Faculdade IMPACTA TECNOLOGIA

Faculdade IMPACTA

Linguagem SQL / Banco de Dados

Aula 03 – Projeto Físico:

DATA DEFINITION LANGUAGE (DDL)

Gustavo Bianchi Maia
gustavo.maia@faculdadeimpacta.com

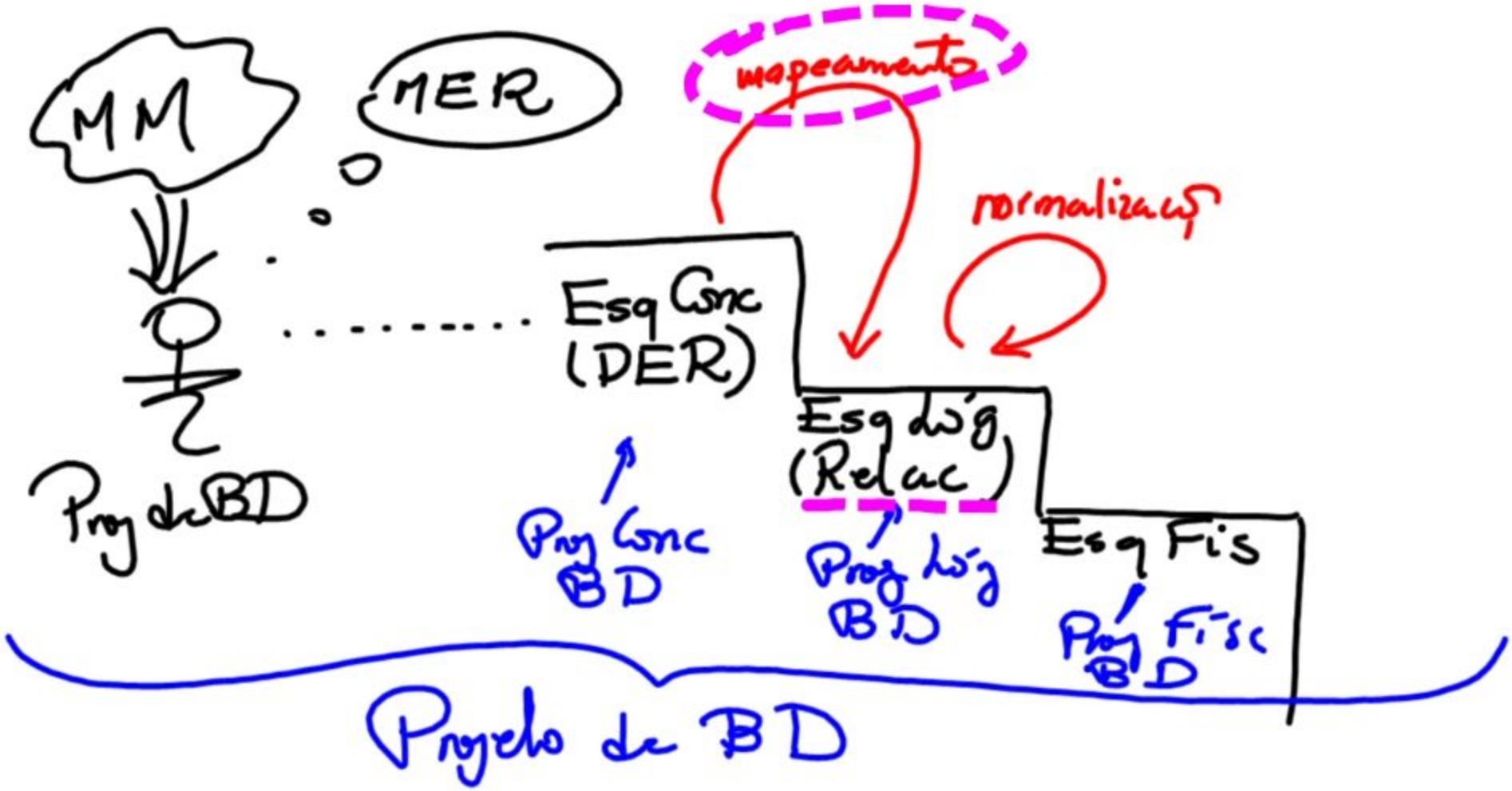
Faculdade IMPACTA TECNOLOGIA

Agenda

- Modelo Físico
  - Tipagem
  - Nulidade
  - Constraints
  - Atributos auto-incrementais
- Syntaxe da sub-linguagem DDL
- Exercícios

Faculdade IMPACTA TECNOLOGIA

Modelo de Dados Relacional



Tipos de Dados

Bancos de dados associam tipos de dados a colunas, expressões, variáveis e parâmetros.

