
- Integridade de Domínio
- Integridade de Vazio
- Integridade de Chave
- Integridade Definida pelo Usuário

Integridade de Domínio

Valores inseridos em uma coluna devem sempre obedecer à definição dos valores que são permitidos para essa coluna – os valores do ***domínio***.

Ex.: em uma coluna que armazena preços de mercadorias, os valores admitidos são do domínio numérico – ou seja, apenas números. Não há preços usando letras para sua representação.

Integridade de Domínio

Employee_id	Name	Salary	Age
1	Andrew	486522	25
2	Angel	978978	30
3	Anamika	697abc	35

This value is out of domain(not INTEGER)so it is not acceptable.

Integridade de Domínio

- Restrições de domínio são a forma mais elementar de restrições de integridade.

- ***Estas*** testam valores inseridos no Banco de Dados, e testam (***efetuam***) consultas para assegurar que as comparações façam sentido.

Integridade de Domínio: Fatores

Os seguintes fatores impactam a integridade de domínio:

- Tipo de Dado do campo
- Representação interna do tipo de dado
- Presença ou não do dado
- Intervalos de valores no domínio
- Conjuntos de valores discretos

Exemplo

Suponha um atributo ***Preço do Produto***: Valor Monetário

Exemplo

Suponha um atributo ***Preço do Produto***: Valor Monetário

- Valor permitido:
  - 25,33
- Valores não permitidos:
  - 25 Reais e 33 centavos
  - -32,33

Neste exemplo, não há preços usando letras ou valores negativos, e o valor 25,33 atende à integridade de domínio (numérico e positivo).

Integridade de Domínio

A cláusula check em SQL-92 permite restringir domínios

Utilize a cláusula ***check*** para assegurar que um domínio de hora em Hora-Salario permita só valores maiores do que o especificado.

```
create domain hora-salario numeric(5,2)
        constraint valor-teste check(valor >=4.00)
```

Integridade de Domínio

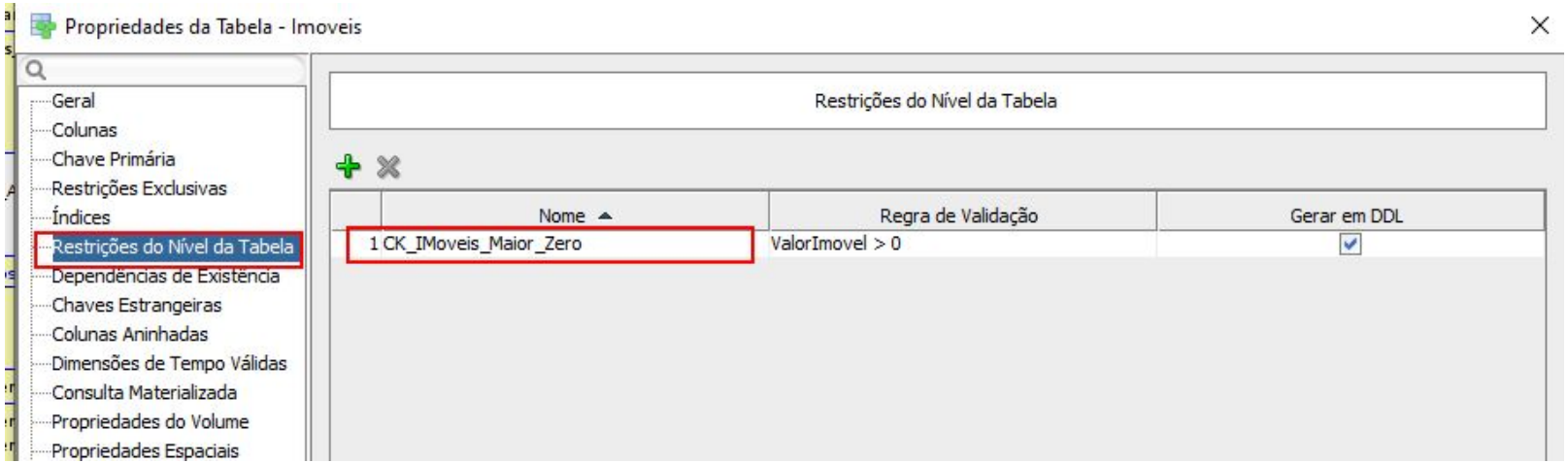
A cláusula check em SQL-92 permite restringir domínios

