

FIAP GRADUAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DATABASE MODELING & SQL

Profa. Rita de Cássia Rodrigues



rita@fiap.com.br

Prof. William Maximiano



profwilliam.junior@fiap.com.br

COMANDOS SQL

(DDL → LINGUAGEM DE DEFINIÇÃO DE DADOS): PARTE 1

- Objetivo
- Conceitos referentes a linguagem de definição de dados
- Revisão dos Conceitos
- Exercícios

- ❑ Aplicação da linguagem de definição de dados;

❑ Linguagem de definição de dados

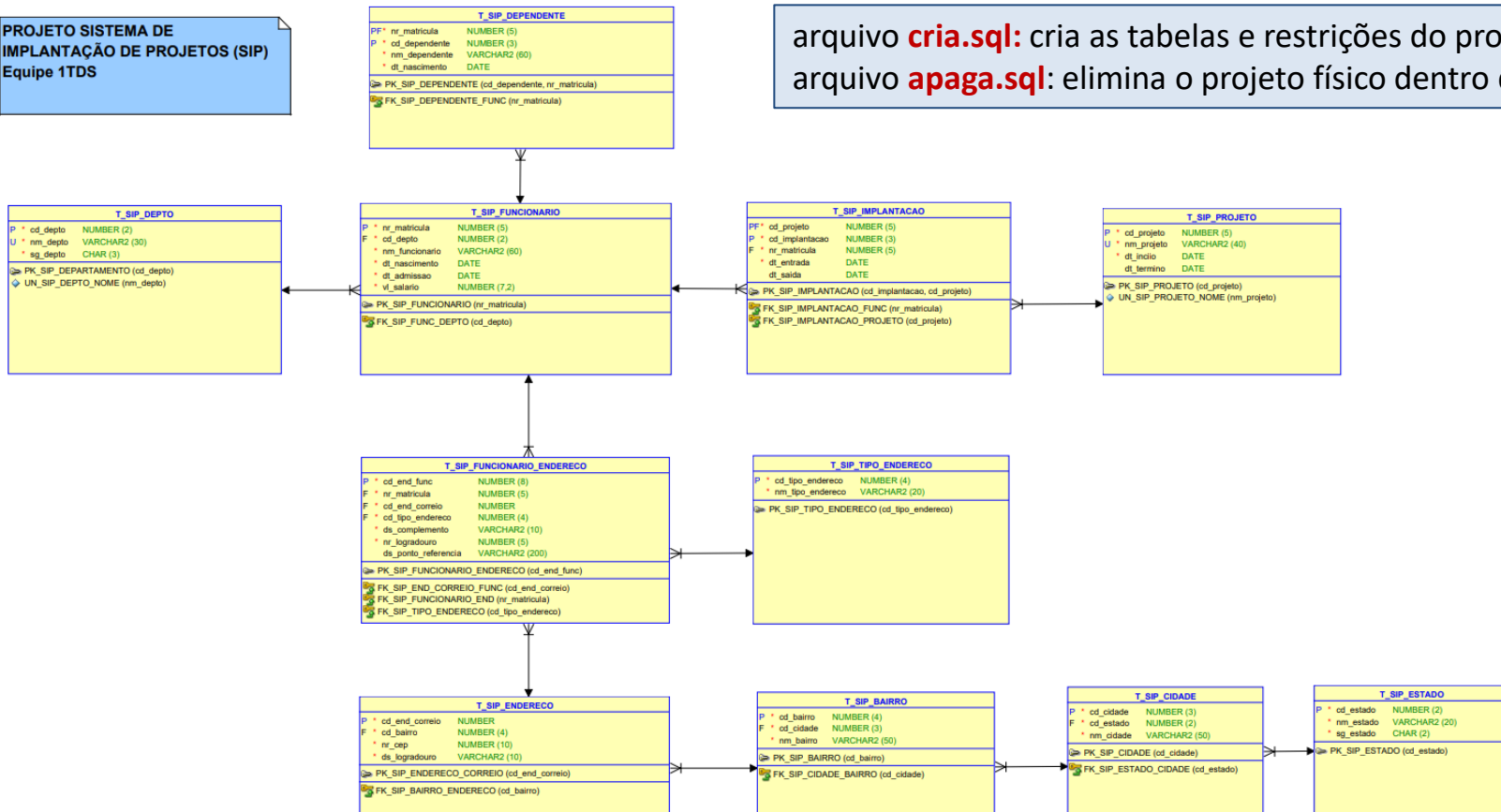
❑ DDL (CREATE, ALTER e DROP)

❑ Constraints

❑ Utilizaremos o projeto SIP como estrutura base do material da disciplina. Entenda o funcionamento dessa solução a partir do banco de dados físico.

PROJETO SISTEMA DE
IMPLANTAÇÃO DE PROJETOS (SIP)
Equipe 1TDS

arquivo **cria.sql**: cria as tabelas e restrições do projeto
arquivo **apaga.sql**: elimina o projeto físico dentro do SGBD





Linguagem SQL

SQL: Structured Query Language (Linguagem de Consulta Estruturada)

Um pouco de história...

É uma **linguagem de definição, manipulação e controle de banco de dados**.

Representa um conjunto de comandos responsáveis pela definição das tabelas, seleção e atualização dos dados em um SGBD.

Início de 1970: O departamento de pesquisas da IBM desenvolveu a linguagem SQL, como forma de interface para o sistema de banco de dados relacional, denominado System R. Inicialmente a linguagem SQL foi chamada de SEQUEL (Structured English Query Language).

