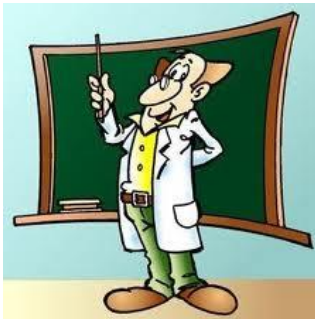




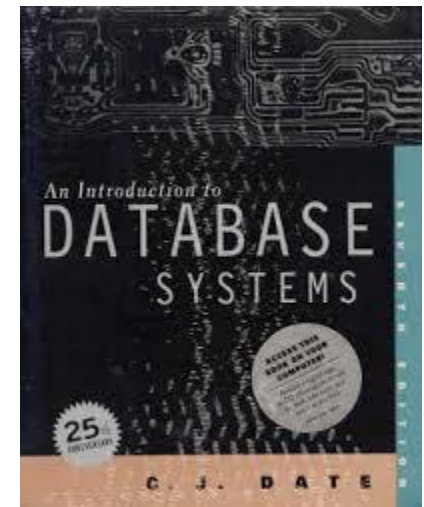
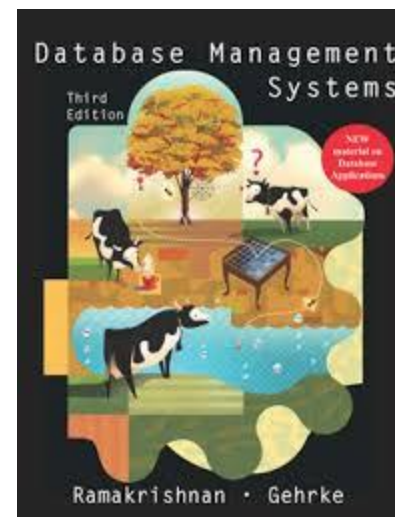
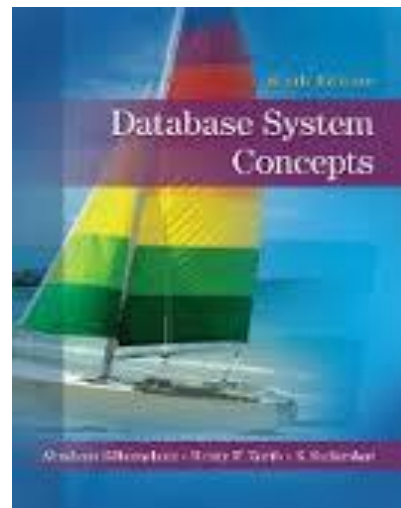
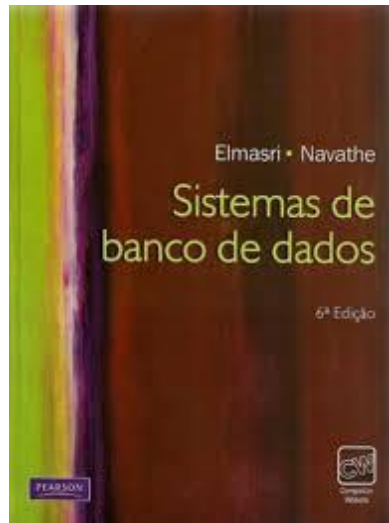
Unidade 3 – SQL Básica – DDL



Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com



Bibliografia





Introdução

- ◆ A linguagem SQL pode ser considerada um dos principais motivos para o sucesso dos bancos de dados relacionais comerciais.





Linguagem SQL

- ◆ SQL – Structured Query Language
- ◆ Originalmente, chamada SEQUEL (Structured English Query Language) e foi criada e implementada pela IBM Research, como interface para o sistema R.





Linguagem SQL

- ◆ Tornou-se a linguagem padrão para bancos de dados relacionais.
- ◆ Sendo padrão, facilita migração de aplicações entre outros modelos de DBMS (rede e hierárquico) e entre outros DBMS relacionais.
- ◆ Cada DBMS relacional tem suas extensões especializadas, mas a conversão entre ambientes é facilitada quando se usa apenas recursos que fazem parte do padrão (núcleo).





Padrões da Linguagem

- Esforço conjunto entre ANSI (American National Standards Institute) e ISO (International Standards Organization).
- SQL-86 ou SQL1 em 1986
- SQL-92 ou SQL2
- SQL3 em 1999
- SQL:2003
- SQL:2006
- SQL-2008





Características

- ◆ Linguagem Declarativa;
- ◆ Instruções para definição de dados, consultas e atualizações (é uma DDL e uma DML);
- ◆ Especificação núcleo – implementado em todos os DBMS que suportam o padrão SQL;
- ◆ Mais extensões especializadas.





Terminologia

- ◆ **Tabela** significa **Relação**
- ◆ **Linha** significa **Tupla**
- ◆ **Coluna** significa **Atributo**





Esquema em SQL

- ◆ Primeiras versões não consideravam o conceito de esquema;
- ◆ Conceito incorporado na versão SQL-92;
- ◆ Identificado por um nome de esquema e descritores;
- ◆ Sintaxe: CREATE SCHEMA;
- ◆ Esquemas incluem tabelas, restrições, views, etc.





Catálogo

- ◆ Uma coleção nomeada de esquemas em um ambiente **SQL**;
- ◆ Um ambiente **SQL** é basicamente uma instalação de um SGBDR compatível com **SQL**;
- ◆ Contém um esquema especial chamado `INFORMATION_SCHEMA`, que oferece informações sobre todos os esquemas no catálogo.





CREATE TABLE

- ◆ Usado para se especificar uma nova relação, dando-lhe um nome e especificando seus atributos e restrições iniciais (por exemplo, **NOT NULL**).
- ◆ As restrições de chave, integridade de entidade e integridade referencial podem também ser especificadas na instrução ALTER TABLE.

CREATE TABLE EMPRESA.FUNCIONARIO ...

ou

CREATE TABLE FUNCIONARIO ...





CREATE TABLE

- ◆ As relações declaradas pelo comando **CREATE TABLE** são chamadas de Tabelas de Base (ou Relações de Base). Isso significa que a relação e suas tuplas são armazenadas em arquivo pelo **DBMS**.
- ◆ As relações de Base são distintas das relações virtuais, criadas pelo comando **CREATE VIEW**, que podem ou não corresponder a um arquivo físico real.





