9

Criação e Gerenciamento de Tabelas

Objetivos deste Capítulo

- Ao concluir este capítulo, você poderá:
  - Categorizar os principais objetos de banco de dados
  - Examinar a estrutura de tabelas
  - Listar os tipos de dados disponíveis para colunas
  - Criar uma tabela simples
  - Compreender como as constraints são criadas quando uma tabela é criada
  - Descrever o funcionamento de objetos de esquema

9-2

Objetivos deste Capítulo

Esta lição apresenta as instruções DDL (Data Definition Language). Você conhecerá os princípios básicos de como criar, alterar e remover tabelas simples. São mostrados os tipos de dados disponíveis em DDL e são apresentados conceitos de esquema. Esta lição também aborda constraints. As mensagens de exceção geradas pela violação de constraints durante as operações DML são mostradas e explicadas.

Principais Objetos do Banco de Dados

- Os principais objetos do banco de dados Oracle são:
  - **Tabela**: é a unidade básica de armazenamento das informações. A tabela é comporta de colunas e linhas.
  - **Visão**: é uma representação lógica de dados de uma ou mais tabelas.
  - **Sequencia**: é um gerador de números. Usado frequentemente para popular uma coluna chave da tabela.
  - **Índice**: é uma estrutura representada por uma ou mais colunas ordenadas que facilita o acesso do Oracle aos dados gravados nas tabelas, e consequentemente melhora o desempenho de algumas consultas.
  - **Sinônimo**: é um apelido que pode ser associado a um objeto do banco de dados.

9-3

Principais Objetos do Banco de Dados

Existem vários tipos de objetos de banco de dados. No projeto do banco de dados, você deve descrever esses objetos para que seja possível criá-los durante a implantação ou atualização do sistema de informações.

Os principais objetos de banco de dados Oracle são:

- **Tabela:** estrutura que armazena os dados.
- **Visão:** subconjunto de dados de uma ou mais tabelas.
- **Seqüência:** gerador números.
- **Índice:** estrutura de colunas ordenadas usada para agilizar o acesso aos dados nas tabelas.
- **Sinônimo:** apelido associado a objetos de banco de dados.

Nomenclatura dos Objetos do Banco de Dados

- Use padrões de nomenclatura dos objetos
- O Oracle não faz distinção de maiúsculas e minúsculas no nome dos objetos de banco de dados
- Regras padrão para nomenclatura de objetos de banco de dados:
  - Devem começar com uma letra
  - Devem ter no máximo 30 caracteres
  - Devem conter apenas os caracteres A a Z, a a z, 0 a 9, _, $ e #
  - Não devem duplicar o nome de outro objeto pertencente ao mesmo usuário
  - Não devem ser palavras reservadas do Oracle

9-4

Regras de Nomeação

É importante que, no desenvolvimento dos sistemas de informação, existam regras para nomenclatura de objetos de bancos de dados. Isso facilita o desenvolvimento e manutenção dos objetos de bancos.

Você deve usar nomes descritivos para tabelas e os outros objetos de banco de dados. Tenham sempre em mente que você pode estar desenvolvendo o banco de dados agora, mas que outras pessoas poderão ter que fazer modificações ou manutenção deles.

Os nomes não fazem distinção entre maiúsculas e minúsculas. Por exemplo, `FUNCIONARIO` é tratado como o mesmo nome que `fUNCIOnaRIO` ou `FuncioNArio`.

Existem regras padrão do Oracle para nomenclatura dos objetos de banco:

- Os nomes de tabelas e colunas devem ter no máximo 30 caracteres.
- Os nomes devem conter apenas os caracteres de A a Z, a a z, 0 a 9, _ (sublinhado), $ e # (caracteres válidos cujo uso não é recomendado).
- Os nomes não devem duplicar o nome de outro objeto pertencente ao mesmo usuário do banco.
- Os nomes não devem ser palavras reservadas do Oracle.

Comando CREATE TABLE

```
CREATE TABLE [esquema.]tabela
          (coluna tipo_de_dado [DEFAULT expressão][, ...]);
```

- Use o comando CREATE TABLE para criar uma tabela.
- Além do nome da tabela, você deve especificar o nome das colunas bem como tipo de dado e tamanho
- É necessário que o usuário possua o privilégio de sistema: CREATE TABLE
- Toda tabela que não é uma tabela temporária ocupa uma área de armazenamento específica (tablespace)

9-5

Comando CREATE TABLE

Use o comando CREATE TABLE para criar tabelas. Esse é um comando DDL (Data Definition Language) e faz parte do conjunto de comandos SQL usados para criar, modificar e remover objetos do banco de dados Oracle. Esses comandos têm efeito imediato sobre o banco de dados, ou seja, não é necessário um commit posterior para efetivar sua execução.

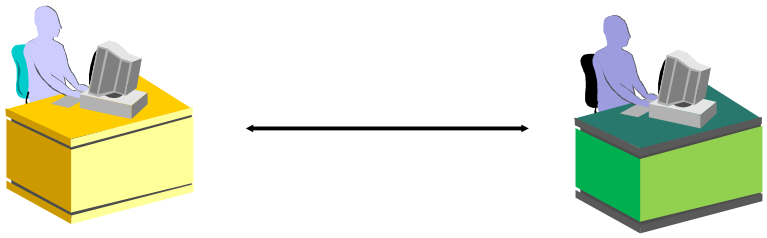
Para criar uma tabela, um usuário deve ter o privilégio de sistema CREATE TABLE e uma área de armazenamento na qual criará a tabela (tablespace).

Na sintaxe:

esquema	é o usuário dono da tabela
tabela	é o nome da tabela que deve ser criada
DEFAULT *expressão*	especifica um valor padrão para quando um valor é omitido no comando INSERT
coluna	é o nome da coluna
tipo_de_dado	é o tipo de dados e o tamanho da coluna

Tabelas de Outro Usuário

- Cada usuário possui seu esquema e os objetos de seu próprio esquema
- Para que um usuário acesse tabelas pertencentes a outros usuários, ele deve usar o nome do dono da tabela como prefixo



USUÁRIO A

```
SELECT *
FROM   usuarioB.funcionario;
```

USUÁRIO B

```
SELECT *
FROM   usuarioA.funcionario;
```

9-6

Tabelas de Outro Usuário

Um *esquema* é o conjunto de objetos que pertencente a um usuário. Os objetos de um esquema são as estruturas lógicas que fazem referência direta aos dados de um banco de dados. Esses objetos incluem tabelas, visões, sinônimos, sequências, procedimentos, índices, etc.

Quando um usuário tenta acessar tabelas de outros usuários, o nome do proprietário da tabela deverá ser incluído como prefixo. Por exemplo, se houver esquemas denominados `UUSUARIOA` e `USUSARIOB` e ambos tiverem uma tabela chamada `FUNCIONARIO`, quando o `USUARIOA` quiser consultar a tabela `FUNCIONARIO` pertencente a `USUARIOB`, ele deverá adicionar o nome do esquema como prefixo ao nome da tabela:

```
SELECT *
FROM   usuariob.funcionario;
```

Se `USUARIOB` quiser consultar a tabela `FUNCIONARIO` pertencente a `USUARIOA`, ele deverá adicionar o nome do esquema como prefixo ao nome da tabela:

```
SELECT *
FROM   usuarioa.funcionario;
```

Opção DEFAULT

- Sintaxe:

```
... data_admissao DATE DEFAULT SYSDATE, ...
```

- O valor padrão é usado se nenhum valor for especificado para a coluna no momento da inserção
- São permitidos valores literais, as expressões e funções SQL
- Não são permitidos nomea de outra coluna ou uma pseudocolunas
- O valor padrão deve corresponder ao tipo de dado da coluna

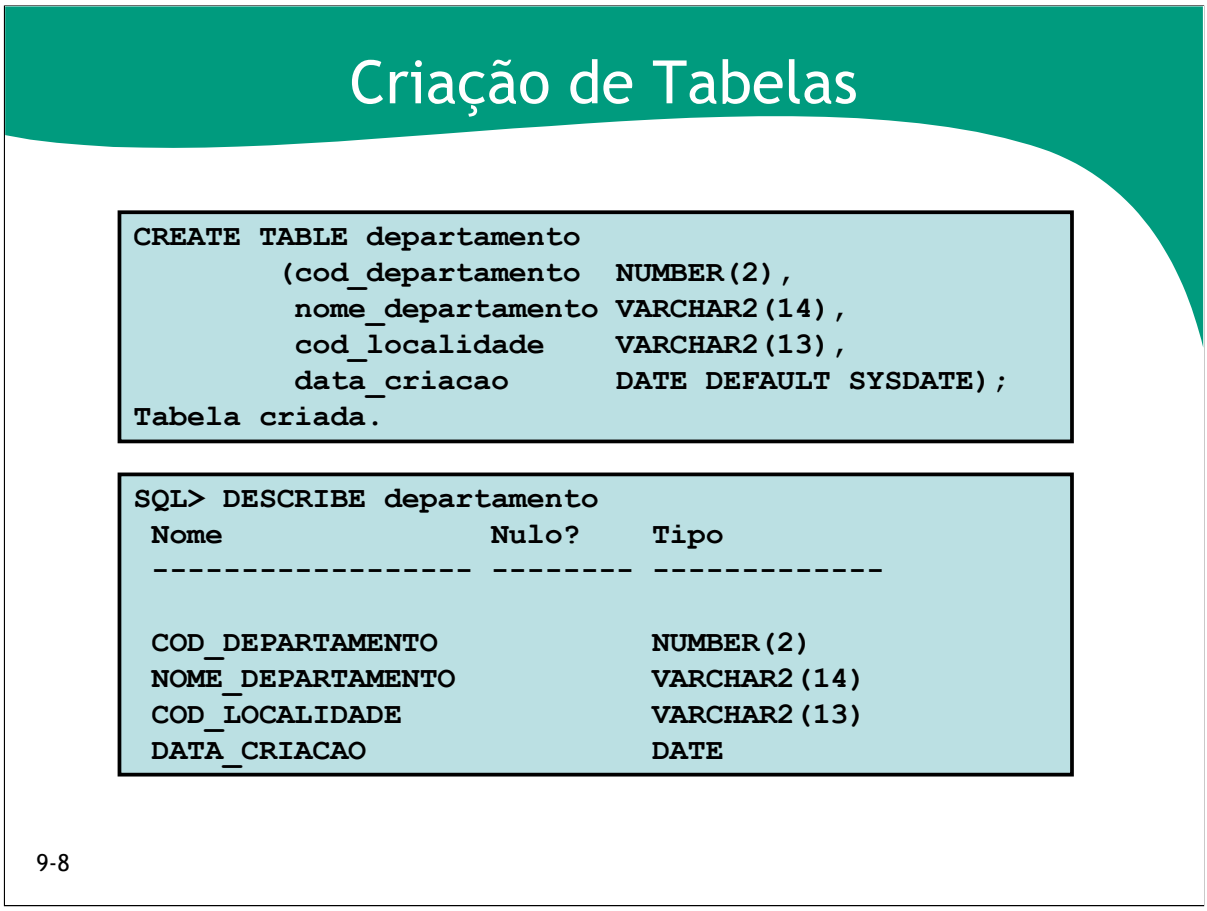
```
CREATE TABLE data_admissao
        (codigo          NUMBER(8),
         data_admissao DATE DEFAULT SYSDATE);
Tabela criada.
```

9-7

Opção DEFAULT

Para especificar que uma coluna possui um valor padrão utilize a opção DEFAULT. Essa opção impedirá que sejam inseridos valores nulos para essa coluna quando uma linha for inserida sem especificar um valor para a coluna. O valor default pode ser um valor literal, uma expressão ou uma função SQL (como SYSDATE ou USER), mas não pode ser o nome de outra coluna ou de uma pseudocoluna (como NEXTVAL ou CURRVAL). A expressão default deve corresponder ao tipo de dados da coluna.