1977: A nomenclatura foi revisada e passou a ser chamada definitivamente de SQL.



Linguagem SQL

SQL: Structured Query Language (Linguagem de Consulta Estruturada)

Um pouco de história...

1986: ANSI (American National Standard Institute – Instituto Nacional Americano de Padrões) em conjunto com a ISO (International Standards Organization – Organização Internacional de Padrões), publicou o padrão de linguagem SQL, conhecida como SQL-86 ou SQL-1, tornando-se a linguagem padrão adotada para os bancos de dados relacionais.

O primeiro SGBD, lançado no mercado, implementando a linguagem SQL, foi o Oracle (que era chamado de Relational Software) em 1979.



Linguagem SQL

SQL: Structured Query Language (Linguagem de Consulta Estruturada)

Características...

- ❑ Embora o SQL seja um padrão de fato, há diferenças nas implementações dessa linguagem nos vários SGBD's de mercado.
- ❑ Cada fabricante de software cria extensões (com sintaxes, funções, recursos, etc.) específicos de seus sistemas, formando diferentes “dialetos” e “versões” da linguagem.
- ❑ É possível escrever aplicações utilizando apenas o SQL padrão ANSI (com os comandos comuns à maioria dos SGBD's).
- ❑ Porém, na prática, muitas vezes faz-se uso das sintaxes proprietárias para aproveitar recursos específicos ou conseguir melhor desempenho.



Linguagem SQL

SQL: Structured Query Language (Linguagem de Consulta Estruturada)

Características...

Em decorrência da padronização, o SQL apresenta algumas características que merecem destaques:

- ☐ As instruções padronizadas seguem a mesma nomenclatura e formato para os diferentes tipos de SGBD, respeitando as particularidades de cada um.
- ☐ A migração de um SGBD para outro não requer grandes mudanças.
- ☐ Quando ocorre a migração de um SGBD, a adaptação dos profissionais é facilitada, com redução de tempo e de custos para os treinamentos, pois as instruções possuem nomes e funcionalidades iguais.
- ☐ Portabilidade entre plataformas.



Linguagem SQL

SQL: Structured Query Language (Linguagem de Consulta Estruturada)

Características...

- ❑ As queries (consultas) do SQL têm uma estrutura semelhante à linguagem natural (em inglês).
- ❑ As instruções de manipulação de dados sempre trabalham com tabelas e/ou retornam resultados em forma de tabelas (com linhas e colunas).
- ❑ É uma linguagem não-procedural, isto é, as sentenças em SQL declaram a tarefa a ser realizada (“o que”) sem a necessidade de especificar os procedimentos passo-a-passo que o sistema deve seguir para executá-la (“como”).
- ❑ Independência física dos dados e dispositivos: não especifica o método de acesso aos dados e nem como estão armazenados (os SGBD’s possuem um “otimizador” que se encarrega da maneira de recuperar os dados).

Linguagem SQL

SQL: Structured Query Language (Linguagem de Consulta Estruturada)

Características...



Para ilustrar vejamos as diferenças de sintaxe entre os SGBD's, para recuperar a primeira linha de uma tabela.



```
SELECT * FROM T_CLIENTE WHERE ROWNUM = 1;
```



```
SELECT TOP 1 * FROM T_CLIENTE;
```



```
SELECT * FROM T_CLIENTE LIMIT 1;
```

Categorias de Instruções da linguagem SQL

Categoria	Descrição
DDL Data Definition Language	Utilizada para definição e descrição dos dados.
DML Data Manipulation Language	Utilizada para manipulação dos dados. Possui uma subcategoria, chamada DCL (Data Control Language), que é utilizada para controlar o acesso aos dados.
TCL Transact Control Language	Utilizada para controle de transações.

Categorias de Instruções da linguagem SQL

Categorias de instruções DDL – *Data Definition Language*

Finalidade	Definição e manutenção das estruturas do banco de dados, tais como a criação do próprio banco de dados e das tabelas que o compõem, além das relações entre as tabelas e os objetos do banco de dados.
Instruções	
Create	Criação de estruturas de objetos do banco de dados.
Alter	Alteração da estrutura de objetos do banco de dados.
Drop	Eliminação das estruturas de objetos do banco de dados.
Truncate	Exclusão física de linhas de tabelas.
Rename	Renomeação de objetos do banco de dados.
Comment	Inclusão de comentários aos objetos do banco de dados.

Referência: Puga, S.; França. Ed.; Goya, M. Banco de Dados – Implementação em SQL, PL/SQL e Oracle 11g. Pearson, 2014. Página: 170

Categorias de Instruções da linguagem SQL

Categorias de instruções DML – *Data Manipulation Language*

Finalidade	Consultas, inserções, exclusões e alterações em um ou mais registros, de uma ou mais tabelas, de maneira simultânea.
Instruções	
Insert	Inserção de dados.
Update	Alteração de dados.
Delete	Exclusão de dados.
Select	Consulta de dados.
Merge	Combinação das instruções insert, update e delete.

Referência: Puga, S.; França. Ed.; Goya, M. Banco de Dados – Implementação em SQL, PL/SQL e Oracle 11g. Pearson, 2014. Página: 170

Categorias de Instruções da linguagem SQL

Subcategorias de instruções DCL – *Data Control Language*

Finalidade	Controle dos privilégios de usuários, de forma que o administrador do banco de dados possa determinar o nível de acesso de um usuário aos objetos do banco de dados, concedendo privilégios ou retirando esse acesso e revogando os privilégios.
Instruções	
Grant	Atribuição de privilégios aos usuários do banco de dados.
Revoke	Revogação de privilégios dos usuários do banco de dados.