- A cláusula **check** usada para restringir um determinado conjunto de valores por meio do uso da cláusula **in**

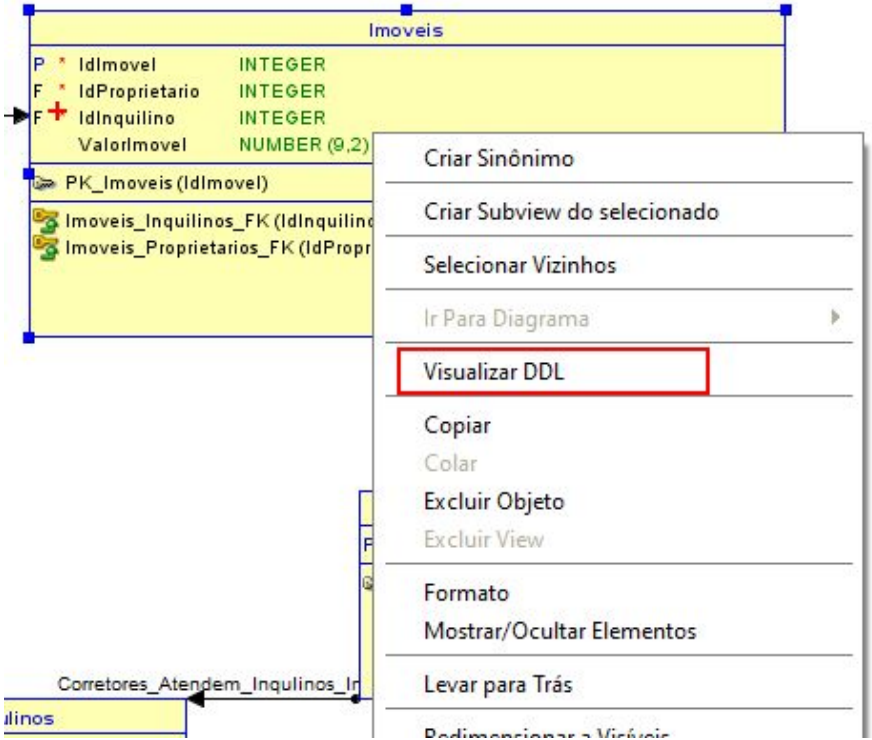
  - Exemplo 3 :

```
create domain tipo_conta char (10)
   constraint teste_tipo_conta check(value in ("corrente",
   "Poupança"))
```

Integridade de Domínio



Integridade de Domínio



Integridade de Domínio

```
1  CREATE TABLE imoveis (
2      idimovel       INTEGER NOT NULL,
3      idproprietario INTEGER NOT NULL,
4      idinquilino    INTEGER NOT NULL,
5      valorimovel    NUMBER(9, 2) NULL
6  )
7  ORGANIZATION HEAP NOCOMPRESS
8      NOCACHE
9          NOPARALLEL
10     NOROWDEPENDENCIES DISABLE ROW MOVEMENT;
11
12 ALTER TABLE imoveis
13     ADD CONSTRAINT ck_imoveis_maior_zero CHECK ( valorimovel > 0 ) NOT DEFERRABLE ENABLE VALIDATE;
14
```

Integridade de Domínio

```
create table cliente (
    nome char(30) NOT NULL,
    sexo char(1) CHECK (sexo IN ('M', 'F')),
    CPF number(11) UNIQUE,
    endereco char(40),
    cidade char(20) DEFAULT 'São Paulo'
)
```

DOMÍNIO E VALORES NULOS

No modelo Relacional para cada coluna da tabela, deve ser especificado um conjunto de valores (alfanumérico, numérico,...) que os campos da respectiva coluna podem assumir:

- Este conjunto de valores é chamado **de domínio da coluna ou domínio do campo;**

No modelo Relacional deve ser especificado se os campos da coluna podem estar vazios ("null" em inglês) ou não:

- Estar vazio indica que o campo não recebeu nenhum valor de seu domínio.