CREATE TABLE

CREATE TABLE FUNCIONARIO

```
(Pnome      VARCHAR(15)      NOT NULL,
Minicial    CHAR,
Unome      VARCHAR(15)      NOT NULL,
Cpf        CHAR(11),        NOT NULL,
Datanasc    DATE,
Endereço    VARCHAR(30),
Sexo        CHAR,
Salario     DECIMAL(10,2),
Cpf_supervisor CHAR(11),    NOT NULL,
Dnr         INT
PRIMARY KEY (Cpf),
FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf),
FOREIGN KEY (Dnr) REFERENCES DEPARTAMENTO(Dnumero));
```

CREATE TABLE DEPARTAMENTO

```
( Dnome      VARCHAR(15)      NOT NULL,
Dnumero     INT              NOT NULL,
Cpf_gerente CHAR(11),        NOT NULL,
Data_inicio_gerente DATE,
PRIMARY KEY (Dnumero),
UNIQUE (Dnome),
FOREIGN KEY (Cpf_gerente) REFERENCES FUNCIONARIO(Cpf);
```

CREATE TABLE LOCALIZACAO_DEP

```
( Dnumero     INT              NOT NULL,
Dlocal        VARCHAR(15)      NOT NULL,
PRIMARY KEY (Dnumero, Dlocal),
FOREIGN KEY (Dnumero) REFERENCES DEPARTAMENTO(Dnumero));
```

CREATE TABLE PROJETO

```
( Projnome    VARCHAR(15)      NOT NULL,
Projnumero    INT              NOT NULL,
Projlocal     VARCHAR(15),
Dnum          INT              NOT NULL,
PRIMARY KEY (Projnumero),
UNIQUE (Projnome),
FOREIGN KEY (Dnum) REFERENCES DEPARTAMENTO(Dnumero));
```

CREATE TABLE TRABALHA_EM

```
( Fcpf        CHAR(9)          NOT NULL,
Pnr          INT              NOT NULL,
Horas        DECIMAL(3,1)      NOT NULL,
PRIMARY KEY (Fcpf, Pnr),
FOREIGN KEY (Fcpf) REFERENCES FUNCIONARIO(Cpf),
FOREIGN KEY (Pnr) REFERENCES PROJETO(Projnumero);
```

CREATE TABLE DEPENDENTE

```
( Fcpf        CHAR(11),        NOT NULL,
Nome_dependente VARCHAR(15)    NOT NULL,
Sexo         CHAR,
Datanasc     DATE,
Parentesco   VARCHAR(8),
PRIMARY KEY (Fcpf, Nome_dependente),
FOREIGN KEY (Fcpf) REFERENCES FUNCIONARIO(Cpf);
```



Comando CREATE TABLE



- ✓ Algumas chaves estrangeiras podem causar erros, especificadas por referências circulares ou porque dizem respeito a uma tabela que ainda não foi criada.
- ✓ Por exemplo, a chave estrangeira **Dnr** na tabela **FUNCIONARIO** se refere à tabela **DEPARTAMENTO**, que ainda não foi criada.
- ✓ Por exemplo, a chave estrangeira **Cpf_Supervisor** na tabela **FUNCIONARIO** é uma referência circular, pois se refere à própria tabela.

```
CREATE TABLE FUNCIONARIO
(Pnome      VARCHAR(15)      NOT NULL,
Minicial    CHAR,
Unome       VARCHAR(15)      NOT NULL,
Cpf         CHAR(11),        NOT NULL,
Datanasc    DATE,
Endereço    VARCHAR(30),
Sexo        CHAR,
Salario     DECIMAL(10,2),
Cpf_supervisor CHAR(11),    NOT NULL,
Dnr         INT
PRIMARY KEY (Cpf),
FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf),
FOREIGN KEY (Dnr) REFERENCES DEPARTAMENTO(Dnumero) );
```

Pode-se omiti-las inicialmente e depois defini-las com o comando **ALTER TABLE**;





Tipos de Dados Numéricos

- ✓ Incluem números inteiros: **INTEGER** ou **INT** e **SMALLINT**
- ✓ Números de ponto flutuante (reais): **FLOAT** ou **REAL** e **DOUBLE PRECISION**
- ✓ O formato dos números pode ser declarado usando **DECIMAL (i,j)** , onde **i**, a precisão, é o número total de dígitos decimais e **j**, a escala, é o número de dígitos após o ponto decimal.
- ✓ O valor padrão para a escala é zero, e para a precisão, é definido pela implementação.





Tipos de Dados de Cadeia de Caracteres

- ✓ Tamanho fixo: **CHAR**(n) ou **CHARACTER**(n)
- ✓ Tamanho variável: **VARCHAR**(n) ou **CHAR VARYING**(n) ou **CHARACTER VARYING**(n)
- ✓ Literais de cadeias de caracteres são definidos entre aspas simples (apóstrofo) e são case sensitives.
- ✓ Alinhamento à **esquerda**. Exemplo: 'Silva' para um atributo do tipo CHAR(10) será armazenado como 'Silva '.
- ✓ Cadeias podem ser concatenadas pelo operador || (barra vertical dupla).
- ✓ Comparação de cadeias de caracteres feita por meio de ordem alfabética (lexicográfica). Por exemplo, 'xyz' é menor que 'XYZ'.





Tipos de Dados de Cadeia de Bits

- ✓ Tamanho fixo: **BIT** (n)
- ✓ Tamanho variável: **BIT VARYING** (n) , onde **n** é o número máximo de bits.
- ✓ O valor padrão para n, o tamanho de uma cadeia de caracteres bits, é 1.
- ✓ Literais de cadeia de bits são colocados entre apóstrofes, mas precedidos por um **B**. Exemplo: **B'10101'**.
- ✓ Outro tipo de dados de cadeia de bits de tamanho variável, chamado **BINARY LARGE OBJECT** OU **BLOB**, também está disponível para especificar colunas que possuem grandes valores binários, como imagens.





Tipo de dados Booleano

- ✓ Valores **TRUE** ou **FALSE** ou **NULL**;
- ✓ Devido a presença de valores **NULL** (nulos), uma lógica de três valores é utilizada, de modo que um terceiro valor possível para um tipo de dados booleano é **UNKNOWN** (indefinido).





Tipo de dados Date e TIME

- ✓ O tipo de dados **DATE** possui 10 posições e seus componentes são DAY (dia), MONTH (mês) e YEAR (ano) no formato DD-MM-YYYY.
- ✓ O tipo de dado **TIME** tem oito posições, com os componentes HOUR (hora), MINUTE (minuto) e SECOND (segundo), no formato HH:MM:SS.
- ✓ Literais são representados por cadeias com apóstrofos precedidos pela palavra-chave **DATE** ou **TIME**. Por exemplo, DATE'27-09-2008' ou TIME'09:12:47'.
- ✓ Além disso, um tipo de dado **TIME(i)**, onde i é chamado de precisão em segundos fracionários de tempo, especifica i+1 posições adicionais para **TIME**.