Referência: Puga, S.; França. Ed.; Goya, M. Banco de Dados – Implementação em SQL, PL/SQL e Oracle 11g. Pearson, 2014. Página: 171

Categorias de Instruções da linguagem SQL

Categorias de instruções TCL – *Transact Control Language*

Finalidade	Controle de transações, consideradas o conjunto de uma ou mais operações DML realizadas no banco de dados.
Instruções	
Commit	Confirmação das manipulações.
Rollback	Desistência das manipulações.
Savepoint	Criação de pontos para o controle das transações.

Referência: Puga, S.; França. Ed.; Goya, M. Banco de Dados – Implementação em SQL, PL/SQL e Oracle 11g. Pearson, 2014. Página: 171

Transformação do Modelo para SQL: Criação de Tabelas



Comando CREATE TABLE

Sintaxe para criar a estrutura de uma tabela definindo as colunas (campos) e as chaves primárias e estrangeiras existentes (relacionamento PK x FK).

Sintaxe:

```
CREATE TABLE <nome-tabela> (  
  <nome-coluna> <tipo-do-dado> [NOT NULL]  
  PRIMARY KEY (nome-coluna-chave)  
  FOREIGN KEY (nome-coluna-chave-estrangeira)  
  REFERENCES <nome-tabela-pai> (nome-coluna-chave-primária));
```

Transformação do Modelo para SQL: Comando ALTER TABLE



Comando ALTER TABLE

Alterar a estrutura de uma tabela acrescentando, alterando, retirando e alterando nomes, formatos das colunas e a integridade referencial definidas em uma determinada tabela.

Sintaxe:

```
ALTER TABLE <nome-tabela>
```

```
DROP COLUMN <nome-coluna>
```

```
ADD <nome-coluna> <tipo-do-dado> [NOT NULL]
```

```
[NOT NULL WITH DEFAULT]
```

```
RENAME <nome-coluna> <novo-nome-coluna>
```

```
MODIFY <nome-coluna> <tipo-do-dado> [NULL]
```

```
[NOT NULL];
```

Transformação do Modelo para SQL: Comando DROP TABLE



Comando DROP TABLE

Deletar a estrutura e os dados existentes em uma tabela. Após a execução deste comando estarão deletados todos os dados, estrutura e índices de acessos que estejam a ela associados.

Sintaxe:

```
DROP TABLE <nome-tabela>;
```

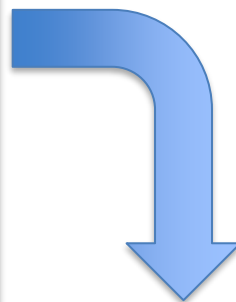
Linguagem SQL

Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DE UMA TABELA



T_SIP_PROJETO		
P *	cd_projeto	NUMBER (5)
U *	nm_projeto	VARCHAR2 (40)
*	dt_inicio	DATE
	dt_termino	DATE
PK_SIP_PROJETO (cd_projeto)		
UN_SIP_PROJETO_NOME (nm_projeto)		



Atenção:

A **constraint NOT NULL**, é declarada logo após a definição do tipo de dado e tamanho da coluna/campo.

Atenção:

O ; (caractere ponto e vírgula), indica o término da linha de instrução.

```
-- CRIAR TABELA: PROJETO
CREATE TABLE T_SIP_PROJETO
(
    CD_PROJETO          NUMBER(5)          NOT NULL ,
    NM_PROJETO          VARCHAR2(40)       NOT NULL ,
    DT_INICIO           DATE                NOT NULL ,
    DT_TERMINO          DATE                NULL
);
```

-- CAMPOS OPCIONAIS ESCREVER NULL. SE OMITIR SERÁ CONSIDERADO COMO NULL

Transformação do Modelo para SQL: CONSTRAINTS

É parte opcional da cláusula **CREATE TABLE** e **ALTER TABLE**. São regras agregadas a colunas e tabelas.

As **restrições(Constraints)** evitam:

- ☐ Que uma tabela seja deletada se houver pendências;
- ☐ Dados inválidos sejam inseridos em branco;
- ☐ Garante a integridade dos dados armazenados.



Tipos de Constraints

- **Not Null**
- **Unique**
- **Check**
- **Default**
- **Primary key**
- **Foreign key**

Transformação do Modelo para SQL: CONSTRAINTS

Adicionando Restrições (Constraints)



Normalmente utilizamos o comando “**ALTER TABLE**” para inserir restrições (**Constraints**).

É uma boa prática adicionar ou eliminar restrições, mas não modificar restrições.

Adicione uma restrição **NOT NULL** utilizando a cláusula **MODIFY** (veremos em breve).

É recomendado que se especifique sempre um nome significativo para as restrições (constraints). Dessa forma quando ocorrer um erro de restrições, ficará mais fácil de identificar o problema.



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DE UMA TABELA CONTENDO UM CAMPO COM VALOR DEFAULT CONSTRAINT DEFAULT

No momento da inserção dos dados, quando uma data de criação do departamento não for informada, será armazenado o valor DEFAULT informado, desta forma o conteúdo da coluna/campo não ficará nulo.

Para isso precisamos utilizar a constraint DEFAULT, conforme o exemplo abaixo.