Observação: CURRVAL e NEXTVAL são explicadas posteriormente neste capítulo.



Criação de Tabelas

O exemplo do slide cria a tabela `DEPARTAMENTO` com quatro colunas: `COD_DEPARTAMENTO`, `NOME_DEPARTAMENTO`, `COD_LOCALIDADE` e `DATA_CRIACAO`. A coluna `DATA_CRIACAO` tem um valor padrão. Se nenhum valor for fornecido no comando `INSERT`, a data do banco de dados no momento da execução do comando `INSERT` será inserida automaticamente como valor dessa coluna.

A confirmação da criação da tabela é obtida pela execução do comando `DESCRIBE`.

Como a criação de uma tabela é um comando DDL, ocorre um commit automático imediatamente ao final da execução do comando, ou seja, se antes de executar o comando `CREATE TABLE` o usuário tiver efetuado algum comando `INSERT`, `UPDATE` ou `DELETE` e não tiver feito o `COMMIT`, após a execução do `CREATE TABLE`, o `COMMIT` será feito automaticamente encerrando a transação que estava aberta.

Tipos de Dados

Tipo de Dados	Descrição
VARCHAR2(*tamanho*)	Caractere de tamanho variável
CHAR(tamanho)	Caractere de tamanho fixo
NUMBER(*p*,*s*)	Numéricos de tamanho variável
DATE	Datas e hora (hora, minuto e segundo)
LONG	Caractere de tamanho variável (até 2 GB)
CLOB	Caractere (até 4 GB)
RAW e LONG RAW	Binários
BLOB	Binários (até 4 GB)
BFILE	Binários armazenados em arquivo externo (até 4 GB)
ROWID	Numero em base 64 (representa o endereço único de uma linha na tabela)

9-9

Tipos de Dados

Você precisa fornecer um tipo de dados para uma coluna quando especifica essa coluna na criação da tabela. Existem vários tipos de dados disponíveis:

Tipo de Dados	Descrição
VARCHAR2(*tamanho*)	Caractere de tamanho variável (É necessário especificar um *tamanho* máximo: o *tamanho* mínimo é 1; o *tamanho* máximo é 4.000.)
CHAR [(*tamanho*)]	Caractere com tamanho fixo em bytes (O *tamanho* default e mínimo é 1; o *tamanho* máximo é 2.000.)
NUMBER [(*p*,*s*)]	Número com precisão *p* e escala *s* (A precisão é o número total de dígitos decimais, e a escala é o número de dígitos à direita da vírgula decimal; a precisão pode variar de 1 a 38, e a escala pode variar de -84 a 127.)
DATE	Data e hora até o segundo mais próximo entre 1º de janeiro de 4712 A.C. e 31 de dezembro de 9999 D.C.
LONG	Caractere de tamanho variável (até 2 GB)
CLOB	Caractere (até 4 GB)

Tipos de Dados (continuação)

Tipo de Dados	Descrição
RAW(*size*)	Binários brutos (É necessário especificar um *tamanho* máximo: o *tamanho* máximo é 2.000.)
LONG RAW	Binários brutos de tamanho variável (até 2 GB)
BLOB	Binários (até 4 GB)
BFILE	Binários armazenados em um arquivo externo (até 4 GB)
ROWID	Sistema numérico de base 64 que representa o endereço único de uma linha na tabela

Existem algumas restrições para o uso de uma coluna do tipo LONG:

- Uma coluna LONG não é copiada quando uma tabela é criada com uma subconsulta.
- Não é possível incluir uma coluna LONG em uma cláusula GROUP BY ou ORDER BY.
- Só se pode usar uma coluna LONG por tabela.
- Não é possível definir constraints em uma coluna LONG.

Por isso não é recomendado que se utilize colunas do tipo LONG e sim colunas CLOB.

Outros Tipos de Dados Data/Hora

- Além do tipo de dado DATE, existem outros tipos de dados Data/Hora que podem ser usados:

Tipo de Dados	Descrição
TIMESTAMP	Data com segundos fracionados
INTERVAL YEAR TO MONTH	Diferença entre duas datas onde a parte significativa é o ano e o mês
INTERVAL DAY TO SECOND	Diferença exata entre dois valores do tipo data

9-11

Outros Tipos de Dados Data/Hora

Tipo de Dados	Descrição
TIMESTAMP	Permite que a hora seja armazenada como uma data com segundos fracionados. Existem variações deste tipo de dados.
INTERVAL YEAR TO MONTH	Usado para representar a diferença entre dois valores de data/hora em que apenas o ano e o mês são significativos.
INTERVAL DAY TO SECOND	Usado para representar a diferença precisa entre dois valores de data/hora, pois representa a diferença em dias, horas, minutos e segundos.

O Oracle armazena tipos data num formato interno proprietário, ou seja, o formato que a data é exibida não é o formato que a informação está armazenada no banco de dados. Um dado do tipo data é armazenado em um campo de tamanho fixo de 7 bytes que corresponde ao século, ano, mês, dia, hora, minuto e segundo.

Tipo de Dados TIMESTAMP

- O tipo de dados TIMESTAMP é uma extensão do tipo de dados DATE.
- Ele armazena ano, mês, dia, hora, minuto, segundo e segundo fracionado.

```
TIMESTAMP[(precisão_da_fração_de_segundo)]
```

- Você também pode especificar o fuso horário.

```
TIMESTAMP[(precisão_da_fração_de_segundo)]
WITH TIME ZONE
```

```
TIMESTAMP[(precisão_da_fração_de_segundo)]
WITH LOCAL TIME ZONE
```

9-12

Tipo de Dados TIMESTAMP

O tipo de dados TIMESTAMP é uma extensão do tipo de dados DATE. Ele armazena o ano, o mês, o dia, a hora,o minuto, o segundo e a fração de segundo de acordo com a precisão especificada na criação da coluna.

O valor da precisão da fração de segundo especifica o número de dígitos da parte fracional do segundo e pode ser um número na faixa de 0 a 9. Quando não especificado, a precisão da fração de segundo padrão é 6.

Exemplo

Neste exemplo, a tabela FUNCIONARIO2 é criada com uma coluna DATA_INICIO cujo tipo de dados é TIMESTAMP:

```
CREATE TABLE funcionario2
  (cod_funcionario NUMBER,
   nome            VARCHAR2(15),
   sobrenome       VARCHAR2(15),
   ...
   data_inicio     TIMESTAMP(7),
   ...);
```

Suponha que duas linhas sejam inseridas na tabela FUNCIONARIO2. A saída exibida mostra o resultado.

Tipo de Dados **TIMESTAMP** (continuação)

```
SELECT data_inicio
FROM   funcionario2;

17-JUN-03 00.00.00.000000
21-SET-03 00.00.00.000000
```

O tipo de dados TIMESTAMP WITH TIME ZONE é uma variante de TIMESTAMP que inclui um deslocamento de fuso horário. O deslocamento de fuso horário é a diferença (em horas e minutos) entre a hora local e o UTC (Universal Time Coordinate, conhecido antes como Horário de Greenwich). Esse tipo de dados é usado para coletar e avaliar as informações de data nas regiões geográficas.

Por exemplo:

```
TIMESTAMP '2003-04-15 8:00:00 -8:00'
```

é o mesmo que

```
TIMESTAMP '2003-04-15 11:00:00 -5:00'
```

Isto é, 8:00 a.m. Pacific Standard Time é o mesmo que 11:00 a.m. Eastern Standard Time. Também é possível especificar esse valor desta forma:

```
TIMESTAMP '2003-04-15 8:00:00 US/Pacific'
```

O tipo de dados TIMESTAMP WITH LOCAL TIME ZONE é outra variante de TIMESTAMP que inclui um deslocamento de fuso horário. Ele difere do tipo de dados TIMESTAMP WITH TIME ZONE, pois os dados armazenados no banco de dados são normalizados para o fuso horário do servidor, e o deslocamento de fuso horário não é armazenado como parte dos dados da coluna. Quando os usuários recuperam os dados, eles são retornados no fuso horário da sessão local desses usuários. O deslocamento de fuso horário representa a diferença (em horas e minutos) entre o horário local e o UTC. Diferentemente de TIMESTAMP WITH TIME ZONE, você pode especificar colunas do tipo TIMESTAMP WITH LOCAL TIME ZONE como parte de uma chave primária ou única, como no exemplo a seguir:

```
CREATE TABLE exemplo_hora
   (data_pedido TIMESTAMP WITH LOCAL TIME ZONE);

INSERT INTO exemplo_hora VALUES('15-JAN-04 09:34:28');

SELECT *
FROM   exemplo_hora;

   DATA_PEDIDO
   ---------------------------
   15-JAN-04 09.34.28.000000
```

O tipo TIMESTAMP WITH LOCAL TIME ZONE é apropriado para aplicações de duas camadas nas quais você deseja exibir datas e horários usando o fuso horário do sistema cliente.

Tipo INTERVAL YEAR TO MONTH

- O tipo de dados INTERVAL YEAR TO MONTH armazena uma diferença entre duas datas usando os campos de data/hora YEAR e MONTH:

```
INTERVAL YEAR [(precisão_do_ano)] TO MONTH
```

- O tipo de dados INTERVAL DAY TO SECOND armazena uma diferença entre duas datas em termos de dias, horas, minutos e segundos:

```
INTERVAL DAY [(precisão_do_dia)]
   TO SECOND [(precisão_da_fração_de_segundo)]
```

9-14

Tipo de Dados INTERVAL YEAR TO MONTH

O tipo de dados INTERVAL YEAR TO MONTH armazena um período usando os campos de data/hora YEAR e MONTH.