Tipo de dados Date e TIME



```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( DATA_NASC DATE NOT NULL, DATA_MORTE DATE, HORA_MORTE TIME(5) );
```

```
INSERT INTO TAB_TESTE VALUES (DATE'2015-10-31' , DATE'2015-12-30', TIME'09:12:47' );
```

```
INSERT INTO TAB_TESTE VALUES (DATE'2009-12-23' , DATE'2014-10-30', TIME'23:59:59.00000' );
```

```
SELECT * FROM TAB_TESTE;
```

	DATA_NASC	DATA_MORTE	HORA_MORTE
	2015-10-31	2015-12-30	09:12:47.00000
	2009-12-23	2014-10-30	23:59:59.00000



Somar/Subtrair datas em MySQL

✓ **date_add** (Adiciona valores a uma data)

✓ **date_sub** (Reduz valores a uma data)

✓ Sintaxe:

- **date_add** (date, INTERVAL **expr** **unit**);
- **date_sub** (date, INTERVAL **expr** **unit**);

✓ **expr**: argumento da expressão. Pode começar com (-) nos casos de Subtração

✓ **unit**: argumento chave que indica a unidade de medida que será usada (anos, dias, ...)



Manipulação de dados em MySQL

- ✓ Adicionando 1 ano sob a data atual:

```
SELECT now() , date_add(now() , INTERVAL 1 YEAR )
```

now()	date_add(now() , INTERVAL 1 YEAR)
2016-03-10 01:40:39	2017-03-10 01:40:39

- ✓ Adicionando 1 mês sob a data atual:

```
SELECT now() , date_add(now() , INTERVAL 1 MONTH )
```

now()	date_add(now() , INTERVAL 1 MONTH)
2016-03-10 01:41:20	2016-04-10 01:41:20





Manipulando datas em MySQL

- ✓ Adicionando 1 dia sob a data atual:

```
SELECT date_add(now() , INTERVAL 1 DAY )
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	now()	date_add(now() , INTERVAL 1 DAY)		
▶	2016-03-10 01:42:22	2016-03-11 01:42:22		

- ✓ Adicionando 1 hora sob a data atual:

```
SELECT date_add(now() , INTERVAL 1 HOUR )
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	now()	date_add(now() , INTERVAL 1 HOUR)		
▶	2016-03-10 01:43:08	2016-03-10 02:43:08		





Tipo Timestamp

- ✓ Inclui os campos **DATE** e **TIME**, podendo ser aplicado como **log** contendo informação da data e hora relativas à inserção de uma tupla em uma relação.

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( NOME CHAR(30) NOT NULL, DATA_STAMP TIMESTAMP );
```

```
INSERT INTO TAB_TESTE VALUES ('Antonio Jose Silva', NOW() );
```

```
INSERT INTO TAB_TESTE VALUES ( 'Paulo de Souza Alves', NOW() );
```

```
SELECT * FROM TAB_TESTE;
```



Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	NOME	DATA_STAMP			
▶	Antonio Jose Silva	2016-03-10 02:07:08			
	Paulo de Souza Alves	2016-03-10 02:07:08			



Restrição NOT NULL

- ✓ SQL permite **NULL** como sendo o valor de um atributo.
- ✓ Se o valor **NULL** não for permitido para um determinado atributo (informação obrigatória) pode-se especificar o valor **NOT NULL**.
- ✓ Isso sempre é especificado de maneira implícita para os atributos que fazem parte da **Chave Primária (PK)** de cada relação, mas pode ser especificado para quaisquer outros atributos.



Restrição NOT NULL

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( idALUNO INTEGER NOT NULL, NOME CHAR(30) );
```

```
INSERT INTO TAB_TESTE VALUES (10, 'Antonio Jose Silva' );
```

```
INSERT INTO TAB_TESTE VALUES ( 10, 'Paulo de Souza Alves' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
SELECT * FROM TAB_TESTE;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	idALUNO	NOME			
▶	10	Antonio Jose Silva			
	10	Paulo de Souza Alves			
	30	NULL			



Restrição NOT NULL

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( idALUNO INTEGER NOT NULL, NOME CHAR(30) );
```

```
INSERT INTO TAB_TESTE VALUES (10, 'Antonio Jose Silva' );
```

```
INSERT INTO TAB_TESTE VALUES ( 10, 'Paulo de Souza Alves' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
DESCRIBE TAB_TESTE
```

Result Grid				
Filter Rows:				
	Field	Type	Null	Key
▶	idALUNO	int(11)	NO	
	NOME	char(30)	YES	



Restrição NOT NULL

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( idAluno INTEGER NOT NULL, NOME CHAR(30),  
PRIMARY KEY (idAluno) );
```

```
INSERT INTO TAB_TESTE VALUES (10, 'Antonio Jose Silva' );
```

```
INSERT INTO TAB_TESTE VALUES ( 10, 'Paulo de Souza Alves' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
DESCRIBE TAB_TESTE;
```



Error Code: Duplicate entry '10' for key 'PRIMARY'



Restrição NOT NULL

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( idAluno INTEGER NOT NULL, NOME CHAR(30),  
PRIMARY KEY (idAluno) );
```

```
INSERT INTO TAB_TESTE VALUES (10, 'Antonio Jose Silva' );
```

```
INSERT INTO TAB_TESTE VALUES ( 20, 'Paulo de Souza Alves' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
SELECT * FROM TAB_TESTE;
```

Result Grid			Filter Rows:		Edit:		Export/Import:	
	idAluno	NOME						
▶	10	Antonio Jose Silva						
	20	Paulo de Souza Alves						
	30	NULL						
★	NULL	NULL						



Restrição NOT NULL

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( idAluno INTEGER NOT NULL, NOME CHAR(30),  
PRIMARY KEY (idAluno) );
```

```
INSERT INTO TAB_TESTE VALUES (10, 'Antonio Jose Silva' );
```

```
INSERT INTO TAB_TESTE VALUES ( 20, 'Paulo de Souza Alves' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
DESCRIBE TAB_TESTE;
```

Result Grid				
Filter Rows:				
	Field	Type	Null	Key
▶	idAluno	int(11)	NO	PRI
	NOME	char(30)	YES	



Restrição NOT NULL

```
DROP TABLE TAB_TESTE;
```

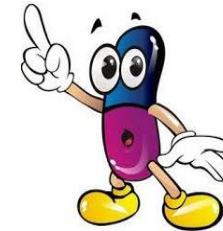
```
CREATE TABLE TAB_TESTE ( idAluno INTEGER, NOME CHAR(30),  
PRIMARY KEY (idAluno) );
```

```
INSERT INTO TAB_TESTE VALUES (10, 'Antonio Jose Silva' );
```

```
INSERT INTO TAB_TESTE VALUES ( 20, 'Paulo de Souza Alves' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
DESCRIBE TAB_TESTE;
```



NOT NULL é default para
campos Primary Key

Result Grid				
Filter Rows:				
	Field	Type	Null	Key
▶	idAluno	int(11)	NO	PRI
	NOME	char(30)	YES	



Restrição NOT NULL

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE ( idAluno INTEGER, NOME CHAR(30),  
PRIMARY KEY (idAluno) );
```

```
INSERT INTO TAB_TESTE VALUES (NULL, 'Antonio Jose Silva' );
```

```
INSERT INTO TAB_TESTE VALUES ( 20, 'Paulo de Souza Alves' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
DESCRIBE TAB_TESTE;
```



Error Code: Column idAluno cannot be null





Cláusula Default para atributo

- ✓ É possível definir-se um valor padrão para um atributo anexando-se a cláusula **DEFAULT** <valor> a uma definição de atributo.
- ✓ Esse valor será atribuído ao atributo, quando um valor explícito não for fornecido.