DOMÍNIO E VALORES NULOS

As colunas nas quais não são admitidos valores vazios são chamadas de **colunas obrigatórias**;

As colunas nas quais podem aparecer campos vazios são chamadas de **colunas opcionais;**

Os SGBD relacional exigem que todas colunas que compõem a PK sejam obrigatórias:

- A mesma exigência não é feita para a FK

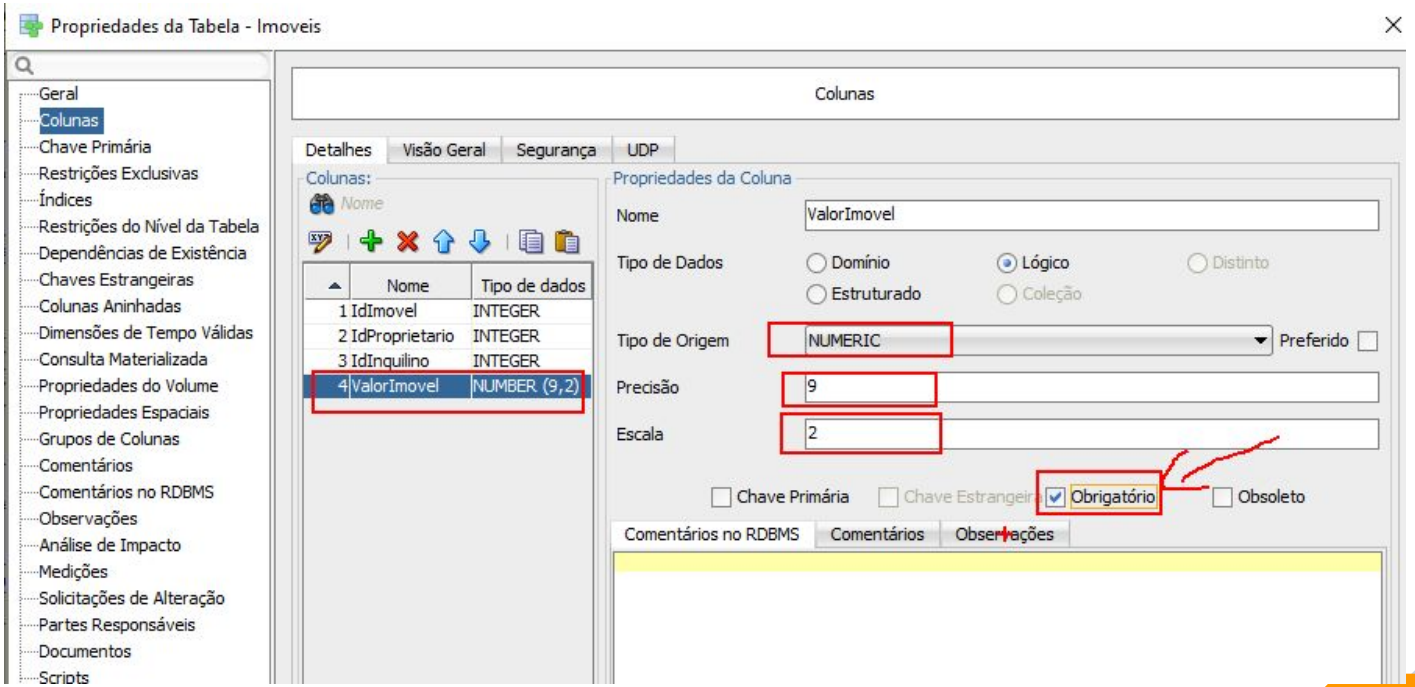
DOMÍNIO E VALORES NULOS

- **Restrições de valor vazio**
  - O cliente 548 não tem nome.
  - Esta tupla se refere a um cliente anônimo, que não tem muito sentido no BD.
  - Este pode ser um caso caso em que se deseja proibir valores vazios, restringindo domínios do atributo nome para ***not null***.
  -

Matricula	Nome	endereco
548		Rua Carvalho 615
549	Pedro	Rua Pedro Chaves 22
...		