Esse tipo de dados para representa a diferença entre dois valores data/hora em que apenas o ano e o mês são significativos. Por exemplo, você pode usar esse valor para definir um lembrete relativo a uma data daqui a 120 meses ou testar se já se passaram 6 meses desde uma data específica.

Na sintaxe:

precisão_do_ano	é o número de dígitos no campo de data/hora YEAR. O valor padrão de precisão_do_ano é 2.

Exemplos

- INTERVAL '123-2' YEAR(3) TO MONTH
  Indica um intervalo de 123 anos e 2 meses
- INTERVAL '123' YEAR(3)
  Indica um intervalo de 123 anos e 0 meses
- INTERVAL '300' MONTH(3)
  Indica um intervalo de 300 meses
- INTERVAL '123' YEAR
  Retorna um erro, pois a precisão padrão 2 e 123 tem 3 dígitos

Tipo de Dados `INTERVAL YEAR TO MONTH` (continuação)

```
CREATE TABLE exemplo_hora2
(duracao_emprestimo INTERVAL YEAR (3) TO MONTH);

INSERT INTO exemplo_hora2 (duracao_emprestimo)
  VALUES (INTERVAL '120' MONTH(3));

SELECT TO_CHAR(sysdate+loan_duration, 'dd-mon-yyyy')
FROM   time_example2;
```

Tipo de Dados `INTERVAL DAY TO SECOND`

O tipo de dados `INTERVAL DAY TO SECOND` armazena um período em termos de dias, horas, minutos e segundos.

Utilize esse tipo de dados para representar a diferença precisa entre dois valores de data/horário. Por exemplo, você pode utilizar esse valor para definir um lembrete relativo a um horário daqui a 36 horas ou registrar o horário entre o início e o término de uma corrida. Para representar com alta precisão longos intervalos de tempo, que incluem vários anos, utilize um valor alto para a parte relativa a dias.

Na sintaxe:

`precisão_do_dia`	é o número de dígitos no campo de data/hora `DAY`. Os valores podem ir de 0 a 9. O padrão é 2.
`precisão_da_fração_de_segundo`	é o número de dígitos da parte fracional do campo de data/hora `SECOND`. Os valores aceitos vão de 0 a 9. O padrão é 6.

Exemplos

- `INTERVAL '4 5:12:10.222' DAY TO SECOND(3)`
  Indica 4 dias, 5 horas, 12 minutos, 10 segundos e 222 milésimos de segundo.
- `INTERVAL '180' DAY(3)`
  Indica 180 dias.
- `INTERVAL '4 5:12:10.222' DAY TO SECOND(3)`
  Indica 4 dias, 5 horas, 12 minutos, 10 segundos e 222 milésimos de segundo
- `INTERVAL '4 5:12' DAY TO MINUTE`
  Indica 4 dias, 5 horas e 12 minutos
- `INTERVAL '400 5' DAY(3) TO HOUR`
  Indica 400 dias e 5 horas.
- `INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)`
  Indica 11 horas, 12 minutos e 10,2222222 segundos.

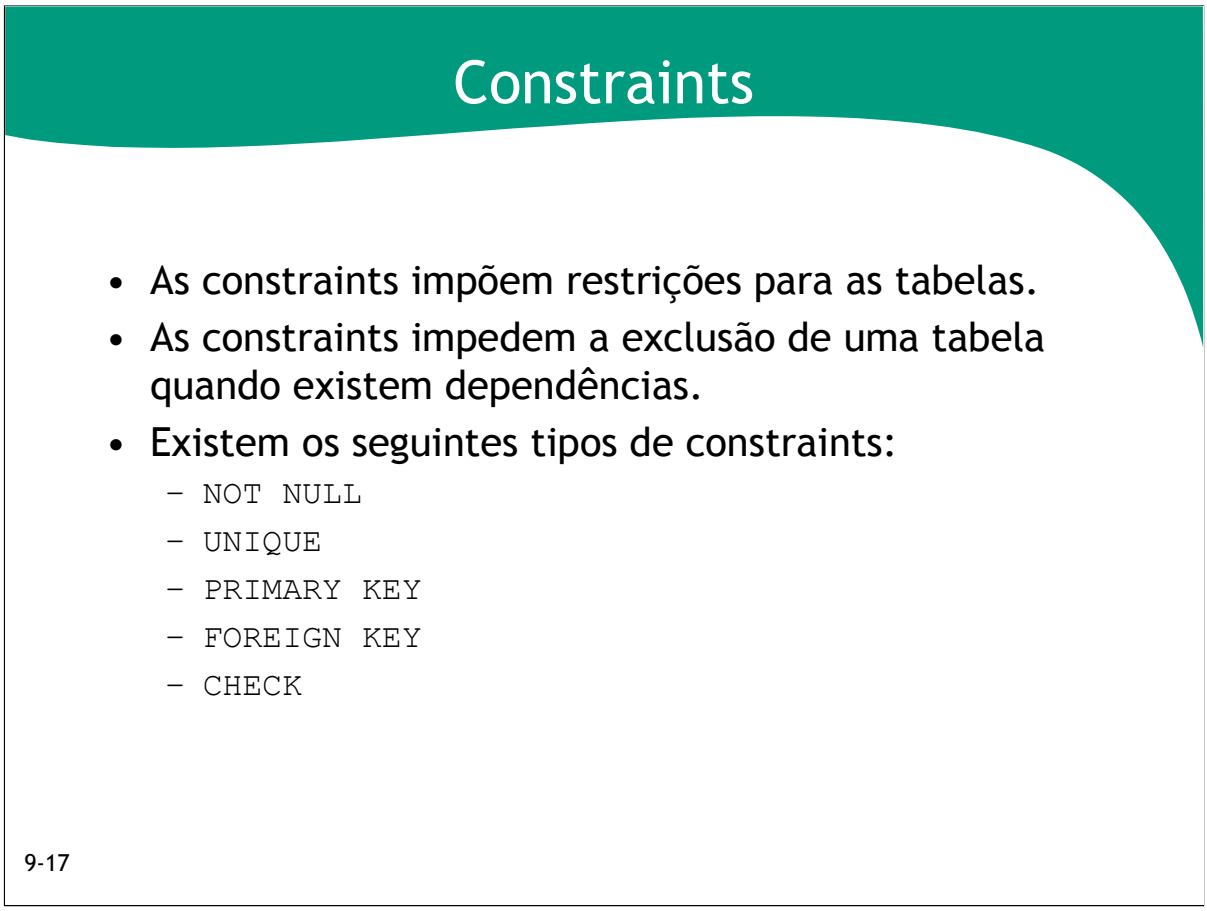
Tipo de Dados **INTERVAL DAY TO SECOND**

Exemplo

```
CREATE TABLE exemplo_hora3
(duracao_dia INTERVAL DAY (3) TO SECOND);

INSERT INTO exemplo_hora3 (duracao_dia)
VALUES (INTERVAL '180' DAY(3));

SELECT sysdate + duracao_dia "Meio Ano"
FROM exemplo_hora3;
```



Constraints

O Oracle usa constraints para impedir a entrada de dados inválidos em tabelas.

Você pode usar constraints para:

- Impor restrições aos dados de uma tabela sempre que uma linha for inserida, atualizada ou excluída dessa tabela. Para que a operação seja bem-sucedida, é necessário obedecer às restrições impostas pelas constraints da tabela.
- Impedir a exclusão de uma tabela quando existem dependências de outras tabelas.
- Fornecer regras para ferramentas como o Oracle Developer.

Constraint	Descrição
NOT NULL	Ccoluna que não pode conter um valor nulo
UNIQUE	Coluna ou a combinação de colunas cujo valores devem ser únicos para todas as linhas na tabela
PRIMARY KEY	Identifica exclusivamente cada linha na tabela
FOREIGN KEY	Impõe um relacionamento de chave estrangeira entre a coluna e uma coluna da tabela referenciada
CHECK	Especifica uma condição que deve ser verdadeira

Diretrizes para Constraints

- Forneça um nome para uma constraint, ou o Oracle irá gerar um nome usando o formato `SYS_C`*n*.
- A constraint pode ser criada durante ou após a criação de uma tabela.
- A constraint pode ser defina para uma coluna ou para a tabela.

9-18

Diretrizes para Constraints

Todas as constraints são objetos armazenados no banco de dados Oracle, e como todo objeto de banco de dados precisa ter um nome. Se você fornecer um nome significativo às constraints, poderá facilitar a referência futura a elas. Os nomes das constraints deverão seguir as regras padrão de nomenclatura. Se você não nomear a constraint, o Oracle gerará um nome com o formato `SYS_C`*n*, em que *n* é um inteiro, para que o nome da constraint seja único.

É possível definir constraints no momento da criação da tabela (comando `CREATE TABLE`) ou depois que a tabela for criada (comando `ALTER TABLE`).

Definição de Constraints

- Sintaxe:

```
CREATE TABLE [esquema.]tabela
      (coluna tipo_de_dado [DEFAULT expressão]
      [constraint_coluna],
      ...
      [constraint_tabela][,...]);
```

- Constraint no nível da coluna:

```
coluna [CONSTRAINT nome_constraint] tipo_constraint,
```

- Constraint no nível da tabela:

```
coluna,...
  [CONSTRAINT nome_constraint] tipo_constraint
  (coluna, ...),
```

9-19

Definição de Constraints

É possível criar as constraints no nível da coluna ou da tabela durante a criação de uma tabela. As constraints definidas no nível da coluna são incluídas quando a coluna é definida. As constraints no nível da tabela são especificadas no final da definição da tabela e devem fazer referência a uma ou mais colunas.

É necessário definir as constraints `NOT NULL` no nível da coluna.

É necessário definir as constraints que se aplicam a mais de uma coluna no nível da tabela.

Na sintaxe:

`esquema`	é igual ao nome do dono da constraint
`tabela`	é o nome da tabela
`DEFAULT expressão`	especifica um valor padrão para quando um valor é omitido no comando `INSERT`
`column`	é o nome da coluna
`tipo_de_dado`	é o tipo de dados e o tamanho da coluna
`constraint_coluna`	é uma constraint que faz parte da definição da coluna
`constraint_tabela`	é uma constraint que faz parte da definição da tabela

Definição de Constraints

- Constraint no nível da coluna:

```
CREATE TABLE funcionario(
  cod_funcionario  NUMBER(6)
    CONSTRAINT func_cod_func_pk PRIMARY KEY,
  nome             VARCHAR2(20),
  ...);
```

1

- Constraint no nível da tabela:

```
CREATE TABLE funcionario(
 cod_funcionario NUMBER(6),
 nome            VARCHAR2(20),
  ...
 cod_cargo       VARCHAR2(10) NOT NULL,
  CONSTRAINT func_cod_func_pk
    PRIMARY KEY (COD_FUNCIONARIO));
```

2

9-20

Definição de Constraints (continuação)

Você pode criar as as constraints junto com criação a tabela. É possível adicionar constraints a uma tabela após a sua criação. Também é possível desativar as constraints temporariamente.