Cláusula Default para atributo

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (  
    idAluno INTEGER,  
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME',  
    PRIMARY KEY (idAluno) );
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (10 );
```

```
INSERT INTO TAB_TESTE (idAluno, NOME) VALUES ( 20, 'PAULO DE SOUZA ALVES' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (40);
```

```
SELECT * FROM TAB_TESTE;
```

Result Grid		Filter Rows:	Edit:	Export/Import
	idAluno	NOME		
▶	10	ALUNO SEM NOME		
	20	PAULO DE SOUZA ALVES		
	30	NULL		
	40	ALUNO SEM NOME		



Cláusula Default para atributo

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (
    idAluno INTEGER,
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME',
    PRIMARY KEY (idAluno) );
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (10 );
```

```
INSERT INTO TAB_TESTE (idAluno, NOME) VALUES ( 20, 'PAULO DE SOUZA ALVES' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (40);
```

```
DESCRIBE TAB_TESTE;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Contents:

	Field	Type	Null	Key	Default	Extra
▶	idAluno	int(11)	NO	PRI	NULL	
	NOME	char(30)	YES		ALUNO SEM NOME	



Restrição de Chave Primária

- ✓ A cláusula **PRIMARY KEY** especifica um ou mais atributos que compõem a chave primária de uma relação.
- ✓ Se uma chave primária tiver um único atributo, a cláusula pode acompanhar o atributo diretamente.
- ✓ Exemplo:

idAluno **INTEGER PRIMARY KEY NOT NULL**





Restrição de Chave Primária

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (  
    idAluno INTEGER PRIMARY KEY NOT NULL ,  
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' );
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (10 );
```

```
INSERT INTO TAB_TESTE (idAluno, NOME) VALUES ( 20, 'PAULO DE SOUZA ALVES' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (40);
```

```
SELECT * FROM TAB_TESTE;
```

Result Grid			Filter Rows:	Edit:	Export/Impc
	idAluno	NOME			
▶	10	ALUNO SEM NOME			
	20	PAULO DE SOUZA ALVES			
	30	NULL			
	40	ALUNO SEM NOME			



Restrição de Chave Primária

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (
    idAluno INTEGER PRIMARY KEY NOT NULL ,
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' );
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (10 );
```

```
INSERT INTO TAB_TESTE (idAluno, NOME) VALUES ( 20, 'PAULO DE SOUZA ALVES' );
```

```
INSERT INTO TAB_TESTE VALUES ( 30, NULL);
```

```
INSERT INTO TAB_TESTE (idAluno) VALUES (40);
```

```
DESCRIBE TAB_TESTE;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Contents:

	Field	Type	Null	Key	Default	Extra
▶	idAluno	int(11)	NO	PRI	NULL	
	NOME	char(30)	YES		ALUNO SEM NOME	



Chave Primária Composta

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (
    idAluno INTEGER ,
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' ,
    CPF CHAR(14),
    PRIMARY KEY (idAluno, CPF) );
```

```
INSERT INTO TAB_TESTE (idAluno,CPF) VALUES (10, '111.222.333-44');
```

```
INSERT INTO TAB_TESTE (idAluno, CPF) VALUES (10, '657.342.469-37');
```

```
SELECT * FROM TAB_TESTE;
```

Result Grid				Filter Rows:	Edit:	Export/Import:	
	idAluno	NOME	CPF				
	10	ALUNO SEM NOME	111.222.333-44				
	10	ALUNO SEM NOME	657.342.469-37				



Chave Primária Composta

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (
    idAluno INTEGER ,
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' ,
    CPF CHAR(14),
    PRIMARY KEY (idAluno, CPF) );
```

```
INSERT INTO TAB_TESTE (idAluno,CPF) VALUES (10, '111.222.333-44');
```

```
INSERT INTO TAB_TESTE (idAluno, CPF) VALUES (10, '657.342.469-37');
```

```
DESCRIBE TAB_TESTE;
```

Result Grid						
		Filter Rows:			Export:	Wrap Cell Content:
	Field	Type	Null	Key	Default	Extra
▶	idAluno	int(11)	NO	PRI	NULL	
	NOME	char(30)	YES		ALUNO SEM NOME	
	CPF	char(14)	NO	PRI	NULL	



Chave Primária Composta

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (  
    idAluno INTEGER ,  
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' ,  
    CPF CHAR(14),  
    PRIMARY KEY (idAluno, CPF) );
```

```
INSERT INTO TAB_TESTE (idAluno,CPF) VALUES (10, '111.222.333-44');
```

```
INSERT INTO TAB_TESTE (idAluno, CPF) VALUES (10, '111.222.333-44');
```

```
DESCRIBE TAB_TESTE;
```



Error Code: Duplicate entry '10 - 111.222.333-44' for key 'PRIMARY'





Restrição de Chave Única

- ✓ A cláusula **UNIQUE** especifica chaves alternativas (secundárias).
- ✓ A cláusula **UNIQUE** também pode ser especificada diretamente para uma chave secundária se esta for um único atributo.





Restrição de Chave Única

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (  
    idAluno INTEGER PRIMARY KEY ,  
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' ,  
    CPF CHAR(14) UNIQUE );
```

```
INSERT INTO TAB_TESTE (idAluno,CPF) VALUES (10, '111.222.333-44');
```

```
INSERT INTO TAB_TESTE (idAluno, CPF) VALUES (20, '111.222.333-44');
```

```
DESCRIBE TAB_TESTE;
```



Error Code: Duplicate entry '111.222.333-44' for key 'CPF'



Restrição de Chave Única

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (  
    idAluno INTEGER PRIMARY KEY ,  
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' ,  
    CPF CHAR(14) UNIQUE );
```