Tipos de dados determinam quais os tipos de valores serão permitidos no armazenamento.

Todos os dados são armazenados nos bancos de dados em formato de Bytes. Essa é a forma como os computadores trabalham, ou sejam, quando irão armazenar a letra A, na realidade, armazenam o código binário “01000001” que a representa.

Quando esse dado precisa ser mostrado, o banco de dados traduz o formato binário gravado, na letra A.

Tipos de Dados

Lower Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00		32	20		64	40	@	96	60	`
1	01	☺	33	21	!	65	41	A	97	61	a
2	02	☻	34	22	"	66	42	B	98	62	b
3	03	♥	35	23	#	67	43	C	99	63	c
4	04	♦	36	24	$	68	44	D	100	64	d
5	05	♣	37	25	%	69	45	E	101	65	e
6	06	♠	38	26	&	70	46	F	102	66	f
7	07	•	39	27	'	71	47	G	103	67	g
8	08	◘	40	28	(	72	48	H	104	68	h
9	09	○	41	29	)	73	49	I	105	69	i
10	0A	◙	42	2A	*	74	4A	J	106	6A	j
11	0B	♂	43	2B	+	75	4B	K	107	6B	k
12	0C	♀	44	2C	,	76	4C	L	108	6C	l
13	0D	♪	45	2D	-	77	4D	K	109	6D	m
14	0E	♫	46	2E	.	78	4E	N	110	6E	n
15	0F	☼	47	2F	/	79	4F	O	111	6F	o
16	10	►	48	30	0	80	50	P	112	70	p
17	11	◄	49	31	1	81	51	Q	113	71	q
18	12	↕	50	32	2	82	52	R	114	72	r
19	13	‼	51	33	3	83	53	S	115	73	s
20	14	¶	52	34	4	84	54	T	116	74	t
21	15	§	53	35	5	85	55	U	117	75	u
22	16	▬	54	36	6	86	56	V	118	76	v
23	17	↨	55	37	7	87	57	W	119	77	w
24	18	↑	56	38	8	88	58	X	120	78	x
25	19	↓	57	39	9	89	59	Y	121	79	y
26	1A	→	58	3A	:	90	5A	Z	122	7A	z
27	1B	←	59	3B	;	91	5B	[	123	7B	{
28	1C	∟	60	3C	<	92	5C	\	124	7C	|
29	1D	↔	61	3D	=	93	5D	]	125	7D	}
30	1E	▲	62	3E	>	94	5E	^	126	7E	~
31	1F	▼	63	3F	?	95	5F	_	127	7F	⌂

Upper Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

- Exemplo de Armazenamento em Bytes - Tabela ASCII

Tipos de Dados

Tipos de dados determinam quais os tipos de valores serão permitidos no armazenamento e os principais tipos são agrupados em categorias conforme mostrado abaixo:

Categorias dos Tipos de Dados	
Numéricos Exatos	Caractere Unicode
Numéricos Aproximados	Binários
Data e Hora	Outros Tipos
Strings de Caractere	

Faculdade IMPACTA TECNOLOGIA

Tipos de Dados

- Numéricos Exatos

Data type	Range	Storage (bytes)
tinyint	0 to 255	1
smallint	-32,768 to 32,767	2
int	2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)	4
Bigint	-2^63 - 2^63-1 (+/- 9 quintillion)	8
bit	1, 0 or NULL	1
decimal/numeric	- 10^38 +1 through 10^38 – 1 when maximum precision is used	5-17
money	-922,337,203,685,477.5808 to 922,337,203,685,477.5807	8
smallmoney	- 214,748.3648 to 214,748.3647	4