Os dois exemplos do slide criam uma constraint de chave primária na coluna `COD_FUNCIONARIO` da tabela `FUNCIONARIO`.

1. O primeiro exemplo usa a sintaxe no nível da coluna para definir a constraint.
2. O segundo exemplo usa a sintaxe no nível da tabela para definir a constraint.

Constraint NOT NULL

- Não permite que a coluna receba valores nulos:

COD_FUNCIONARIO	SOBRENOME	EMAIL	TELEFONE	DATA_ADMISSAO	COD_CARGO	SALARIO	COD_DEPARTAMENTO
100	Carlos	RCARLOS	051.123.4567	17/06/97	AD_PRES	24000	90
101	Martins	NMARTINS	051.123.4568	21/09/99	AD_VP	17000	90
102	da Silva	LSILVA	051.123.4569	13/01/03	AD_VP	17000	90
103	Honorato	AHONORATO	081.423.4567	03/01/00	IT_PROG	9000	60
104	Osterno	POSTERNO	081.423.4568	21/05/01	IT_PROG	6000	60
176	Voorhees	EVOORHEES	+44.1644.429265	24/03/98	VE_REP	8600	80
178	Yamamura	SYAMAMURA	+44.1644.429263	24/05/99	VE_REP	7000	-
200	Trajano	LTRAJANO	051.123.4444	17/09/97	AD_ASST	4400	10
201	Thacker	WTHACKER	+1.515.123.5555	17/02/06	MK_GER	13000	20
202	Kramer	CKRAMER	+1.603.123.6666	17/08/07	MK_ANA	6000	20
205	Almeida	SALMEIDA	051.123.8080	07/06/04	CT_GER	12000	110
206	Nascimento	RNASCIMENTO	051.123.8181	07/06/04	CTPUB_GER	8300	110

Constraint NOT NULL (Nenhuma linha desta coluna pode conter um valor nulo.)

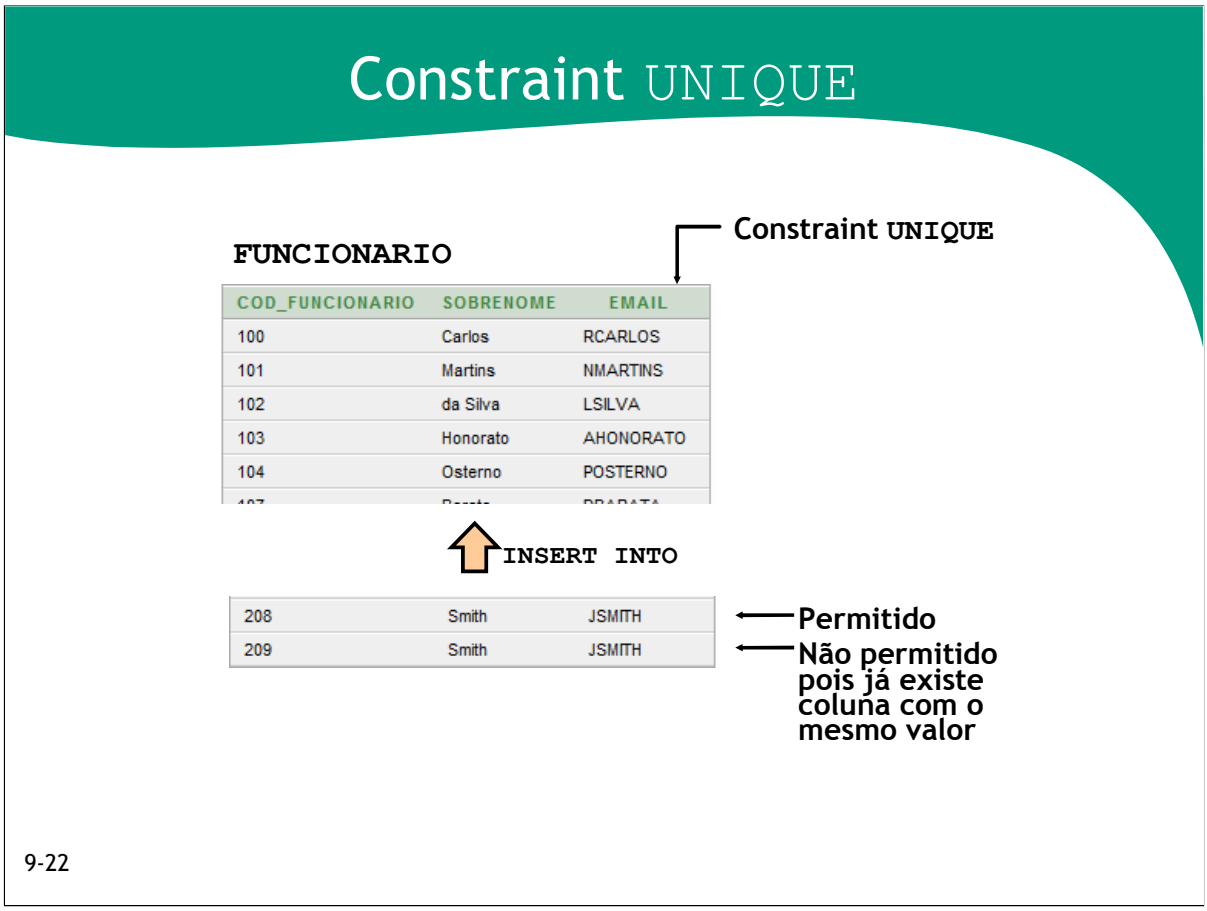
Constraint NOT NULL

Ausência da constraint NOT NULL (Qualquer linha desta coluna pode conter um valor nulo.)

9-21

Constraint NOT NULL

A constraint NOT NULL garante que a coluna não contenha valores nulos. As colunas sem essa constraint podem conter valores nulos por padrão. É necessário definir as constraints NOT NULL no nível da coluna.



Constraint UNIQUE

A constraint UNIQUE exige que todos os valores de uma coluna ou conjunto de colunas (chave) sejam únicos, ou seja, duas linhas de uma tabela não podem conter valores duplicados para essa coluna ou conjunto de colunas. A coluna (ou o conjunto de colunas) incluída na definição da constraint UNIQUE é denominada *chave exclusiva*. Se a constraint UNIQUE incluir mais de uma coluna, esse grupo de colunas será denominado *chave exclusiva composta*.

As constraints UNIQUE permitirão a entrada de valores nulos, a menos que você também defina constraints NOT NULL para as mesmas colunas. Na verdade, qualquer linha de coluna sem a constraint NOT NULL pode incluir valores nulos, já que os valores nulos não são considerados iguais a nenhum valor. Um valor nulo em uma coluna (ou em todas as colunas da chave exclusiva composta) sempre atende a uma constraint UNIQUE.

Observação: Por conta do mecanismo de pesquisa de constraints UNIQUE em mais de uma coluna, você não pode atribuir valores idênticos nas colunas não nulas de uma chave exclusiva composta parcialmente nula.

Constraint UNIQUE

```
CREATE TABLE funcionario(
    cod_funcionario NUMBER(6),
    nome            VARCHAR2(25) NOT NULL,
    email           VARCHAR2(25),
    salario         NUMBER(8,2),
    comissao        NUMBER(2,2),
    data_admissao   DATE NOT NULL,
...
    CONSTRAINT func_email_uk UNIQUE(email));
```

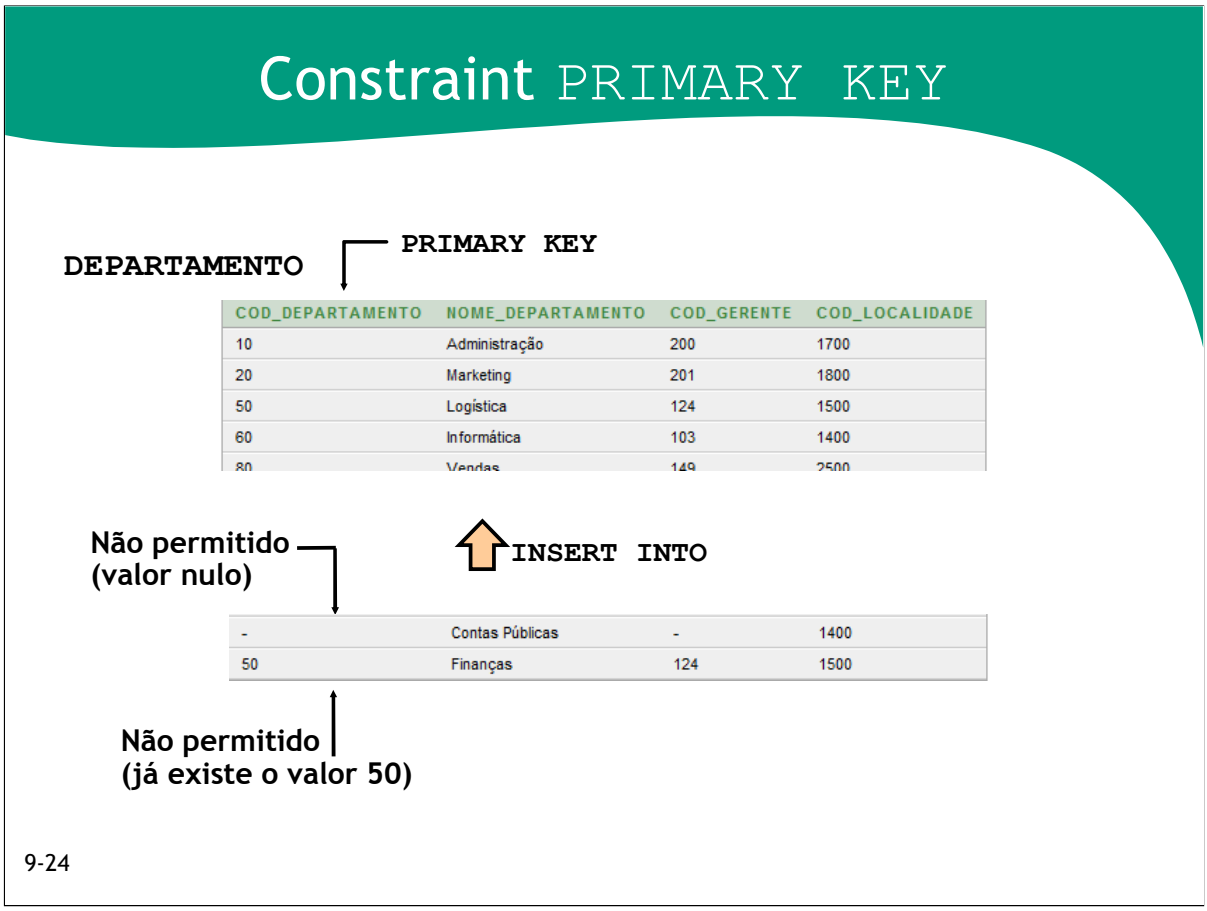
9-23

Constraint UNIQUE (continuação)

É possível definir constraints UNIQUE no nível da coluna ou da tabela. Uma chave exclusiva composta é criada por meio da definição no nível da tabela.