```
INSERT INTO TAB_TESTE (idAluno,CPF) VALUES (10, '657.789.123-52');
```

```
INSERT INTO TAB_TESTE (idAluno, CPF) VALUES (20, '111.222.333-44');
```

```
SELECT * FROM TAB_TESTE;
```

Result Grid

Filter Rows:

Edit:

Export/Import:

	idAluno	NOME	CPF
▶	10	ALUNO SEM NOME	657.789.123-52
	20	ALUNO SEM NOME	111.222.333-44
★	NULL	NULL	NULL



Restrição de Chave Única

```
DROP TABLE TAB_TESTE;
```

```
CREATE TABLE TAB_TESTE (
    idAluno INTEGER PRIMARY KEY ,
    NOME CHAR(30) DEFAULT 'ALUNO SEM NOME' ,
    CPF CHAR(14) UNIQUE );
```

```
INSERT INTO TAB_TESTE (idAluno,CPF) VALUES (10, '657.789.123-52');
```

```
INSERT INTO TAB_TESTE (idAluno, CPF) VALUES (20, '111.222.333-44');
```

```
DESCRIBE TAB_TESTE;
```

Result Grid Filter Rows: Export: Wrap Cell Content:						
	Field	Type	Null	Key	Default	Extra
▶	idAluno	int(11)	NO	PRI	NULL	
	NOME	char(30)	YES		ALUNO SEM NOME	
	CPF	char(14)	YES	UNI	NULL	



Restrição de Integridade Referencial

- ✓ Especificada pela cláusula **FOREIGN KEY** (chave estrangeira)
- ✓ Uma restrição de integridade referencial pode ser violada quando tuplas são inseridas ou excluídas, ou quando um valor de atributo de chave estrangeira ou chave primária for modificado.
- ✓ Nesses casos, a ação padrão que SQL toma para uma violação é rejeitar a operação de atualização que causará uma violação, o que é conhecido por **RESTRICT**.
- ✓ Porém, pode-se se empregar uma ação alternativa para ser tomada, conectando-se uma cláusula de ação de disparo referencial a qualquer chave estrangeira. As opções incluem **SET NULL**, **CASCADE** e **SET DEFAULT**. Essas opções são qualificadas nas operações **ON DELETE** ou **ON UPDATE**.



Regra de Chave Estrangeira (Foreign Key)

- ✓ A chave estrangeira ocorre quando um atributo de uma relação (**FK**) for chave primária em outra relação (**PK**).
- ✓ Em outras palavras, sempre que houver o relacionamento 1:N entre duas tabelas, a tabela 1 receberá a chave primária e a tabela N receberá a chave estrangeira.
- ✓ Os atributos de FK e PK têm o mesmo domínio. Um valor de FK deve existir em PK, ou ser NULL.

DEPARTAMENTO

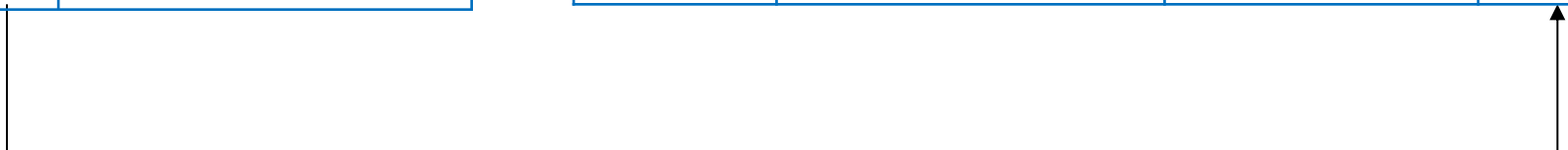
PK

<u>idDepto</u>	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

FK

<u>idFunc</u>	NomeFunc	CPF	idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40





Regra de Chave Estrangeira (Foreign Key)

- ✓ A chave estrangeira ocorre quando um atributo de uma relação (**FK**) for chave primária em outra relação (**PK**).
- ✓ Em outras palavras, sempre que houver o relacionamento 1:N entre duas tabelas, a tabela 1 receberá a chave primária e a tabela N receberá a chave estrangeira.
- ✓ Os atributos de FK e PK têm o mesmo domínio. Um valor de FK deve existir em PK, ou ser NULL.

DEPARTAMENTO

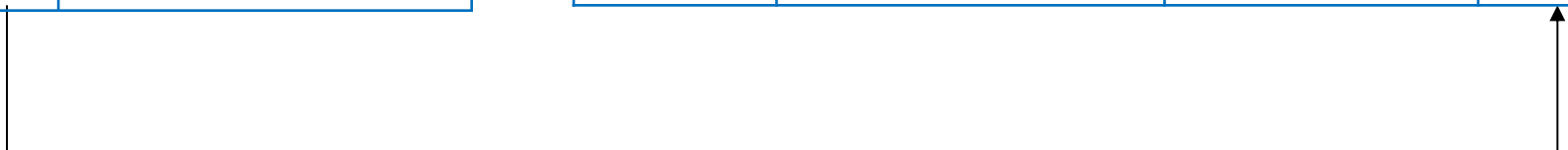
PK

<u>idDepto</u>	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK**FK**

<u>idFunc</u>	NomeFunc	CPF	idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40





Restrição de Integridade Referencial

DROP TABLE DEPARTAMENTO;

CREATE TABLE DEPARTAMENTO (

idDepto **INTEGER PRIMARY KEY** ,
NomeDepto **CHAR** (30));

DEPARTAMENTO

PK

idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK**FK**

idFunc	NomeFunc	CPF	idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40

INSERT INTO DEPARTAMENTO **VALUES** (10, 'Financeiro');

INSERT INTO DEPARTAMENTO **VALUES** (20, 'Compras');

INSERT INTO DEPARTAMENTO **VALUES** (40, 'Vendas');

INSERT INTO DEPARTAMENTO **VALUES** (55, 'Financeiro');

SELECT * FROM DEPARTAMENTO;

Result Grid			Filter Rows:	Edit:
	idDepto	NomeDepto		
▶	10	Financeiro		
	20	Compras		
	40	Vendas		
	55	Financeiro		



Restrição de Integridade Referencial

DROP TABLE DEPARTAMENTO;

CREATE TABLE DEPARTAMENTO (

idDepto **INTEGER PRIMARY KEY** ,
NomeDepto **CHAR** (30));

DEPARTAMENTO

PK idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK idFunc	NomeFunc	CPF	FK idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40

INSERT INTO DEPARTAMENTO **VALUES** (10, 'Financeiro');

INSERT INTO DEPARTAMENTO **VALUES** (20, 'Compras');

INSERT INTO DEPARTAMENTO **VALUES** (40, 'Vendas');

INSERT INTO DEPARTAMENTO **VALUES** (55, 'Financeiro');