Tipos de Dados

- Dados Caractere Non-Unicode

Data Type	Range	Storage
CHAR(n)	1-8000 characters	n bytes, padded
VARCHAR(n)	1-8000 characters	n+2 bytes
VARCHAR(MAX)	1-2^31-1 characters	Actual length + 2

- Dados Caractere Unicode

Data Type	Range	Storage
NCHAR(n)	1-4000 characters	2*n bytes, padded
NVARCHAR(n)	1-4000 characters	(2*n) +2 bytes
NVARCHAR(MAX)	1-2^31-1 characters	(Actual length) * 2 + 2

Tipos de Dados

- Data e Hora

Data Type	Storage (bytes)	Date Range	Accuracy	Recommended Entry Format
DATETIME	8	January 1, 1753 to December 31, 9999	3-1/3 milliseconds	'YYMMDD hh:mm:ss:nnn'
SMALLDATETIME	4	January 1, 1900 to June 6, 2079	1 minute	'YYMMDD hh:mm:ss:nnn'
DATETIME2	6 to 8	January 1, 0001 to December 31, 9999	100 nanoseconds	'YYMMDD hh:mm:ss.nnnnnn'
DATE	3	January 1, 0001 to December 31, 9999	1 day	'YYYY-MM-DD'
TIME	3 to 5		100 nanoseconds	'hh:mm:ss:nnnnnnn'
DATETIMEOFFSET	8 to 10	January 1, 0001 to December 31, 9999	100 nanoseconds	'YY-MM-DD hh:mm:ss:nnnnnnn [+|-]hh:mm'

Tipos de Dados

- Data e Hora

Data Type	Language-Neutral Formats	Examples
DATETIME	'YYYYMMDD hh:mm:ss.nnn' 'YYYY-MM-DDThh:mm:ss.nnn' 'YYYYMMDD'	'20120212 12:30:15.123' '2012-02-12T12:30:15.123' '20120212'
SMALLDATETIME	'YYYYMMDD hh:mm' 'YYYY-MM-DDThh:mm' 'YYYYMMDD'	'20120212 12:30' '2012-02-12T12:30' '20120212'
DATETIME2	'YYYY-MM-DD' 'YYYYMMDD hh:mm:ss.nnnnnnn' 'YYYY-MM-DD hh:mm:ss.nnnnnnn' 'YYYY-MM-DDThh:mm:ss.nnnnnnn' 'YYYYMMDD' 'YYYY-MM-DD'	'20120212 12:30:15.1234567' '2012-02-12 12:30:15.1234567' '2012-02-12T12:30:15.1234567' '20120212' '2012-02-12'
DATE	'YYYYMMDD' 'YYYY-MM-DD'	'20120212' '2012-02-12'
TIME	'hh:mm:ss.nnnnnnn'	'12:30:15.1234567'
DATETIMEOFFSET	'YYYYMMDD hh:mm:ss.nnnnnnn [+|-]hh:mm' 'YYYY-MM-DD hh:mm:ss.nnnnnnn [+|-]hh:mm' 'YYYYMMDD' 'YYYY-MM-DD'	'20120212 12:30:15.1234567 +02:00' '2012-02-12 12:30:15.1234567 +02:00' '20120212' '2012-02-12'

Tipos de Dados

- Numéricos Aproximados

Data Type	Range	Storage (bytes)
float(n)	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308	Depends on value of n, 4 or 8 bytes
real	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38	4

- Binários

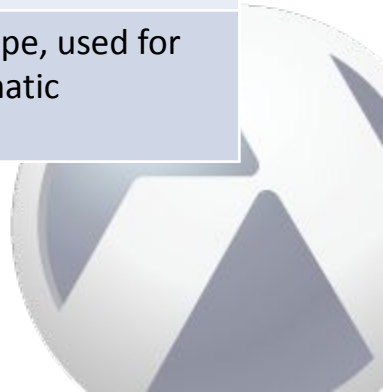
Data Type	Range	Storage (bytes)
binary(n)	1-8000 bytes	n bytes
varbinary(n)	1-8000 bytes	n bytes + 2
varbinary(MAX)	1-2.1 billion (approx) bytes	actual length + 2