O exemplo do slide aplica a constraint UNIQUE à coluna EMAIL da tabela FUNCIONARIO. O nome da constraint é FUNC_EMAIL_UK.

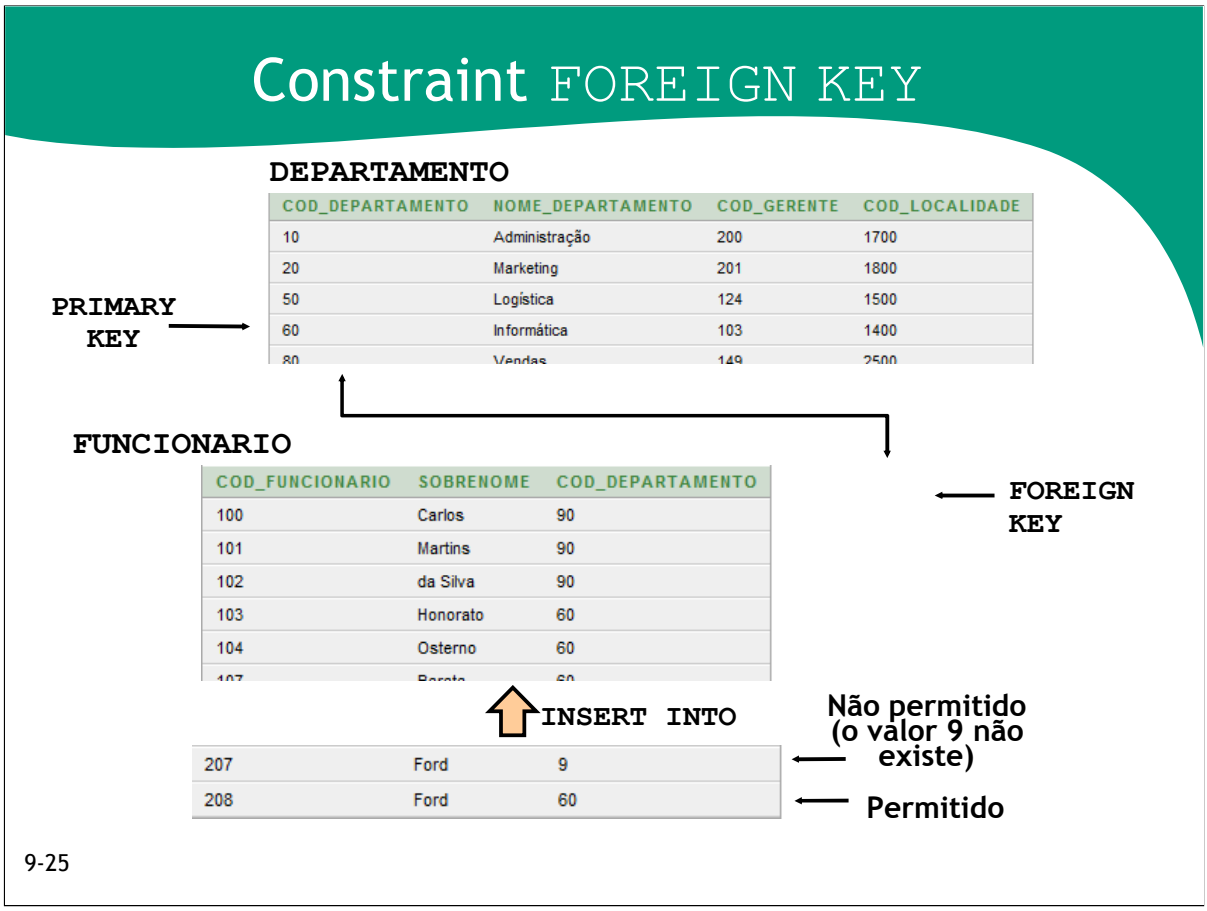
Observação: O servidor Oracle impõe a constraint UNIQUE criando implicitamente um índice exclusivo na(s) coluna(s) de chave exclusiva.



Constraint `PRIMARY KEY`

A constraint `PRIMARY KEY` cria uma chave primária para a tabela. Só é possível criar uma única chave primária para cada tabela. A constraint `PRIMARY KEY` é uma coluna ou um conjunto de colunas que identifica com exclusividade cada linha de uma tabela. Essa constraint impõe a exclusividade da coluna ou da combinação de colunas e garante que nenhuma coluna que faça parte da chave primária possa conter um valor nulo.

Observação: Como a exclusividade faz parte da definição de constraint de chave primária, o servidor Oracle impõe a exclusividade criando implicitamente um índice exclusivo na(s) coluna(s) de chave primária.

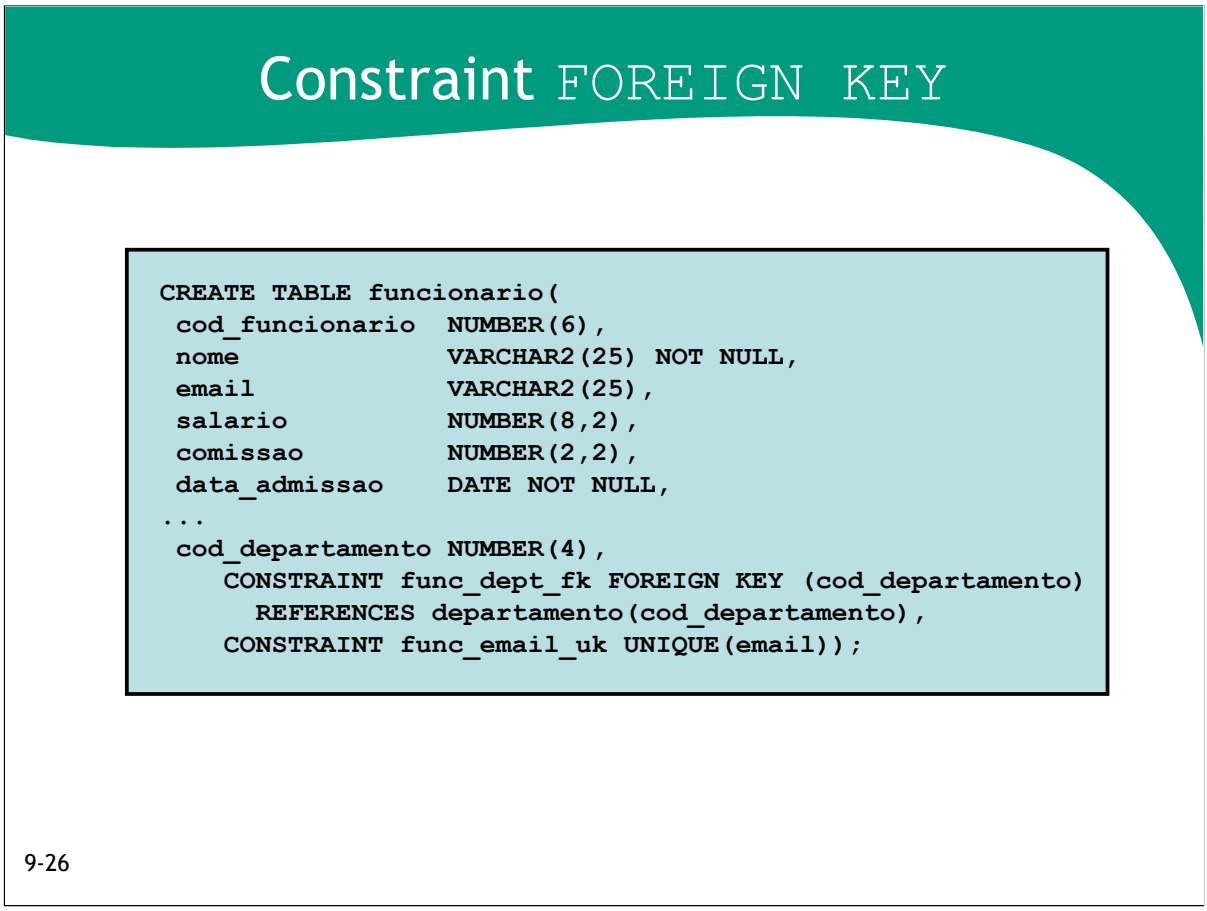


Constraint `FOREIGN KEY`

A constraint `FOREIGN KEY` (ou constraint de integridade referencial) designa uma coluna ou uma combinação de colunas como chave estrangeira e estabelece um relacionamento dessa(s) coluna(s) com uma chave primária ou exclusiva na mesma ou em outra tabela.

No exemplo do slide, a coluna `COD_DEPARTAMENTO` foi definida como a chave estrangeira da tabela `FUNCIONARIO` (tabela filha ou dependente); ela faz referência à coluna `COD_DEPARTAMENTO` da tabela `DEPARTAMENTO` (tabela pai ou referenciada).

Um valor de chave estrangeira deve corresponder a um valor existente na tabela pai ou deve ser `NULL`.



Constraint `FOREIGN KEY` (continuação)

É possível definir constraints `FOREIGN KEY` no nível da coluna ou da tabela. Uma chave estrangeira composta deve ser criada por meio da definição no nível da tabela.

O exemplo do slide define uma constraint `FOREIGN KEY` na coluna `COD_DEPARTAMENTO` da tabela `FUNCIONARIO` usando a sintaxe no nível da tabela. O nome da constraint é `FUNC_DEPT_FK`.