Neste exemplo estamos utilizando o valor default SYSDATE, que no Oracle é a função que retorna a data e horário do sistema do servidor Oracle.

```
CREATE TABLE T_DEPTO
(
    CD_DEPTO    NUMBER(3)    NOT NULL,
    NM_DEPTO    VARCHAR(30)  NOT NULL,
    DT_CRIACAO  DATE         DEFAULT SYSDATE
);
```



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHAVE PRIMÁRIA COMO PARTE DO ALTER TABLE

T_SIP_PROJETO

P *	cd_projeto	NUMBER (5)
U *	nm_projeto	VARCHAR2 (40)
*	dt_inicio	DATE
	dt_termino	DATE

PK_SIP_PROJETO (cd_projeto)
UN_SIP_PROJETO_NOME (nm_projeto)

LEMBRANDO...

CHAVE PRIMÁRIA ou PRIMARY KEY, é a restrição/regra que identifica cada registro (linha/ocorrência/instância/tupla) da tabela (entidade) de forma única.

```
-- CHAVE PRIMARIA DA TABELA PROJETO
ALTER TABLE T_SIP_PROJETO
ADD CONSTRAINT PK_SIP_PROJETO
PRIMARY KEY (CD_PROJETO);
```

A adição de constraints em uma tabela pode ser feita de duas maneiras:

1. Como parte do **CREATE TABLE**, ou seja, durante a criação de uma tabela. Nesse caso a constraint pode ser criada de duas formas:
 1. no nível de coluna, ou
 2. no nível de tabela.
2. Como parte do **ALTER TABLE**, ou seja, após a criação de uma tabela (é o mais comum)



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHAVE PRIMÁRIA COMO PARTE DO CREATE TABLE

```
-- CRIAR TABELA: T SIP PROJETO
-- CONSTRAINT PRIMARY KEY - NIVEL DE TABELA
CREATE TABLE T_SIP_PROJETO
(
    CD_PROJETO          NUMBER(5)          NOT NULL ,
    NM_PROJETO          VARCHAR2(40)       NOT NULL ,
    DT_INICIO           DATE                NOT NULL ,
    DT_TERMINO          DATE                NULL  ,
    CONSTRAINT PK_SIP_PROJETO PRIMARY KEY (CD_PROJETO)
);
```

```
-- CRIAR TABELA: T SIP PROJETO
-- CONSTRAINT PRIMARY KEY - NIVEL DE COLUNA
CREATE TABLE T_SIP_PROJETO
(
    CD_PROJETO          NUMBER(5)          NOT NULL CONSTRAINT PK SIP PROJETO PRIMARY KEY,
    NM_PROJETO          VARCHAR2(40)       NOT NULL ,
    DT_INICIO           DATE                NOT NULL ,
    DT_TERMINO          DATE                NULL
);
```

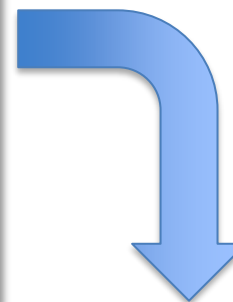


Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHAVE PRIMÁRIA COMPOSTA

ATENÇÃO: Para implementação de **chaves compostas**, ou qualquer *constraint* que referencie **mais de uma coluna**, é necessário defini-la **fora** da especificação da coluna.

T_SIP_DEPENDENTE		
PF *	nr_matricula	NUMBER (5)
P *	cd_dependente	NUMBER (3)
*	nm_dependente	VARCHAR2 (60)
*	dt_nascimento	DATE
🔑 PK_SIP_DEPENDENTE (nr_matricula, cd_dependente)		
🔗 FK_SIP_DEPENDENTE_FUNC (nr_matricula)		



CONSTRAINT CHAVE
PRIMÁRIA COMPOSTA
COMO PARTE DO ALTER
TABLE

```
-- CHAVE PRIMARIA DA TABELA DEPENDENTE
ALTER TABLE T_SIP_DEPENDENTE
ADD CONSTRAINT PK_SIP_DEPENDENTE
PRIMARY KEY (NR_MATRICULA, CD_DEPENDENTE);
```



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHAVE PRIMARIA COMPOSTA COMO PARTE DO CREATE TABLE

```
-- CRIAR TABELA: DEPENDENTE
-- CONSTRAINT CHAVE PRIMÁRIA - NIVEL DE TABELA
CREATE TABLE T_SIP_DEPENDENTE
(
    NR_MATRICULA          NUMBER(5)          NOT NULL ,
    CD_DEPENDENTE         NUMBER(2)          NOT NULL ,
    NM_DEPENDENTE         VARCHAR2(50)       NOT NULL ,
    DT_NASCIMENTO         DATE               NOT NULL ,
    CONSTRAINT PK SIP_DEPENDENTE PRIMARY KEY (NR_MATRICULA, CD_DEPENDENTE)
);
```



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT UNIQUE COMO PARTE DO ALTER TABLE

T_SIP_PROJETO

P *	cd_projeto	NUMBER (5)
U *	nm_projeto	VARCHAR2 (40)
*	dt_inicio	DATE
	dt_termino	DATE

PK SIP PROJETO (cd_projeto)

UN_SIP_PROJETO_NOME (nm_projeto)

LEMBRANDO...