Faculdade IMPACTA TECNOLOGIA

Tipos de Dados

- Outros Tipos

Data Type	Range	Storage (bytes)	Remarks
rowversion	Auto-generated	8	Successor type to timestamp
uniqueidentifier	Auto-generated	16	Globally unique identifier (GUID)
xml	0-2 GB	0-2 GB	Stores XML in native hierarchical structure
cursor	N/A	N/A	Not a storage data type
hierarchyid	N/A	Depends on content	Represents position in a hierarchy
sql_variant	0-8000 bytes	Depends on content	Can store data of various data types
table	N/A	N/A	Not a storage data type, used for query and programmatic operations



Data Definition Language

Para persistir os dados em um banco de dados, precisamos armazená-los em objetos do tipo tabela. Comandos de criação, alteração ou eliminação de objetos, fazem parte da categoria de comandos DDL (Data Definition Language).

Nesta aula aprenderemos a sintaxe básica para criação, alteração e eliminação de tabelas e nas próximas aulas iremos aprofundar o assunto, atrelando propriedades nas tabelas, de forma que desempenhem determinadas funcionalidades como verificação de valores, valores padrão e relacionamento entre outras tabelas.

Para iniciar a criação de tabelas, precisaremos definir adequadamente suas colunas de forma que permitam a movimentação de informações segundo seus tipos de dados e obrigatoriedade ou não de informação.

Também veremos como programar para que determinadas colunas sejam preenchidas automaticamente com valores que determinaremos.

Data Definition Language

Sintaxe básica para o comando de criação de tabelas.

Uma atenção especial deve ser dada em que banco de dados estamos criando a estrutura de dados, portanto usamos o comando USE <database> para posicionarmos na base de dados requerida:

```
USE <database>
GO

CREATE TABLE <nome da tabela>
    (
    <nome coluna 1> <tipo da coluna> (<tamanho da coluna>) [NOT NULL]
    , <nome coluna 2> <tipo da coluna> (<tamanho da coluna>) [NOT NULL]
    , ...
    );
```

Data Definition Language

O comando anterior é uma simplificação da cláusula CREATE TABLE, sendo que a mesma cláusula aceita inúmeros parâmetros e objetos associados. Como exemplo do comando mostrado, as cláusulas abaixo criam a tabela Aluno na base de dados FIT:

```
USE FIT
GO

CREATE TABLE Aluno
    (
    Matricula int
    , Nome varchar(20)
    , MeioNome varchar(20)
    , SobreNome varchar(20)
    );
```

Data Definition Language

A cláusula NULL e NOT NULL definem se o preenchimento de determinada coluna é ou não obrigatória. A cláusula NULL é a padrão, ou seja, quando não mencionamos nada como na criação da tabela do slide anterior, os campos PERMITIRÃO valores NULL.

Se quisermos OBRIGATORIAMENTE que um campo seja preenchido, utilizamos o NOT NULL imediatamente após a definição do tipo de dados da coluna:

```
CREATE TABLE Veiculo
    (
    Placa char(8) NOT NULL
    , Marca varchar(20)  NOT NULL
    , NomeProprietario varchar(60)
    );
```

Data Definition Language

Podemos precisar que uma determinada coluna GERE números automaticamente, como por exemplo, número de matrícula.

Os bancos de dados possuem funções específicas para geração de números e podem ser associadas a uma coluna. O SQL Server permite que qualquer tabela tenha uma campo com autonumeração, mas apenas uma coluna da tabela pode receber essa funcionalidade:

IDENTITY (Seed, Increment)

Onde SEED representará o primeiro número a ser gerado pela série e o INCREMENT significa de quanto em quanto os próximos números serão gerados.

Data Definition Language

Perceba que dependendo do tipo de dados, o IDENTITY não se encaixará. Veja alguns exemplos da função:

IDENTITY (Seed, Increment)

IDENTITY ou IDENTITY (1, 1) □ É o padrão da função (qualquer tipo numérico)

IDENTITY (0, 1) □ Qualquer tipo numérico. Inicia no 0 e incrementa de 1 em 1

IDENTITY (-32768, 1) □ Smallint ou maior. Inicia no -32768, -32767, ..., 0, 1, 2, ...