Também é possível definir a chave estrangeira no nível da coluna, desde que a constraint seja baseada em uma única coluna. A diferença na sintaxe é que as palavras-chave `FOREIGN KEY` não aparecem. Por exemplo:

```
CREATE TABLE funcionario
(...
cod_departamento NUMBER(4)
CONSTRAINT func_dept_fk
REFERENCES departamento(cod_departamento),
...
)
```

Palavras-chave da FOREIGN KEY

- FOREIGN KEY: Define a coluna da tabela filha no nível de constraint da tabela
- REFERENCES: Identifica a tabela e a coluna da tabela pai
- ON DELETE CASCADE: Exclui as linhas dependentes da tabela filha quando uma linha da tabela pai é excluída
- ON DELETE SET NULL: Converte os valores da chave estrangeira dependente em nulos

9-27

Palavras-chave da Constraint FOREIGN KEY

A chave estrangeira é definida na tabela filha e a tabela que contém a coluna referenciada é a tabela pai. Essa chave é definida por meio de uma combinação das seguintes palavras-chave:

- FOREIGN KEY é usada para definir a coluna da tabela filha no nível de constraint de tabela.
- REFERENCES identifica a tabela e a coluna da tabela pai.
- ON DELETE CASCADE indica que, quando a linha da tabela pai é excluída, as linhas dependentes da tabela filha também são deletadas.
- ON DELETE SET NULL converte os valores da chave estrangeira em nulos quando o valor pai é removido.

Sem a opção ON DELETE CASCADE ou ON DELETE SET NULL, não será possível excluir ou alterar a linha da tabela pai se for feita referência a essa linha a partir da tabela filha.

Constraint CHECK

- Define uma condição que cada linha deve atender
- As seguintes expressões não são permitidas:
  - Referências às pseudocolunas CURRVAL, NEXTVAL, LEVEL e ROWNUM
  - Chamadas de funções SYSDATE, UID, USER e USERENV
  - Consultas que fazem referência a outros valores em outras linhas

```
..., salario  NUMBER(2)
     CONSTRAINT func_salario_minimo_ck
            CHECK (salario > 460),...
```

9-28

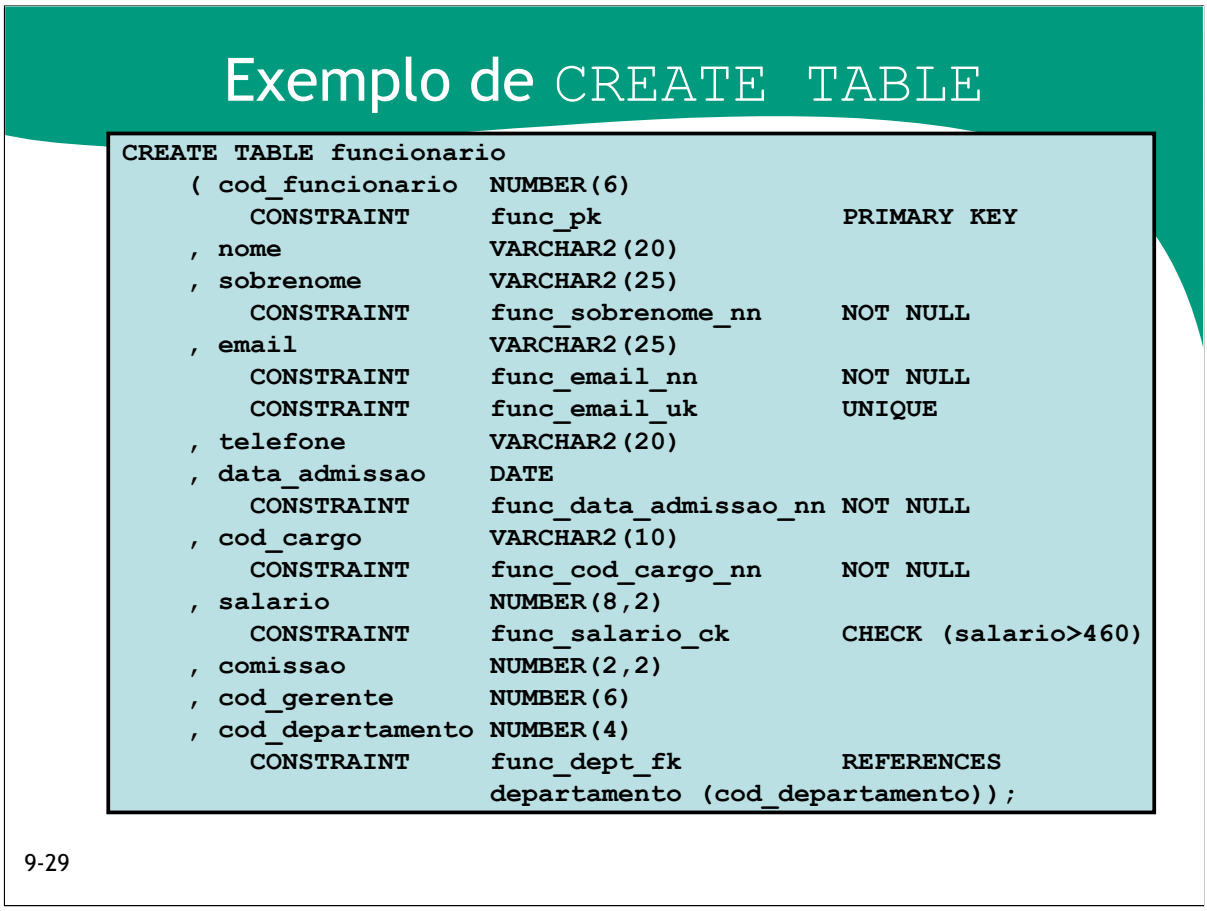
Constraint CHECK

A constraint CHECK define uma condição que cada linha deve atender. A condição de CHECK pode usar a mesma construção que as condições de uma consulta, com as seguintes exceções:

- Referências às pseudocolunas CURRVAL, NEXTVAL, LEVEL e ROWNUM
- Chamadas de funções SYSDATE, UID, USER e USERENV
- Consultas que fazem referência a outros valores em outras linhas

Não há limite ao número de constraints CHECK que você pode definir em uma coluna.

É possível definir constraints CHECK no nível da coluna ou da tabela.



Exemplo de `CREATE TABLE`

O exemplo do slide mostra um comando completo usado para criar a tabela `FUNCIONARIO`.

Erro de Constraints de Integridade

```
UPDATE funcionario
SET    cod_departamento = 55
WHERE  cod_departamento= 110;
```

```
UPDATE funcionario
*
ERRO na linha 1:
ORA-02291: restrição de integridade
(PROFESSOR.FUNCIONARIO_DEPARTAMENTO_FK)
violada - chave mãe não localizada
```

- O departamento 55 não existe.

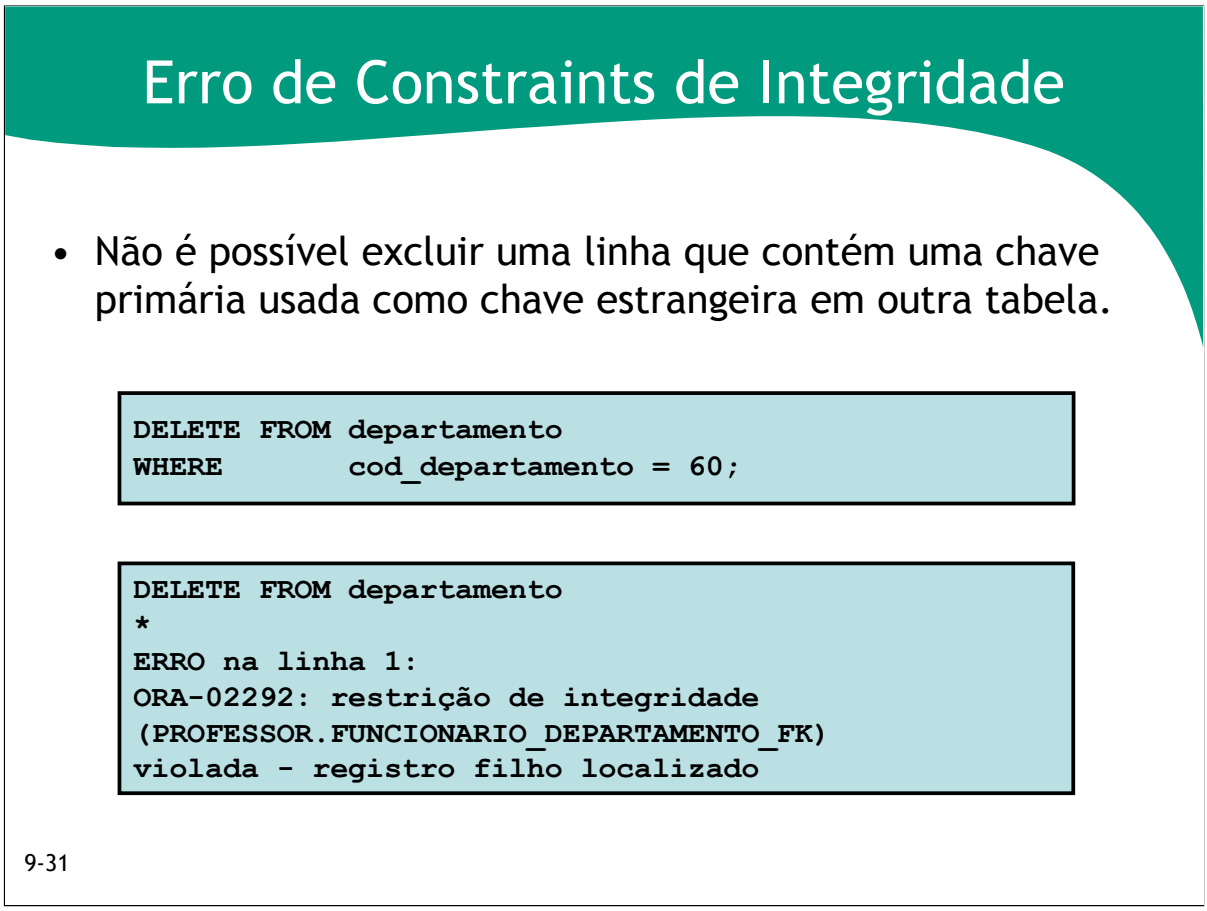
9-30

Erro de Constraints de Integridade

Quando há constraints em colunas, um erro será exibido se você tentar violar a regra da constraint.

Por exemplo, se você tentar atualizar um registro com um valor vinculado a uma constraint de integridade, um erro será exibido.

No exemplo do slide, como o departamento 55 não existe na tabela pai `DEPARTAMENTO`, é exibido o erro ORA-02291 de violação de *chave pai*.



Erro de Constraint de Integridade (continuação)

Se você tentar excluir um registro com um valor vinculado a uma constraint de integridade, um erro será exibido.

O exemplo do slide tenta excluir o departamento 60 da tabela `DEPARTAMENTO`, mas ocorre um erro porque o código do departamento é usado como chave estrangeira na tabela `FUNCIONARIO`. Se o registro pai que você tenta excluir tiver registros filhos, será exibido o erro `ORA-02292` de violação de *registro filho encontrado*.