DESCRIBE DEPARTAMENTO;

Field	Type	Null	Key	Default	Extra
idDepto	int(11)	NO	PRI	NULL	
NomeDepto	char(30)	YES		NULL	



Restrição de Integridade Referencial

DROP TABLE FUNCIONARIO;

CREATE TABLE FUNCIONARIO (

idFunc **INTEGER PRIMARY KEY**,

NomeFunc **CHAR** (30),

CPF **CHAR** (14) ,

idDepto **INTEGER** ,

FOREIGN KEY (idDepto) **REFERENCES** DEPARTAMENTO(idDepto));

DEPARTAMENTO

PK idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK idFunc	NomeFunc	CPF	FK idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40



INSERT INTO FUNCIONARIO **VALUES** (3456, 'Paulo de Souza Alves', '345.987.123-98',10);




INSERT INTO FUNCIONARIO **VALUES** (9872, 'Jose da Silva', '987.243.098-01',10);

INSERT INTO FUNCIONARIO **VALUES** (4680, 'Angela Silva Medeiros', '567.982.045-27',40);

SELECT * FROM FUNCIONARIO;

Result Grid



Filter Rows:

Edit:




	idFunc	NomeFunc	CPF	idDepto
▶	3456	Paulo de Souza Alves	345.987.123-98	10
	4680	Angela Silva Medeiros	567.982.045-27	40
	9872	Jose da Silva	987.243.098-01	10



Restrição de Integridade Referencial

DROP TABLE FUNCIONARIO;

CREATE TABLE FUNCIONARIO (

idFunc **INTEGER PRIMARY KEY**,

NomeFunc **CHAR** (30),

CPF **CHAR** (14) ,

idDepto **INTEGER** ,

FOREIGN KEY (idDepto) **REFERENCES** DEPARTAMENTO(idDepto));

INSERT INTO FUNCIONARIO **VALUES** (3456, 'Paulo de Souza Alves', '345.987.123-98',10);

INSERT INTO FUNCIONARIO **VALUES** (9872, 'Jose da Silva', '987.243.098-01',10);

INSERT INTO FUNCIONARIO **VALUES** (1890, 'Pedro Rangel de Souza', '112.872.340-81',**NULL**);

INSERT INTO FUNCIONARIO **VALUES** (4680, 'Angela Silva Medeiros', '567.982.045-27',40);

SELECT * FROM FUNCIONARIO;

DEPARTAMENTO

PK idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK idFunc	NomeFunc	CPF	FK idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40

idFunc	NomeFunc	CPF	idDepto
1890	Pedro Rangel de Souza	112.872.340-81	NULL
3456	Paulo de Souza Alves	345.987.123-98	10
4680	Angela Silva Medeiros	567.982.045-27	40
9872	Jose da Silva	987.243.098-01	10

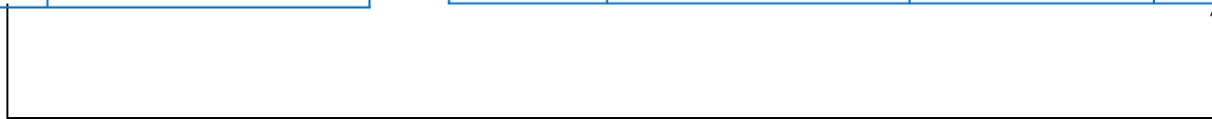


DEPARTAMENTO

PK <u>idDeppto</u>	NomeDeppto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK <u>idFunc</u>	NomeFunc	CPF	FK idDeppto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40



Posso deletar o departamento 10 ?

DELETE FROM DEPARTAMENTO WHERE idDeppto = 10;



DEPARTAMENTO

PK <u>idDepto</u>	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK <u>idFunc</u>	NomeFunc	CPF	FK <u>idDepto</u>
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40



A resposta depende das opções de tratamento da integridade referencial que estão definidas na tabela de FUNCIONARIO!



Recriando a tabela de FUNCIONARIO com a opção RESTRICT

DROP TABLE FUNCIONARIO;

CREATE TABLE FUNCIONARIO (

idFunc **INTEGER PRIMARY KEY**,

NomeFunc **CHAR** (30),

CPF **CHAR** (14) ,

idDepto **INTEGER** ,

FOREIGN KEY (idDepto) **REFERENCES** DEPARTAMENTO(idDepto)
ON DELETE RESTRICT ON UPDATE RESTRICT);

DEPARTAMENTO

PK idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK idFunc	NomeFunc	CPF	FK idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098-01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40

INSERT INTO FUNCIONARIO **VALUES** (3456, 'Paulo de Souza Alves', '345.987.123-98',10);

INSERT INTO FUNCIONARIO **VALUES** (9872, 'Jose da Silva', '987.243.098-01',10);

INSERT INTO FUNCIONARIO **VALUES** (1890, 'Pedro Rangel de Souza', '112.872.340-81',**NULL**);

INSERT INTO FUNCIONARIO **VALUES** (4680, 'Angela Silva Medeiros', '567.982.045-27',40);

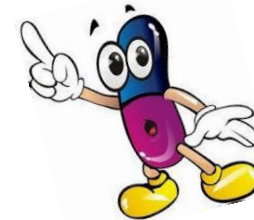
SELECT * FROM FUNCIONARIO;

idFunc	NomeFunc	CPF	idDepto
1890	Pedro Rangel de Souza	112.872.340-81	NULL
3456	Paulo de Souza Alves	345.987.123-98	10
4680	Angela Silva Medeiros	567.982.045-27	40
9872	Jose da Silva	987.243.098-01	10



Opção DELETE ON RESTRICT

DELETE FROM DEPARTAMENTO WHERE idDepto = 10;



DELETE FROM DEPARTAMENTO WHERE idDepto = 10
Error Code: 1451. Cannot delete or update a parent row: a foreign
key constraint fails (`teste`.`funcionario`, CONSTRAINT
`funcionario_ibfk_1` FOREIGN KEY (`idDepto`) REFERENCES
`departamento` (`idDepto`))





Opção UPDATE ON RESTRICT