IDENTITY (255, -1) □ Tinyint ou maior. Inicia no 255, 254, 253, ...

IDENTITY (0, 10) □ Tinyint ou maior. Inicia no 0, 10, 20, ...

Data Definition Language

Note que por definição, uma coluna autonumerada será automaticamente colocado como NOT NULL, mesmo não escrevendo explicitamente esta cláusula. O exemplo abaixo mostra a aplicação do IDENTITY:

```
CREATE TABLE Veiculo
    (
    idVeiculo INT NOT NULL  IDENTITY
    , Placa char(8) NOT NULL
    , Marca varchar(20)  NOT NULL
    );
CREATE TABLE Aluno
    (
    Matricula int IDENTITY (500, 1)
    , Nome varchar(20)
    , MeioNome varchar(20)
    , SobreNome varchar(20)
    );
```

Data Definition Language

PRIMARY KEY - Podemos ter uma e apenas uma chave primária em cada tabela. Por definição chave primária possui um número único para todos os registros ou seja, dado um valor correspondente a chave primária de uma tabela, o retorno máximo de registros é uma linha.

A chave primária pode ser simples (apenas uma coluna) ou composta (mais de uma coluna).

A definição da chave primária (primary key) segue o seguinte modelo:

CONSTRAINT <nome da primary key> PRIMARY KEY (coluna1, coluna2, ...)

Data Definition Language

Exemplo:

```
CREATE TABLE Aluno
    (
    Matricula int  NOT NULL IDENTITY (500, 1)
    , Nome varchar(20)
    , MeioNome varchar(20)
    , SobreNome varchar(20)
    , CONSTRAINT pkAluno PRIMARY KEY (Matricula)
    );
```

Data Definition Language

FOREIGN KEY – Chave Estrangeira faz o relacionamento entre uma ou mais coluna de uma tabela com a chave primária ou única de outra tabela. O formato do comando deve referir as colunas de ambas as tabelas e a tabela onde temos a chave primária.

Uma tabela pode ter várias chaves estrangeiras para outras tabelas, representando o relacionamento que possui com cada uma das outras tabelas.

CONSTRAINT <nome da foreign key> FOREIGN KEY (coluna1, coluna2, ...)
REFERENCES <tabela da primary key> (coluna1, coluna2, ...)

Data Definition Language

Exemplo:

```
CREATE TABLE Aluno
	(
	Matricula int not null IDENTITY (500, 1)
	, Nome varchar(20)
	, CONSTRAINT pkAluno PRIMARY KEY (Matricula)
	);

CREATE TABLE Prova
	(
	idProva int NOT NULL  IDENTITY (1, 1)
	, Matricula int NOT NULL
	, Nota decimal(4,2) NOT NULL
	, CONSTRAINT pkProva PRIMARY KEY (idProva)
	, CONSTRAINT fkProva FOREIGN KEY (Matricula)
	  REFERENCES Aluno(Matricula)
	);
```

Data Definition Language

```
CREATE TABLE Aluno
    (
    Matricula int not null IDENTITY (500, 1)
    , Nome varchar(20)
    , CONSTRAINT pkAluno
      PRIMARY KEY (Matricula)
    );
```

```
CREATE TABLE Prova
    (
    idProva int NOT NULL  IDENTITY (1, 1)
    , Matricula int  NOT NULL
    , Nota decimal(4,2) NOT NULL
    , CONSTRAINT pkProva PRIMARY KEY (idProva)
    , CONSTRAINT fkProva FOREIGN KEY (Matricula)
      REFERENCES Aluno(Matricula)
    );
```

Matricula	Nome
500	José
501	Pedro
502	Mario

idProva	Matricula	Nota
1	500	9
2	500	8
3	502	7
4	502	3
5	502	1

Data Definition Language

UNIQUE – Chave Única é semelhante a chave primária, fazendo com que o campo envolvido seja único na tabela. Podemos ter várias chaves únicas em uma tabela, diferentemente da chave primária, onde só podemos ter uma.