Este comando funciona porque não há funcionários no departamento 70:

```
DELETE FROM departamento
WHERE       cod_departamento = 70;

1 linha deletada.
```

Criação de Tabela com uma Subconsulta

- Crie uma tabela e insira linhas combinando a instrução `CREATE TABLE` e a opção de subconsulta `AS`.

```
CREATE TABLE tabela
           [(coluna, coluna...)]
AS subconsulta;
```

- Estabeleça uma correspondência entre as colunas especificadas e o número de colunas da subconsulta.
- Defina colunas com nomes e valores padrão.

9-32

Criação de Tabela com uma Subconsulta

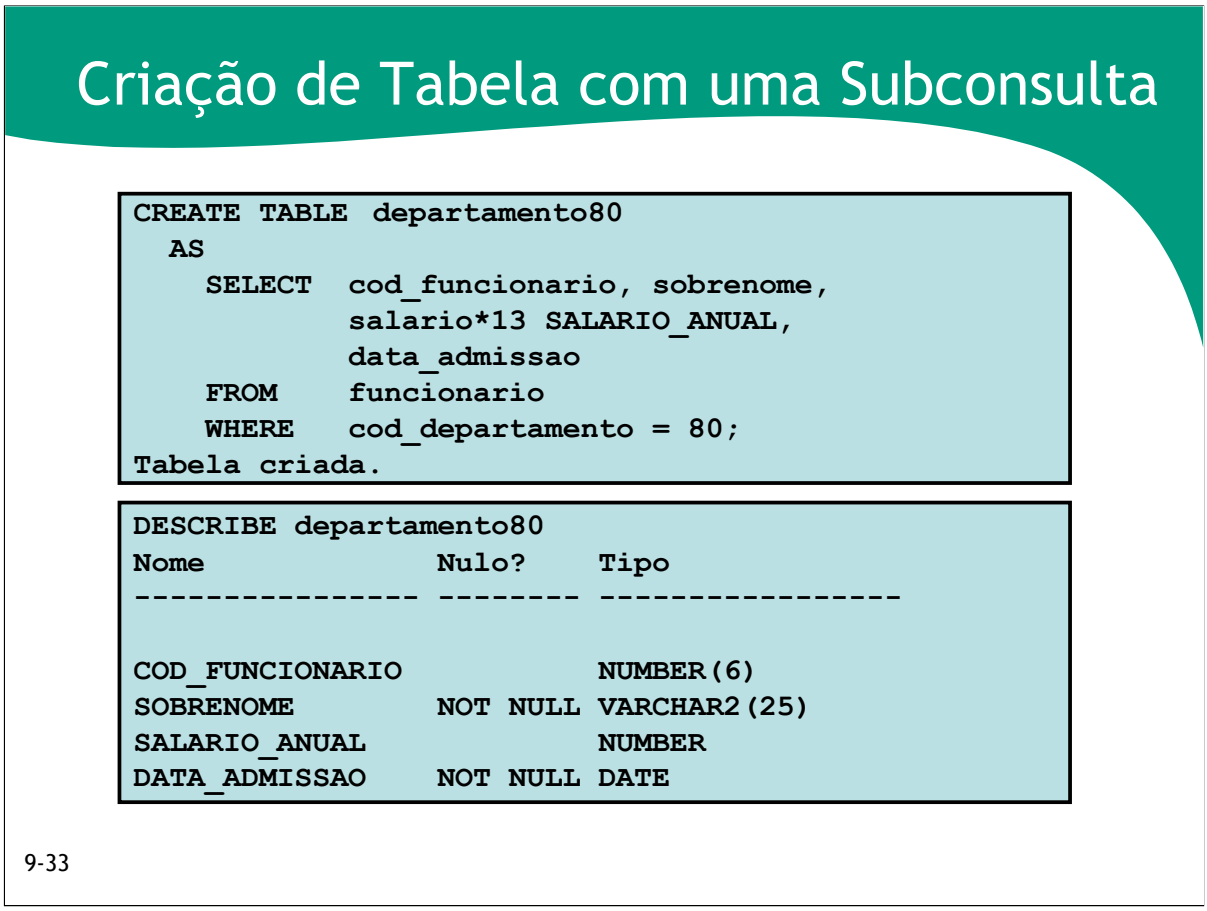
Um segundo método para criar uma tabela é aplicar a cláusula de subconsulta `AS`, que cria a tabela e insere linhas retornadas da subconsulta.

Na sintaxe:

tabela	é o nome da tabela
coluna	é o nome da coluna, o valor padrão e a constraint de integridade
subconsulta	é o comando `SELECT` que define o conjunto de linhas a ser inserido na nova tabela

Diretrizes

- A tabela é criada com os nomes de colunas especificados, e as linhas recuperadas pelo comando `SELECT` são inseridas nessa tabela.
- A definição da coluna só poderá conter o nome da coluna e o valor padrão.
- Se forem fornecidas especificações de colunas, o número de colunas deverá ser igual ao número de colunas da lista da subconsulta `SELECT`.
- Se não forem fornecidas especificações de colunas, os nomes das colunas da tabela serão iguais aos nomes das colunas incluídos na subconsulta.
- As regras de integridade não serão passadas para a nova tabela, somente as definições de tipos de dados das colunas.



Criação de Tabela com uma Subconsulta (continuação)

O exemplo do slide cria uma tabela denominada `DEPARTAMENTO80`, que contém detalhes de todos os funcionários que trabalham no departamento 80. Observe que os dados da tabela `DEPARTAMENTO80` originam-se da tabela `FUNCIONARIO`.

Você pode verificar a existência de uma tabela de banco de dados e as definições das colunas com o comando SQL*Plus `DESCRIBE`.

Forneça um apelido de coluna quando selecionar uma expressão. A expressão `SALARIO*13` recebe o apelido `SALARIO_ANUAL`. Sem o apelido, o seguinte erro é gerado:

```
   ERRO na linha 3:
ORA-00998: esta expressão deve ser nomeada com um apelido
de coluna
```

Comando ALTER TABLE

- Use o comando ALTER TABLE para adicionar, modificar ou eliminar colunas e também para definir um valor padrão para a coluna

```
ALTER TABLE tabela
ADD         (coluna tipo_de_dado [DEFAULT expressão]
            [, coluna tipo_de_dado]...);
```

```
ALTER TABLE tabela
MODIFY      (coluna tipo_de_dado [DEFAULT expressão]
            [, coluna tipo_de_dado]...);
```

```
ALTER TABLE tabela
DROP        (coluna);
```

9-34

Comando ALTER TABLE

Após criar uma tabela, você talvez precise alterar sua estrutura por uma destas razões:

- Para acrescentar uma coluna omitida.
- Para alterar a definição de uma coluna.
- Para remover colunas.

Para fazer isso, use o comando ALTER TABLE.

Na sintaxe:

tabela	é o nome da tabela
ADD|MODIFY|DROP	é o tipo de modificação
Coluna	é o nome da nova coluna
tipo_de_dado	é o tipo de dados e o tamanho da nova coluna
DEFAULT *expressão*	especifica o valor padrão para uma nova coluna

Adicionar uma Coluna

- Use a cláusula `ADD` para adicionar colunas.

```
ALTER TABLE departamento80
ADD         (cod_cargo VARCHAR2(9));
Tabela alterada.
```

- A nova coluna será a última coluna.

COD_FUNCIONARIO	SOBRENOME	SALARIO_ANUAL	DATA_ADMISSAO	COD_CARGO
149	Schultz	136500	29/01/00	-
174	Sue	143000	11/05/06	-
176	Voorhees	111800	24/03/98	-

3 linhas retornadas em 0,03 segundos

9-35

Adicionar uma Coluna

Você pode adicionar ou modificar colunas. Não é possível especificar onde a coluna será exibida. A nova coluna será a última coluna.

O exemplo do slide adiciona uma coluna denominada `COD_CARGO` à tabela `DEPARTAMENTO80`. A coluna `COD_CARGA` torna-se a última coluna da tabela.

Se a tabela já contiver linhas quando uma coluna for adicionada, inicialmente, todas as linhas da nova coluna terão valores nulos. Não é possível adicionar uma coluna `NOT NULL` obrigatória a uma tabela que contém dados nas outras colunas. Só é possível adicionar uma coluna `NOT NULL` a uma tabela vazia.

Alteração de Coluna

- Você pode alterar o tipo de dados, o tamanho e o valor padrão de uma coluna.

```
ALTER TABLE departamento80
MODIFY      (sobrenome VARCHAR2(30));
Tabela alterada.
```

- Uma alteração no valor padrão afeta apenas as inserções subseqüentes na tabela.

9-36

Alteração de Coluna

Você pode alterar a definição de uma coluna usando o comando `ALTER TABLE` com a cláusula `MODIFY`. A alteração de uma coluna pode ser uma alteração do tipo de dados, tamanho e valor padrão da coluna.

Você pode aumentar a largura ou a precisão de uma coluna numérica.

É possível aumentar a largura de colunas numéricas ou de caracteres.

Você pode diminuir a largura de uma coluna se:

- A coluna contiver apenas valores nulos
- A tabela não tiver nenhuma linha
- A redução na largura da coluna não for inferior aos valores existentes nessa coluna

Você poderá alterar o tipo de dados se a coluna contiver apenas valores nulos. A exceção a essa regra é a conversão de `CHAR` em `VARCHAR2`, que pode ser feita com dados contidos nas colunas.

Você só poderá converter uma coluna com o tipo de dados `CHAR` em `VARCHAR2` ou vice-versa se a coluna contiver valores nulos ou se seu tamanho não for alterado.

Uma alteração no valor padrão de uma coluna afeta apenas as inserções subseqüentes na tabela.

Exclusão de Coluna

- Use a cláusula `DROP COLUMN` para excluir as colunas que não são mais necessárias da tabela.

```
ALTER TABLE  departamento80
DROP COLUMN  cod_cargo;
Tabela alterada.
```

COD_FUNCIONARIO	SOBRENOME	SALARIO_ANUAL	DATA_ADMISSAO
149	Schultz	136500	29/01/00
174	Sue	143000	11/05/06
176	Voorhees	111800	24/03/98

3 linhas retornadas em 0,00 segundos

9-37

Exclusão de Coluna

Você pode excluir uma coluna de uma tabela usando o comando `ALTER TABLE` com a cláusula `DROP COLUMN`.

A coluna pode ou não conter dados.

Com o comando `ALTER TABLE`, só é possível eliminar uma coluna por vez.

Não é possível excluir a última coluna de uma tabela nem recuperar a coluna depois de excluída.

Não é possível eliminar uma coluna se ela fizer parte de uma constraint ou de uma chave de índice, a menos que a opção `CASCADE CONSTRAINTS` seja usada.