- Um valor de campo pode assumir o valor vazio ("null" em inglês)

DOMÍNIO E VALORES NULOS



DOMÍNIO E VALORES NULOS

```
CREATE TABLE imoveis (
    idimovel        INTEGER NOT NULL,
    idproprietario  INTEGER NOT NULL,
    idinquilino     INTEGER NOT NULL,
    valorimovel     NUMBER(9, 2) NOT NULL
)
ORGANIZATION HEAP NOCOMPRESS
    NOCACHE
        NOPARALLEL
    NOROWDEPENDENCIES DISABLE ROW MOVEMENT;
```

Valor padrão

Pode-se definir um valor padrão para uma coluna acrescentando à sua definição a cláusula DEFAULT. Esta cláusula permite substituir automaticamente os valores nulos por um valor inicial desejado.

```
create table cliente(
    nome char(30) NOT NULL,
    sexo char(1),
    CPF number(11) NOT NULL,
    endereco char(40),
    cidade char(20) DEFAULT 'São Paulo',
    sexo char(10) DEFAULT 'masculino'
)
```

Restrições de Chave primária

- Restrições de Chave
  - **Regra "Chave Primária"**
    - Restringe que cada linha de uma tabela deve ser identificada por um valor único.

- • Pode ser simples ou composta.

Chave simples

```
Create table medico
     (codigoM integer not null,
      nome varchar(30) not null,
      endereco varchar(35),
      PRIMARY KEY (matricula) )
```

chave composta

```
Create table consulta
     (codigoMedico integer not null,
      codigoPaciente integer not null,
      data date not null,
      PRIMARY KEY (codigoMedico,
codigoPaciente, data))
```

Restrições de Chave primária

RESTRIÇÕES DE CHAVE

- Restringem os valores permitidos em determinados atributos de relações.
  - Lembrar definição de chave
- São especificadas como parte da instrução CREATE TABLE.
  - Chaves primárias são especificadas usando `PRIMARY KEY`
  - Outras chaves podem também ser especificadas usando `UNIQUE`

```
CREATE TABLE Executivos(
   id          CHAR(20),
   nome        CHAR(50),
   login       CHAR(16),
   idade       INTEGER,
   salario     NUMERIC,
   CONSTRAINT nome_idade_u UNIQUE (nome, idade),
   CONSTRAINT executivos_pk PRIMARY KEY (id)
);
```

Restrições Chave Alternativa

- **Chave Candidata**
  - Restrições ***Unique*** garantem que os dados contidos em uma coluna ou um grupo de colunas é o único em relação a todas as linhas da tabela.
    - Sintaxe: Quando escrita como um restrição de coluna
    - CREATE TABLE produto (nroProduto **integer UNIQUE,** nome varchar(30), preco real).
      - Sintaxe: Quando escrita como uma restrição de tabela.
      - CREATE TABLE produto (nroProduto integer, nome varchar(30), preco_real, **UNIQUE (Prodct_no)**);

Restrição de Unicidade

```
CREATE TABLE Executivos(
   id          CHAR(20),
   nome        CHAR(50),
   login       CHAR(16),
   idade       INTEGER,
   salario     NUMERIC,
   CONSTRAINT nome_idade_u UNIQUE (nome, idade),
   CONSTRAINT executivos_pk PRIMARY KEY (id)
);
```

- Podem existir várias declarações `UNIQUE` mas só uma chave primária
- Fora do standard SQL-92, uma implementação pode atribuir mais significado aos campos chave primária e, automaticamente um índice para melhorar o desempenho
  - É típico serem criadas *tabelas de hash*
- Em alguns sistemas podem ser criadas chaves em conjunto com índices

```
CREATE UNIQUE INDEX nomeIdx ON Actor(nome);
```

- Pode proibir-se a existência de valores nulos para atributos
  - Através do uso de `NOT NULL`
- `PRIMARY KEY = UNIQUE + NOT NULL`

Integridade Referencial

Uma restrição de Integridade Referencial assegura que valores de uma coluna em uma tabela são válidos baseados nos valores em uma outra tabela relacionada.

```
create table movimento (
    agencia number(5),
    conta number(7),
    valor number(16,2),
    primary key (agencia, conta),
    foreign key (agencia) references agencias,
    foreign key (conta) references contas
)
```

Integridade Referencial

Tipos de integridade	Tipo de constraint
Domínio	DEFAULT CHECK REFERENCIAL
Entidade	PRIMARY KEY UNIQUE
Referencial	FOREIGN KEY CHECK