Por exemplo, numa tabela de Cliente, podemos ter um campo que é o número do cliente, CPF e RG. Todos os três campos não permitem repetição na tabela. Podemos eleger qualquer um desses campos como chave primária, por exemplo número do cliente. Neste caso o CPF e o RG poderiam ter chaves únicas, já que a tabela só permite uma chave primária. comando deve referir as colunas de ambas as tabelas e a tabela onde temos a chave primária.

CONSTRAINT <nome da unique key> UNIQUE (coluna1, coluna2, ...)

Data Definition Language

Exemplo:

```
CREATE TABLE Cliente
    (
    NumCliente int not null IDENTITY (1, 1)
    , CPF int NOT NULL
    , RG int NOT NULL
    , CONSTRAINT pkCliente PRIMARY KEY (NumCliente)
    , CONSTRAINT uqClienteCPF UNIQUE (CPF)
    , CONSTRAINT uqClienteRG UNIQUE (RG)
    );
```

Data Definition Language

Podemos alterar tabelas depois de criadas, adicionando novos campos, removendo campos existentes ou alterando tipos de dados e configurações de colunas existentes.

O comando básico para isso é o ALTER TABLE. Para adicionar uma coluna, utilizamos a claúsula ADD. Para eliminar, DROP COLUMN. Para alterar, ALTER COLUMN. Estes comandos só mexem em uma coluna por vez, ou seja, se quisermos adicionar três colunas, teremos que emitir um comando para cada coluna.

ALTER TABLE <nome da tabela> ADD <nome da coluna> <tipo de dados>

ALTER TABLE <nome da tabela> ADD CONSTRAINT <nome da constraint> ...

ALTER TABLE <nome da tabela> ALTER COLUMN <nome da coluna> <tipo de dados>

ALTER TABLE <nome da tabela> DROP COLUMN <nome da coluna>

ALTER TABLE <nome da tabela> DROP CONSTRAINT <nome da constraint>

Data Definition Language

Exemplos:

```
ALTER TABLE Cliente ADD Nome varchar(30) NOT NULL
ALTER TABLE Cliente ADD SobreNome varchar(30) NOT NULL
```

```
ALTER TABLE Cliente ALTER COLUMN Nome varchar(50) NULL
```

```
ALTER TABLE Cliente DROP COLUMN SobreNome
```

Data Definition Language

Para eliminar colunas existentes em uma tabela, usamos o comando DROP TABLE.

DROP TABLE <nome da tabela1>, <nome da tabela2>, ..., <nome da tabela n>

Note que se uma tabela possui uma chave primária e esta chave participa de relacionamentos com outras tabelas como chave estrangeira, não será permitida a eliminação da tabela que contém a chave primária, ou seja, neste caso, precisamos eliminar as tabelas que mencionam essa chave primária, para depois, conseguirmos eliminar a tabela de chave primária.

Como exemplo, onde temos a tabela Aluno (chave primária), relacionada com a tabela Prova (chave estrangeira), se quisermos eliminar a tabela Aluno, precisamos primeiro, eliminar a tabela Prova, ou retirar as contraints destes objetos, liberando a “amarração” entre eles.

DROP TABLE Prova, Aluno

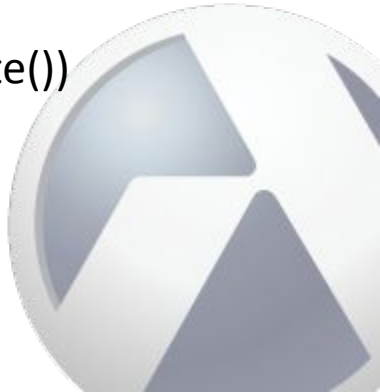
Data Definition Language

Não podem fazer referência a uma outra coluna da tabela, ou a outras tabelas, exibições ou procedimentos armazenados. As definições DEFAULT serão removidas quando a tabela for descartada.