UNIQUE CONSTRAINT, é a restrição/regra que indica que uma determinada coluna (atributo/campo) de uma tabela não poderá ter repetição, ou seja, não poderá ter valores repetidos para o conteúdo da coluna.

```
-- CONSTRAINT UNIQUE NA TABELA PROJETO -> CAMPO NOME
ALTER TABLE T_SIP_PROJETO
ADD CONSTRAINT UN_SIP_PROJETO_NOME
    UNIQUE (NM_PROJETO);
```



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT UNIQUE COMO PARTE DO CREATE TABLE

```
-- CRIAR TABELA: T SIP PROJETO
-- CONSTRAINT UNIQUE - NIVEL DE TABELA
CREATE TABLE T_SIP_PROJETO
(
    CD_PROJETO          NUMBER(5)          NOT NULL ,
    NM_PROJETO          VARCHAR2(40)       NOT NULL ,
    DT_INICIO           DATE               NOT NULL ,
    DT_TERMINO          DATE               NULL ,
    CONSTRAINT UN SIP PROJETO NOME UNIQUE (NM PROJETO)
);
```

```
-- CRIAR TABELA: T SIP PROJETO
-- CONSTRAINT UNIQUE - NIVEL DE COLUNA
CREATE TABLE T_SIP_PROJETO
(
    CD_PROJETO          NUMBER(5)          NOT NULL ,
    NM_PROJETO          VARCHAR2(40)       NOT NULL ,
    DT_INICIO           DATE               NOT NULL ,
    DT_TERMINO          DATE               NULL ,
    CONSTRAINT UN_SIP_PROJETO_NOME UNIQUE,
);
```





Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHECK COMO PARTE DO ALTER TABLE

T_SIP_PROJETO

P * cd_projeto **NUMBER (5)**
U * nm_projeto **VARCHAR2 (40)**
* dt_inicio **DATE**
* dt_termino **DATE**

 **PK_SIP_PROJETO (cd_projeto)**
 **UN_SIP_PROJETO_NOME (nm_projeto)**

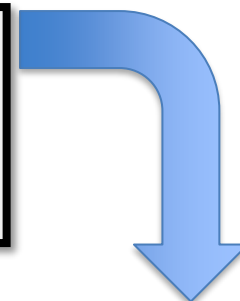
LEMBRANDO...

CHECK CONSTRAINT, é a restrição/regra que define o domínio de um determinado campo (atributo). É uma constraint de verificação.

Restrições do Nível da Tabela



	Nome ▲	Regra de Validação	Gerar em DDL
1	CK_SIP_PROJETO_DATA	DT_TERMINO > DT_INICIO	<input checked="" type="checkbox"/>



```
-- CONSTRAINT CHECK NA TABELA PROJETO -> VALIDAÇÃO DOS CAMPOS DE DATA
ALTER TABLE T_SIP_PROJETO
ADD CONSTRAINT CK_SIP_PROJETO_DATA
CHECK (DT_TERMINO > DT_INICIO);
```



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHECK COMO PARTE DO CREATE TABLE

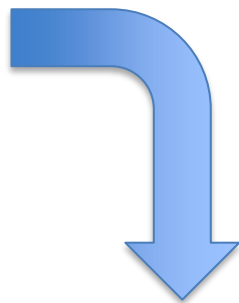
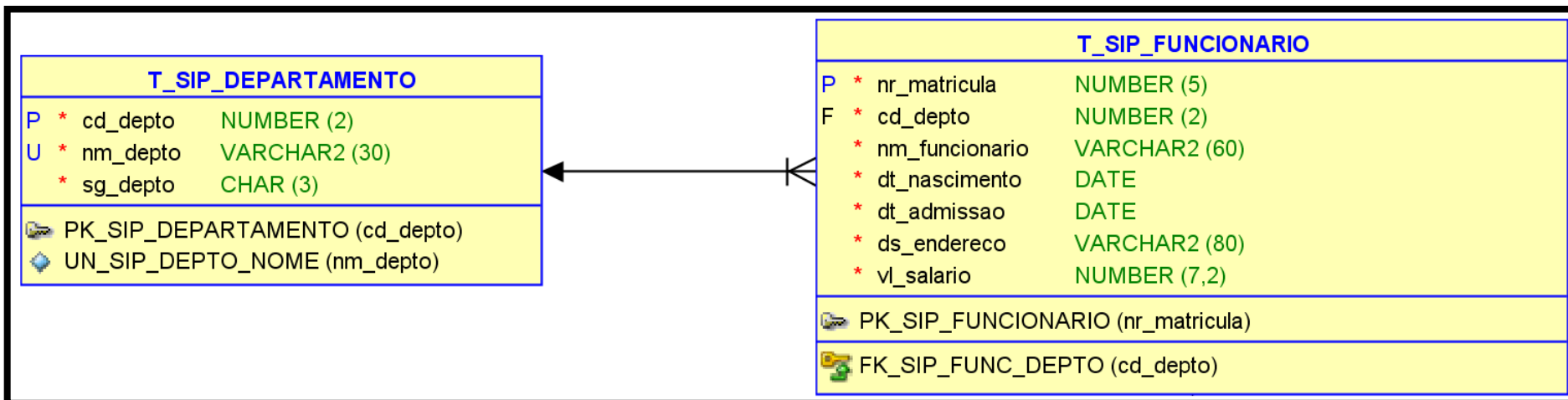
```
-- CRIAR TABELA: T SIP PROJETO
-- CONSTRAINT CHECK - NIVEL DE TABELA
CREATE TABLE T_SIP_PROJETO
(
    CD_PROJETO          NUMBER(5)          NOT NULL ,
    NM_PROJETO          VARCHAR2(40)       NOT NULL ,
    DT_INICIO           DATE                NOT NULL ,
    DT_TERMINO          DATE                NULL  ,
    CONSTRAINT CK SIP PROJETO DATA CHECK (DT_TERMINO > DT_INICIO)
);
```

Atenção: Caso a *check constraint* faça referência a **mais de uma coluna**, ela deve ser definida separada da especificação de uma coluna específica, ou seja, ou como parte do comando `alter table`, ou como parte do comando `create table`, apenas no nível de tabela.



Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHAVE ESTRANGEIRA COMO PARTE DO ALTER TABLE



LEMBRANDO...

CHAVE ESTRANGEIRA ou **FOREIGN KEY**, é a restrição/regra que estabelece a relação-relacionamento (associação) entre duas tabelas (entidades).

```
-- CRIAR O RELACIONAMENTO ENTRE FUNCIONARIO E DEPARTAMENTO
ALTER TABLE T_SIP_FUNCIONARIO
ADD CONSTRAINT FK_SIP_FUNC_DEPTO
FOREIGN KEY (CD_DEPTO)
REFERENCES T_SIP_DEPTO (CD_DEPTO);
```




Transformação do Modelo Relacional para SQL

EXEMPLO DE CRIAÇÃO DA CONSTRAINT CHAVE ESTRANGEIRA COMO PARTE DO CREATE TABLE

```
-- CRIAR TABELA: DEPENDENTE
-- CONSTRAINT CHAVE ESTRANGEIRA - NIVEL DE TABELA
CREATE TABLE T_SIP_FUNCIONARIO
(
    NR_MATRICULA          NUMBER(5)          NOT NULL ,
    CD_DEPTO              NUMBER(3)          NOT NULL ,
    NM_FUNCIONARIO        VARCHAR2(50)       NOT NULL ,
    DT_NASCIMENTO         DATE               NOT NULL ,
    DT_ADMISSAO           DATE               NOT NULL ,
    DS_ENDERECO           VARCHAR2(100)      NOT NULL ,
    VL_SALARIO            NUMBER(7,2)        NOT NULL ,
    CONSTRAINT FK_SIP_FUNC_DEPTO FOREIGN KEY (CD_DEPTO) REFERENCES T_SIP_DEPTO (CD_DEPTO)
);
```

```
-- CRIAR TABELA: DEPENDENTE
-- CONSTRAINT CHAVE ESTRANGEIRA - NIVEL DE COLUNA
CREATE TABLE T_SIP_FUNCIONARIO
(
    NR_MATRICULA          NUMBER(5)          NOT NULL ,
    CD_DEPTO              NUMBER(3)          NOT NULL CONSTRAINT FK_SIP_FUNC_DEPTO REFERENCES T_SIP_DEPTO (CD_DEPTO),
    NM_FUNCIONARIO        VARCHAR2(50)       NOT NULL ,
    DT_NASCIMENTO         DATE               NOT NULL ,
    DT_ADMISSAO           DATE               NOT NULL ,
    DS_ENDERECO           VARCHAR2(100)      NOT NULL ,
    VL_SALARIO            NUMBER(7,2)        NOT NULL
);
```



Transformação do Modelo Relacional para SQL

EXEMPLO PARA ELIMINAR TABELAS

```
DROP TABLE T_SIP_PROJETO;
```

EXEMPLO PARA ELIMINAR UMA TABELA E CONSTRAINTS DEPENDENTES

Constraints dependentes (relacionamentos – integridade referencial)

```
DROP TABLE T_SIP_PROJETO CASCADE CONSTRAINTS;
```

EXEMPLO PARA ELIMINAR TABELAS (sem enviar para a lixeira)

```
DROP TABLE T_SIP_PROJETO PURGE;
```

EXEMPLO PARA ELIMINAR UMA TABELA E CONSTRAINTS DEPENDENTES (sem enviar para a lixeira)

```
DROP TABLE T_SIP_PROJETO CASCADE CONSTRAINTS PURGE;
```

ATENÇÃO!

A cláusula PURGE (a partir da versão 10g do Oracle), no final da instrução, permite que a tabela seja eliminada definitivamente, sem que fique na “lixeira” (**RECYCLEBIN**):





Transformação do Modelo Relacional para SQL

EXEMPLO DE CONSULTA DA DESCRIÇÃO DE UMA TABELA

```
DESCRIBE T_SIP_PROJETO;  
-- OU  
DESC T_SIP_PROJETO;
```

RESULTADO DA CONSULTA DA DESCRIÇÃO DE UMA TABELA

Nome	Nulo?	Tipo
CD_PROJETO	NOT NULL	NUMBER(5)
NM_PROJETO	NOT NULL	VARCHAR2(40)
DT_INICIO	NOT NULL	DATE
DT_TERMINO		DATE

Transformação do Modelo para SQL - Comando ALTER TABLE

PARA EXEMPLIFICAR OUTRAS APLICAÇÕES DO COMANDO ALTER TABLE, IREMOS CONSTRUIR A TABELA ABAIXO:



```
-- CRIAREMOS UMA ESTRUTURA EXEMPLO, PARA APLICARMOS OS COMANDOS DDL (ALTER TABLE)
CREATE TABLE T_TESTE_DDL
(
    CD_TESTE      NUMBER(3)          NOT NULL ,
    NM_TESTE      VARCHAR2(30)       NOT NULL ,
    DS_TESTE      VARCHAR2(60)       NULL ,
    DT_TESTE      DATE               NULL
);
-- CHAVE PRIMARIA
ALTER TABLE T_TESTE_DDL ADD CONSTRAINT PK_TESTE_DDL PRIMARY KEY (CD_TESTE);
-- CONSTRAINT UNIQUE
ALTER TABLE T_TESTE_DDL ADD CONSTRAINT UN_TESTE_DDL_NOME UNIQUE (NM_TESTE);
ALTER TABLE T_TESTE_DDL ADD CONSTRAINT UN_TESTE_DDL_DESC UNIQUE (DS_TESTE);
```

Transformação do Modelo para SQL - Comando ALTER TABLE



Comando ALTER TABLE

Alterar a estrutura de uma tabela acrescentando, alterando, retirando e alterando nomes, formatos das colunas e a integridade referencial definidas em uma determinada tabela.