```
UPDATE DEPARTAMENTO  
SET idDepto = 55  
WHERE idDepto = 10;
```



Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails
(`teste`.`funcionario`, CONSTRAINT
`funcionario_ibfk_1` FOREIGN KEY (`idDepto`)
REFERENCES `departamento` (`idDepto`))





Recriando a tabela de FUNCIONARIO com a opção SET NULL

DROP TABLE FUNCIONARIO;

CREATE TABLE FUNCIONARIO (

idFunc **INTEGER PRIMARY KEY**,

NomeFunc **CHAR** (30),

CPF **CHAR** (14) ,

idDepto **INTEGER** ,

FOREIGN KEY (idDepto) **REFERENCES** DEPARTAMENTO(idDepto)
ON DELETE SET NULL ON UPDATE SET NULL);

DEPARTAMENTO

PK idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK idFunc	NomeFunc	CPF	FK idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40

INSERT INTO FUNCIONARIO **VALUES** (3456, 'Paulo de Souza Alves', '345.987.123-98',10);

INSERT INTO FUNCIONARIO **VALUES** (9872, 'Jose da Silva', '987.243.098-01',10);

INSERT INTO FUNCIONARIO **VALUES** (1890, 'Pedro Rangel de Souza', '112.872.340-81',**NULL**);

INSERT INTO FUNCIONARIO **VALUES** (4680, 'Angela Silva Medeiros', '567.982.045-27',40);

SELECT * FROM FUNCIONARIO;

idFunc	NomeFunc	CPF	idDepto
1890	Pedro Rangel de Souza	112.872.340-81	NULL
3456	Paulo de Souza Alves	345.987.123-98	10
4680	Angela Silva Medeiros	567.982.045-27	40
9872	Jose da Silva	987.243.098-01	10



Opção DELETE ON SET NULL

DELETE FROM DEPARTAMENTO WHERE idDeppto = 10;
SELECT * FROM FUNCIONARIO;



DEPARTAMENTO

PK	
idDeppto	NomeDeppto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK			FK
idFunc	NomeFunc	CPF	idDeppto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40

Result Grid				
		Filter Rows:		Edit:
	idFunc	NomeFunc	CPF	idDeppto
▶	1890	Pedro Rangel de Souza	112.872.340-81	NULL
	3456	Paulo de Souza Alves	345.987.123-98	NULL
	4680	Angela Silva Medeiros	567.982.045-27	40
	9872	Jose da Silva	987.243.098-01	NULL



Recriando a tabela de FUNCIONARIO com a opção CASCADE

DROP TABLE FUNCIONARIO;

CREATE TABLE FUNCIONARIO (

idFunc **INTEGER PRIMARY KEY**,

NomeFunc **CHAR** (30),

CPF **CHAR** (14) ,

idDepto **INTEGER** ,

FOREIGN KEY (idDepto) **REFERENCES** DEPARTAMENTO(idDepto)
ON DELETE CASCADE ON UPDATE CASCADE);

DEPARTAMENTO

PK idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK idFunc	NomeFunc	CPF	FK idDepto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098-01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40



INSERT INTO FUNCIONARIO **VALUES** (3456, 'Paulo de Souza Alves', '345.987.123-98',10);

INSERT INTO FUNCIONARIO **VALUES** (9872, 'Jose da Silva', '987.243.098-01',10);

INSERT INTO FUNCIONARIO **VALUES** (1890, 'Pedro Rangel de Souza', '112.872.340-81',**NULL**);

INSERT INTO FUNCIONARIO **VALUES** (4680, 'Angela Silva Medeiros', '567.982.045-27',40);

SELECT * FROM FUNCIONARIO;

Result Grid				
Filter Rows: <input type="text"/>				
Edit:				
	idFunc	NomeFunc	CPF	idDepto
▶	1890	Pedro Rangel de Souza	112.872.340-81	NULL
	3456	Paulo de Souza Alves	345.987.123-98	10
	4680	Angela Silva Medeiros	567.982.045-27	40
	9872	Jose da Silva	987.243.098-01	10



Opção DELETE ON CASCADE

DELETE FROM DEPARTAMENTO WHERE idDeppto = 10;
SELECT * FROM FUNCIONARIO;






DEPARTAMENTO

PK	
idDeppto	NomeDeppto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO

FUNCIONARIO

PK			FK
idFunc	NomeFunc	CPF	idDeppto
3456	Paulo de Souza Alves	345.987.123-98	10
9872	José da Silva	987.243.098.01	10
1890	Pedro Rangel de Souza	112.872.340-81	
4680	Angela Silva Medeiros	567.982.045-27	40

Result Grid				
Filter Rows: <input type="text"/>				
Edit:   				
	idFunc	NomeFunc	CPF	idDeppto
▶	1890	Pedro Rangel de Souza	112.872.340-81	NULL
	4680	Angela Silva Medeiros	567.982.045-27	40



Restrições de Domínio

- ✓ Especificam que, dentro de cada tupla, o valor de cada atributo **A** deve ser um valor do domínio de **A**.
- ✓ Assim, pode-se especificar um intervalo de valores de um tipo de dados ou como um tipo de dado enumerado.
- ✓ A cláusula **CHECK** pode ser especificada na definição do atributo para definir a restrição de domínio.
- ✓ Exemplo: `idDepto INT NOT NULL CHECK (idDepto > 0 AND idDepto < 1000);`





Restrições de Domínio

- ✓ Alternativamente, pode-se especificar um conjunto de valores possíveis por meio da cláusula **ENUM**.
- ✓ A cláusula ENUM permite que um dos valores possa ser utilizado como valor do atributo.
- ✓ Exemplo: sexo **ENUM** ('M', 'F') **NOT NULL** ;





Nomeando Restrições

- ✓ Uma restrição pode receber um nome, seguindo a palavra reservada **CONSTRAINT**.
- ✓ Um nome de restrição é usado para identificar uma restrição em particular caso ela deva ser removida mais tarde e substituída por outra.
- ✓ Exemplo: **CONSTRAINT** REST

```
FOREIGN KEY (idDepto) REFERENCES DEPARTAMENTO(idDepto) ;
```




Inserção de Dados

- ✓ Feita pelo comando **INSERT**;
- ✓ Deve ser especificado o nome da Relação e uma lista de valores para a tupla.

```
U1:  INSERT INTO FUNCIONARIO  
      VALUES      ( 'Ricardo', 'K', 'Marini',  
                    '65329865388', '30-12-  
                    1962', 'Rua Itapira, 44,  
                    Santos, SP', 'M', 37.000,  
                    '65329865388', 4 );
```



Deleção de Dados

- ✓ Feita pelo comando **DELETE** ;
- ✓ Remove tuplas em uma Relação;
- ✓ Inclui uma cláusula WHERE para selecionar as tuplas a serem excluídas.

U4A:	DELETE FROM	FUNCIONARIO
	WHERE	Unome='Braga';
U4B:	DELETE FROM	FUNCIONARIO
	WHERE	Cpf='12345678966';
U4C:	DELETE FROM	FUNCIONARIO
	WHERE	Dnr=5;
U4D:	DELETE FROM	FUNCIONARIO;



Atualização de Dados

- ✓ Feita pelo comando **UPDATE** ;
- ✓ Modifica valores de atributos de uma ou mais tuplas selecionadas;
- ✓ Cláusula SET adicional no comando **UPDATE**. Especifica os atributos a serem modificados e seus novos valores.