<nome da coluna> <tipo de dados> CONSTRAINT <nome do default>
DEFAULT (<valor, texto, data,
função escalar>)

Exemplos:

MBAExterior VARCHAR(100) CONSTRAINT dfTextoNA DEFAULT 'Não'
Desconto DECIMAL(9, 2) CONSTRAINT dfDesconto DEFAULT 0
DataVenda DATE NOT NULL CONSTRAINT dfDataVenda DEFAULT (getdate())



Data Definition Language

Exemplos

Para Criação:

```
CREATE TABLE Venda
    (
    DataVenda date not null CONSTRAINT dfDataVenda DEFAULT (getdate())
    , Quantidade smallint not null CONSTRAINT dfQtd DEFAULT (1)
    , NumeroCliente int not null
    , CONSTRAINT pkVenda PRIMARY KEY (DataVenda)
    , CONSTRAINT fkVenda FOREIGN KEY (NumeroCliente)
      REFERENCES Cliente(idCliente)
    );
```

Para alteração:

```
ALTER TABLE Venda ADD CONSTRAINT dfDataVenda (getdate()) FOR DataVenda
ALTER TABLE Venda ADD CONSTRAINT dfQtd DEFAULT (1) FOR Quantidade
```

Data Definition Language

Os tipos de dados incluem uma restrição ao preenchimento dos dados em uma coluna, assim, quando definimos uma coluna como TINYINT, sabemos que os valores permitidos irão de 0 a 255.

No mesmo exemplo anterior poderíamos querer que os valores permitidos, além de serem numéricos, pudessem assumir somente os valores de 18 a 90. Esse tipo de restrição pode ser conseguida aplicando REGRAS, que é o significado da cláusula CHECK.

Uma coluna pode ter qualquer número de restrições CHECK e os critérios podem incluir diversas expressões lógicas combinadas com AND e OR. Várias restrições CHECK são validadas na ordem de criação.

Data Definition Language

A avaliação do critério de pesquisa deve usar uma expressão Booleana (true/false) como base e não pode fazer referência a outra tabela.

A restrição CHECK no nível de coluna pode fazer referência somente à coluna restrita.

Restrições CHECK oferecem a mesma função de validação dos dados durante instruções INSERT e UPDATE.

Se existirem uma ou mais restrições CHECK para uma coluna, todas as restrições serão avaliadas.

CONSTRAINT <nome da regra> CHECK (<coluna com expressão booleana>)

Data Definition Language

Exemplo:

```
CONSTRAINT ckIdade CHECK (Idade <= 100)

CONSTRAINT ckTaxa CHECK (Taxa >= 1 and Taxa <= 5)

CONSTRAINT CK_emp_id CHECK (emp_id LIKE '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]' OR emp_id LIKE '[A-Z]-[A-Z][1-9][0-9][0-9][0-9][0-9][FM]')

CONSTRAINT CK_emp_id CHECK (emp_id IN ('1389', '0736', '0877', '1622', '1756') OR emp_id LIKE '99[0-9][0-9]')
```

Data Definition Language

Exemplo:

```
create table Cliente
(
	idCliente smallint identity(-32767, 1)
	, Telefone VARCHAR(14)
		, DataEntrada datetime
	, Idade tinyint not null,
	, constraint ckIdade CHECK (Idade between 18 and 90)
	, constraint ckTelefone CHECK
			(
			Telefone LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]' OR
		Telefone LIKE '([0-9][0-9][0-9]) [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9])'
		)
)
```

Data Definition Language

As Constraints Primary key, Foreing Key, Unique, Default e Check, podem ser incluídas ou retiradas de uma tabela utilizando os mesmos comandos mencionados anteriormente. Exemplo:

```
ALTER TABLE Cliente Drop Constraint ckIdade

ALTER TABLE Cliente Add Constraint  ckIdade CHECK (idade between 18 and 90)

ALTER TABLE Venda Alter Column DataVenda date NOT NULL
ALTER TABLE Venda ADD Constraint dfDtVenda DEFAULT (getdate()) FOR DataVenda
```

Faculdade IMPACTA TECNOLOGIA

Obrigado!

Aula Gravada por:

Prof. Msc. Gustavo Bianchi Maia

gustavo.maia@faculdadeimpacta.com

Material criado e oferecido por :

Prof. Sand Jacques Onofre