Sintaxe:

```
ALTER TABLE <nome-tabela>
```

```
DROP COLUMN <nome-coluna>
```

```
ADD <nome-coluna> <tipo-do-dado> [NOT NULL]  
[NOT NULL WITH DEFAULT]
```

```
RENAME <nome-coluna> <novo-nome-coluna>
```

```
MODIFY <nome-coluna> <tipo-do-dado> [NULL]  
[NOT NULL];
```

Transformação do Modelo para SQL - Comando ALTER TABLE

ADICIONAR UMA OU MAIS COLUNAS A UMA TABELA JÁ EXISTENTE



EXEMPLO DE ADIÇÃO DE UMA COLUNA NA TABELA JÁ EXISTENTE

```
ALTER TABLE T_TESTE_DDL ADD DS_EMAIL VARCHAR2(80);
```

EXEMPLO DE ADIÇÃO DE DUAS OU MAIS COLUNAS NA TABELA JÁ EXISTENTE

```
ALTER TABLE T_TESTE_DDL ADD ( DS_OBS1 VARCHAR2(50) NOT NULL, DS_OBS2 VARCHAR2(50) );
```

Observações importantes!

Podem ser acrescentadas colunas com respectivas constraints, da mesma forma que foi exemplificado no material anterior, onde falamos sobre criação de constraints.

Caso a tabela contenha registros, é comum nos depararmos com o problema de não conseguir adicionar uma coluna com NOT NULL, caso isto ocorra, será necessário adicionar a coluna (opcional – NULL), popular a coluna e posteriormente alterá-la para NOT NULL.

Transformação do Modelo para SQL - Comando ALTER TABLE



MODIFICAR UMA COLUNA DE UMA TABELA JÁ EXISTENTE

Utilizando o comando ALTER TABLE com a cláusula **MODIFY**, podemos modificar várias definições de uma coluna: tipo, tamanho, obrigatoriedade (*not null*) e valor *default*.

EXEMPLO AUMENTO DO TAMANHO DE UMA COLUNA

```
ALTER TABLE T_TESTE_DDL MODIFY DS_OBS1 VARCHAR2(80);
```

EXEMPLO DE MODIFICAÇÃO DO TIPO DE DADO DA COLUNA

```
ALTER TABLE T_TESTE_DDL MODIFY NM_TESTE CHAR(30);
```

EXEMPLO DE ALTERAÇÃO DE UMA COLUNA NULL (OPCIONAL) PARA NOT NULL (OBRIGATÓRIA)

```
ALTER TABLE T_TESTE_DDL MODIFY DS_TESTE NOT NULL;
```

EXEMPLO DE MODIFICAÇÃO DO VALOR DEFAULT

```
ALTER TABLE T_TESTE_DDL MODIFY DT_TESTE DEFAULT SYSDATE;
```

Transformação do Modelo para SQL - Comando ALTER TABLE

MODIFICAR UMA COLUNA DE UMA TABELA JÁ EXISTENTE



EXEMPLO DE ALTERAÇÃO DE TIPO DE DADO, TAMANHO E OBRIGATORIEDADE DE UM COLUNA

```
ALTER TABLE T_TESTE_DDL MODIFY NM_TESTE VARCHAR2(40) NULL;
```

EXEMPLO DE ALTERAÇÃO DE UMA COLUNA NOT NULL (OBRIGATÓRIA) PARA NULL (OPCIONAL)

```
ALTER TABLE T_TESTE_DDL MODIFY DS_OBS1 NULL;
```

Observações importantes!

Para modificar o **datatype (tipo de dados)** de uma coluna, ela deve estar **vazia** (todos os registros sem valor nesta coluna) ou deve haver **compatibilidade** entre o tipo atual e o novo.

Situação 1: Ao tentar alterar uma coluna do tipo VARCHAR para NUMBER (ou vice-versa), caso haja dados, resultará em erro.

Situação 2: Mudar o tipo de dados de VARCHAR para CHAR é possível, mesmo que já existam dados, desde que a capacidade (tamanho) estejam adequadas.

Transformação do Modelo para SQL - Comando ALTER TABLE

REMOVER COLUNAS E CONSTRAINTS



Utilizando o comando ALTER TABLE com a cláusula **DROP**, podemos remover (eliminar/excluir) colunas e constraints.