```
U5: UPDATE PROJETO
    SET Projlocal = 'Santo André', Dnum
      = 5
    WHERE Projnumero=10;
```



```

18▼ CREATE TABLE Aluno (
19     idAluno TINYINT(2) UNSIGNED NOT NULL,
20     PRIMARY KEY (idAluno)
21 );

```

Tipos Numéricos				
Tipo	Uso		Tamanho	
		Atributo	MIN	MAX
TINYINT	Um inteiro muito pequeno	Signed:	-128	127
		Unsigned	0	255



```
34 ALTER TABLE Aluno
35 ADD NomeMae VARCHAR(45),
36 ADD NomePai VARCHAR(45);
```

	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO	PRI	NULL	
	Nome	varchar(45)	NO		NULL	
	Estado	char(2)	YES		NULL	
	Email	varchar(80)	NO		NULL	
	Peso	decimal(5,2)	YES		NULL	
	NomeMae	varchar(45)	YES		NULL	
	NomePai	varchar(45)	YES		NULL	



```
63 ALTER TABLE Aluno
64 ADD DataNascimento DATE,
65 ADD DataMatricula DATETIME,
66 ADD DataAtualizacaoRegistro TIMESTAMP;
```

		Formato	MIN	MAX
DATE	Data		'1000-01-01'	'9999-12-31'
		OBS	Formato: 'YYYY-MM-DD'	
DATETIME	Data e horário		'1000-01-01 00:00:00'	'9999-12-31 23:59:59'
		OBS	Formato: 'YYYY-MM-DD HH:MM:SS'	
TIMESTAMP	Timestamp		'1970-01-01 00:00:00'	aproximadamente 2037
		OBS	Formato: YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD ou YYMMDD, dependendo se M é 14 (ausente), 12, 8 ou 6, podendo ser strings ou números. Este tipo é recomendável para instruções de INSERT ou UPDATE pois é automaticamente marcado com os valores da operação mais recente quando não informado.	

	Field	Type	Null	Key	Default
▶	idAluno	tinyint(3) unsigned zerofill	NO	PRI	NULL
	Nome	varchar(45)	NO		NULL
	Estado	char(2)	YES		NULL
	email	varchar(80)	NO		NULL
	Peso	decimal(5,2)	YES		NULL
	NomeMae	varchar(40)	NO		NULL
	NomePai	varchar(40)	YES		NULL
	DataNascimento	date	YES		NULL
	DataMatricula	datetime	YES		NULL
	DataAtualizacaoRegistro	timestamp	NO		CURRENT_TIMESTAMP



```

54 ALTER TABLE Aluno
55 CHANGE NomeMae NomeMae VARCHAR(40) NOT NULL,
56 CHANGE NomePai NomePai VARCHAR(40);

```

	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO	PRI	NULL	
	Nome	varchar(45)	NO		NULL	
	Estado	char(2)	YES		NULL	
	email	varchar(80)	NO		NULL	
	Peso	decimal(5,2)	YES		NULL	
	NomeMae	varchar(45)	YES		NULL	
	NomePai	varchar(45)	YES		NULL	



```
58 ALTER TABLE Aluno
59 CHANGE Email EmailAluno VARCHAR(80);
```

	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO	PRI	NULL	
	Nome	varchar(45)	NO		NULL	
	Estado	char(2)	YES		NULL	
	email	varchar(80)	NO		NULL	
	Peso	decimal(5,2)	YES		NULL	
	NomeMae	varchar(40)	NO		NULL	
	NomePai	varchar(40)	YES		NULL	



```
38 ALTER TABLE Aluno DROP Email;
```

	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO	PRI	<small>HULL</small>	
	Nome	varchar(45)	NO		<small>HULL</small>	
	Estado	char(2)	YES		<small>HULL</small>	
	Peso	decimal(5,2)	YES		<small>HULL</small>	
	NomeMae	varchar(45)	YES		<small>HULL</small>	
	NomePai	varchar(45)	YES		<small>HULL</small>	



```
38 ALTER TABLE Aluno
39 DROP Email,
40 DROP NomePai; |
```

	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO	PRI	NULL	
	Nome	varchar(45)	NO		NULL	
	Estado	char(2)	YES		NULL	
	Peso	decimal(5,2)	YES		NULL	
	NomeMae	varchar(45)	YES		NULL	



```

38▼ CREATE TABLE Aluno (
39     idAluno TINYINT(3) UNSIGNED ZEROFILL NOT NULL,
40     Nome VARCHAR(45) NOT NULL,
41     Estado CHAR(2),
42     Peso DECIMAL(5,2)
43 );

```

O que há de errado aqui ?



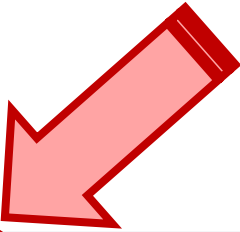
	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO		NULL	
	Nome	varchar(45)	NO		NULL	
	Estado	char(2)	YES		NULL	
	Peso	decimal(5,2)	YES		NULL	



```

38▼ CREATE TABLE Aluno (
39     idAluno TINYINT(3) UNSIGNED ZEROFILL NOT NULL,
40     Nome VARCHAR(45) NOT NULL,
41     Estado CHAR(2),
42     Peso DECIMAL(5,2)
43 );
  
```

O que há de errado aqui ?

	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO		NULL	
	Nome	varchar(45)	NO		NULL	
	Estado	char(2)	YES		NULL	
	Peso	decimal(5,2)	YES		NULL	



```
46 ALTER TABLE Aluno
47 ADD PRIMARY KEY (idAluno);
```

	Field	Type	Null	Key	Default	Extra
►	idAluno	tinyint(3) unsigned zerofill	NO	PRI	NULL	
	Nome	varchar(45)	NO		NULL	
	Estado	char(2)	YES		NULL	
	Peso	decimal(5,2)	YES		NULL	



Posso facilitar minha vida se por acaso o idAluno
for um código numérico sequencial?





```
69▼ CREATE TABLE Aluno (
70     idAluno INT(10) UNSIGNED ZEROFILL NOT NULL AUTO_INCREMENT,
71     Nome VARCHAR(45) NOT NULL,
72     PRIMARY KEY (idAluno)
73 );
```

	Field	Type	Null	Key	Default	Extra
▶	idAluno	int(10) unsigned zerofill	NO	PRI	NULL	auto_increment
	Nome	varchar(45)	NO		NULL	



Deletando uma tabela



```
48  
49 DROP TABLE Aluno;  
50
```




Deletando o banco de dados



DROP DATABASE TESTE