EXEMPLO DE REMOÇÃO (EXCLUSÃO) DE UMA COLUNA DA TABELA

```
ALTER TABLE T_TESTE_DDL DROP COLUMN DS_OBS2;
```

EXEMPLO DE REMOÇÃO (EXCLUSÃO) DE VÁRIAS COLUNAS DA TABELA

Observação: Utilizar parênteses após a cláusula DROP, sem a utilização da palavra COLUMN.

```
ALTER TABLE T_TESTE_DDL DROP (DS_TESTE, DS_OBS1);
```

Transformação do Modelo para SQL - Comando ALTER TABLE REMOVER COLUNAS E CONSTRAINTS



EXEMPLO DE REMOÇÃO (EXCLUSÃO) DA CONSTRAINT CHAVE PRIMÁRIA

```
ALTER TABLE T_TESTE_DDL DROP CONSTRAINT PK_TESTE_DDL;
```

Observação: Caso a chave primária participe de um relacionamento (vínculo com outra tabela, onde uma chave estrangeira, aponta para esta chave primária), para que a constraint chave primária, possa ser eliminada, será necessário eliminar o vínculo entre as tabelas relacionados, para isso incluimos a opção CASCADE.

```
ALTER TABLE T_TESTE_DDL DROP CONSTRAINT PK_TESTE_DDL CASCADE;
```

Transformação do Modelo para SQL - Comando ALTER TABLE

REMOVER COLUNAS E CONSTRAINTS



EXEMPLO DE REMOÇÃO (EXCLUSÃO) DA CONSTRAINT FOREIGN KEY

```
ALTER TABLE T_TESTE_DDL DROP CONSTRAINT FK_<NOME DA CONSTRAINT>;
```

EXEMPLO DE REMOÇÃO (EXCLUSÃO) DA CONSTRAINT UNIQUE

```
ALTER TABLE T_TESTE_DDL DROP CONSTRAINT UN_TESTE_DDL_NOME;
```

EXEMPLO DE REMOÇÃO (EXCLUSÃO) DA CONSTRAINT CHECK

```
ALTER TABLE T_TESTE_DDL DROP CONSTRAINT CK_<NOME DA CONSTRAINT>;
```

EXEMPLO DE REMOÇÃO (EXCLUSÃO) DA CONSTRAINT DEFAULT

Observação: Na realidade não removemos uma clausula DEFAULT, e sim retornamos o valor DEFAULT para NULL.

```
ALTER TABLE T_TESTE_DDL MODIFY DT_TESTE DEFAULT NULL;
```

Transformação do Modelo para SQL - Comando ALTER TABLE

RENOMEAR TABELAS, COLUNAS E CONSTRAINTS



Utilizando o comando ALTER TABLE com a cláusula **RENAME**, podemos renomear (alterar o nome) tabelas, colunas e constraints.

EXEMPLO PARA RENOMEAR UMA TABELA

```
ALTER TABLE T_TESTE_DDL RENAME TO T_TESTE_DDL_NOVO;
```

EXEMPLO PARA RENOMEAR UMA COLUNA

```
ALTER TABLE T_TESTE_DDL_NOVO RENAME COLUMN DS_EMAIL TO DS_EMAIL_NOVO;
```

NOME ATUAL

NOME NOVO

EXEMPLO PARA RENOMEAR UMA CONSTRAINT

```
ALTER TABLE T_TESTE_DDL RENAME CONSTRAINT UN_TESTE_DDL_DESC TO UN_TESTE_DDL_DESC_NOVO;
```

NOME ATUAL

NOME NOVO

Transformação do Modelo para SQL - Comando ALTER TABLE HABILITAR E DESABILITAR RESTRIÇÕES (CONSTRAINTS)



Quando uma constraint é criada, ela “nasce” habilitada, ou seja, pronta para ser utilizada durante transações de inserção/alteração.

Podemos desabilitar (desligar) uma constraint, uma vez desligada, ela não é considerada durante transações de inserção/alteração.

EXEMPLO HABILITAR/REABILITAR UMA CONSTRAINT

```
ALTER TABLE T_TESTE_DDL ENABLE CONSTRAINT UN_TESTE_DDL_DESC_NOVO;
```

EXEMPLO DESABILITAR UMA CONSTRAINT

```
ALTER TABLE T_TESTE_DDL DISABLE CONSTRAINT UN_TESTE_DDL_DESC_NOVO;
```

Sintaxe:

```
ALTER TABLE <NOME_TABELA> ENABLE CONSTRAINT <NOME DA CONSTRAINT>;
```

```
ALTER TABLE <NOME_TABELA> DISABLE CONSTRAINT <NOME DA CONSTRAINT>;
```

Transformação do Modelo para SQL - Comando ALTER TABLE

CRIANDO COMENTÁRIOS ASSOCIADOS A TABELA E COLUNAS



O comando **COMMENT** permite adicionar comentários sobre tabelas e colunas.

Os comentários são armazenados no **dicionários de dados** (*é uma coleção de metadados que contém definições e representações de elementos de dados*) do SGBD, sendo muito útil para enriquecer os descritivos dos **metadados** (*informações sobre os dados*).

EXEMPLO DE COMENTÁRIO SOBRE A TABELA

```
COMMENT ON TABLE T_TESTE_DDL_NOVO IS 'Tabela utilizada para exemplificar comandos DDL';
```

EXEMPLO DE COMENTÁRIO SOBRE A COLUNA

NOME TABELA PONTO NOME COLUNA

```
COMMENT ON COLUMN T_TESTE_DDL_NOVO.NM_TESTE  
IS 'Coluna contendo um nome para exemplificar na tabela TESTE';
```



- MACHADO, Felipe Nery R. Banco de Dados - Projeto e Implementação. Érica, 2004.
- Páginas: 330, 331.
- ELMASRI, R.; NAVATHE, S.B. Sistemas de Banco de Dados: Fundamentos e Aplicações. Pearson, 2005. Páginas: 153, 154.
- Puga, S.; França. Ed.; Goya, M. Banco de Dados – Implementação em SQL, PL/SQL e Oracle 11g. Pearson, 2014.

Copyright © 2022 Profa. Rita de Cássia Rodrigues

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).