A exclusão de uma coluna pode demorar um pouco quando a tabela tem muitos registros. Nesse caso, talvez seja melhor defini-la como não utilizada e excluí-la quando o número de usuários usando o sistema diminuir, a fim de evitar bloqueios demorados na tabela.

Algumas colunas não podem ser excluídas. Este é o caso das colunas que fazem parte da chave de particionamento de uma tabela particionada ou das colunas que fazem parte da chave primária de uma tabela organizada por índice.

A Opção SET UNUSED

- Use a opção SET UNUSED para marcar uma ou mais colunas como não utilizadas.
- Use a opção DROP UNUSED COLUMNS para remover todas as colunas marcadas como não utilizadas de uma só vez.

```
ALTER TABLE  <tabela>
SET   UNUSED(<coluna>);
OU
ALTER TABLE  <tabela>
SET   UNUSED COLUMN <coluna>;
```

```
ALTER TABLE <tabela>
DROP  UNUSED COLUMNS;
```

9-38

A Opção SET UNUSED

A opção SET UNUSED marca uma ou mais colunas como não utilizadas de forma que elas possam ser excluídas quando a demanda de recursos do sistema for menor. A especificação dessa cláusula, na verdade, não remove as colunas de destino de cada linha da tabela (isto é, não libera o espaço em disco usado por essas colunas). Portanto, o tempo de resposta é mais rápido do que se você executasse a cláusula DROP. As colunas não utilizadas são tratadas como se tivessem sido excluídas, embora os respectivos dados permaneçam nas linhas da tabela. Você não terá mais acesso a uma coluna depois que ela for marcada como não utilizada. Uma consulta SELECT * não recuperará dados de colunas não utilizadas. Além disso, os nomes e os tipos de colunas marcadas como não utilizadas não serão exibidos durante um comando DESCRIBE, e você poderá adicionar uma nova coluna à tabela com o mesmo nome da coluna não utilizada. As informações da opção SET UNUSED são armazenadas na visão de dicionário de dados USER_UNUSED_COL_TABS.

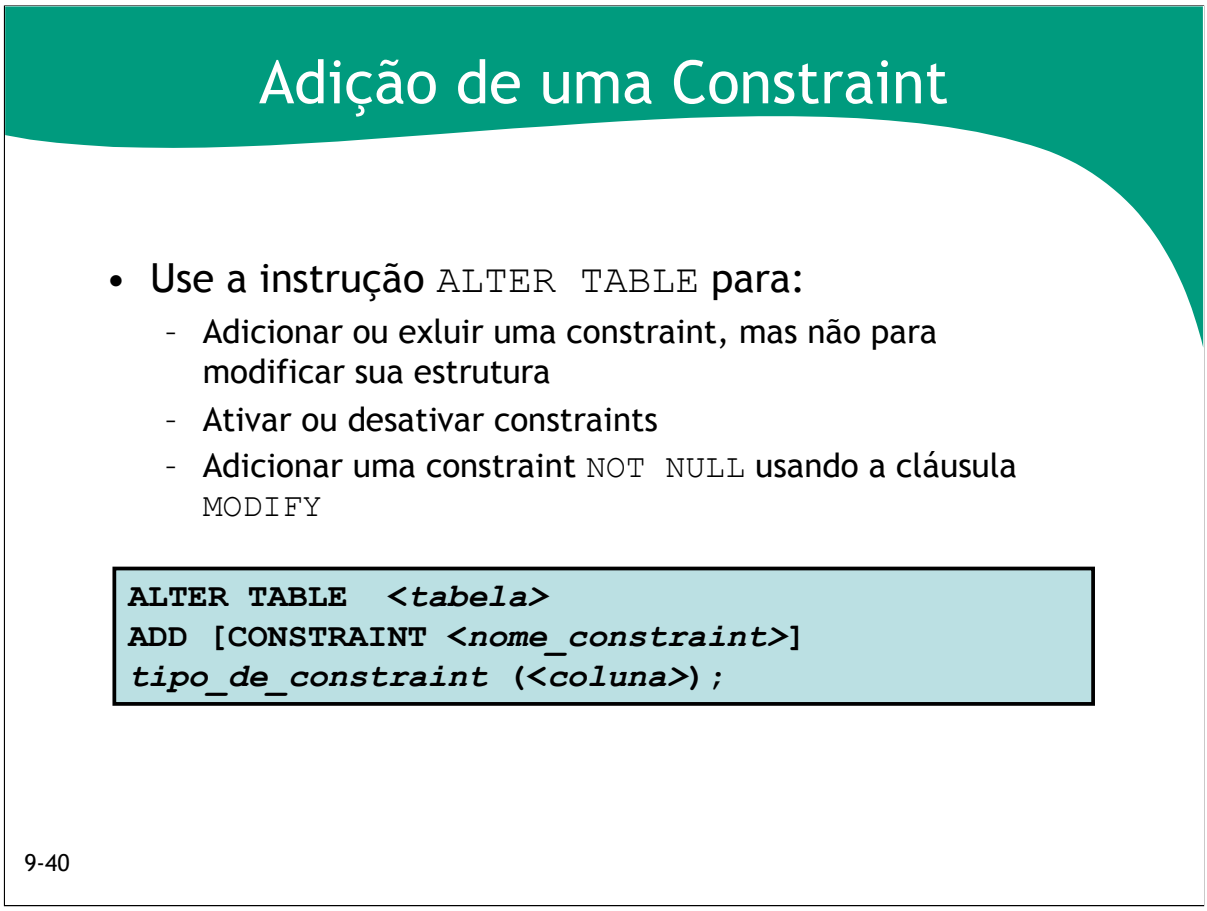
As diretrizes para definir colunas como UNUSED são semelhantes às usadas para excluí-las.

A Opção DROP UNUSED COLUMNS

A opção DROP UNUSED COLUMNS remove da tabela todas as colunas marcadas como não utilizadas de uma única vez. Você poderá usar essa opção quando quiser liberar o espaço extra em disco de colunas não utilizadas da tabela. Se a tabela não contiver colunas não utilizadas, o comando não retornará erros.

```
ALTER TABLE  departamento80
SET   UNUSED (sobrenome);
Tabela alterada.

ALTER TABLE  departamento80
DROP  UNUSED COLUMNS;
Tabela alterada.
```



Adição de uma Constraint

Você pode adicionar uma constraint a tabelas existentes usando o comando `ALTER TABLE` com a cláusula `ADD`.

Na sintaxe:

tabela	é o nome da tabela
nome_constraint	é o nome da constraint
tipo_de_constraint	é o tipo de constraint
coluna	é o nome da coluna afetada pela constraint

A sintaxe de nome de constraint é opcional, embora recomendada. Se você não nomear suas constraints, o sistema gerará nomes para elas.

Você pode adicionar, excluir, ativar ou desativar uma constraint, mas não pode modificar sua estrutura.

Você pode adicionar uma constraint `NOT NULL` a uma coluna existente usando a cláusula `MODIFY` da instrução `ALTER TABLE`.

Você só poderá definir uma coluna `NOT NULL` se a tabela estiver vazia ou se a coluna tiver um valor para cada linha.

Adição de uma Constraint

- Adicione uma constraint `FOREIGN KEY` à tabela `FUNCIONARIO2` para indicar que já deve existir um gerente como funcionário válido nessa tabela.

```
ALTER TABLE funcionario2
MODIFY cod_funcionario PRIMARY KEY;
Tabela alterada.
```

```
ALTER TABLE funcionario2
ADD CONSTRAINT func_gerente_fk
  FOREIGN KEY(cod_gerente)
  REFERENCES funcionario2(cod_funcionario);
Tabela alterada.
```

9-41

Adição de uma Constraint (continuação)

O primeiro exemplo do slide modifica a tabela `FUNCIONARIO2` para adicionar uma constraint `PRIMARY KEY` à coluna `COD_FUNCIONARIO`. Observe que, como não foi especificado nenhum nome de constraint, a constraint é automaticamente nomeada pelo servidor Oracle.

O segundo exemplo do slide cria uma constraint `FOREIGN KEY` na tabela `FUNCIONARIO2`. A constraint garante que exista um gerente na condição de funcionário válido na tabela `FUNCIONARIO2`.

ON DELETE CASCADE

- Quando registros da tabela pai forem excluídos e existirem registros filhos que dependam deles, esses registros filhos serão excluídos em cascata.

```
ALTER TABLE funcionario2 ADD CONSTRAINT func_dept_fk
FOREIGN KEY (cod_departamento)
REFERENCES departamento ON DELETE CASCADE;
Tabela alterada.
```

9-42

ON DELETE CASCADE

A ação `ON DELETE CASCADE` permite a exclusão, mas não a atualização, de dados da chave pai referenciados pela tabela filha. Quando os dados da chave pai são excluídos, também são excluídos todas as linhas da tabela filha que dependem dos valores excluídos da tabela pai. Para especificar isso inclua a opção `ON DELETE CASCADE` na definição da constraint `FOREIGN KEY`.

Exclusão de uma Constraint

- Exclua a constraint func_gerente_fk da tabela FUNCIONARIO2.

```
ALTER TABLE funcionario2
DROP CONSTRAINT func_gerente_fk;
Tabela alterada.
```

- Exclua a constraint PRIMARY KEY da tabela DEPARTAMENTO2 e exclua a constraint FOREIGN KEY associada da coluna FUNCIONARIO2.COD_DEPARTAMENTO.

```
ALTER TABLE departamento2
DROP PRIMARY KEY CASCADE;
Tabela alterada.
```

9-43

Exclusão de uma Constraint

Para excluir uma constraint, você pode usar o comando ALTER TABLE com a cláusula DROP. A opção CASCADE da cláusula DROP exclui também as constraints dependentes.

Sintaxe

```
ALTER TABLE tabela
DROP  PRIMARY KEY | UNIQUE (coluna) |
CONSTRAINT constraint [CASCADE];
```

onde:

tabela	é o nome da tabela
coluna	é o nome da coluna afetada pela constraint
constraint	é o nome da constraint

Desativar Constraints

- A cláusula `DISABLE` do comando `ALTER TABLE` desativa uma constraint.
- Use a opção `CASCADE` para desativar as constraints de integridade dependentes.

```
ALTER TABLE funcionario2
DISABLE CONSTRAINT func_dept_fk;
Tabela alterada.
```

9-44

Desativar Constraints

Você pode desativar uma constraint sem excluí-la, ou recriá-la usando o comando `ALTER TABLE` com a cláusula `DISABLE`.

Sintaxe

```
ALTER TABLE tabela
DISABLE CONSTRAINT constraint [CASCADE];
```

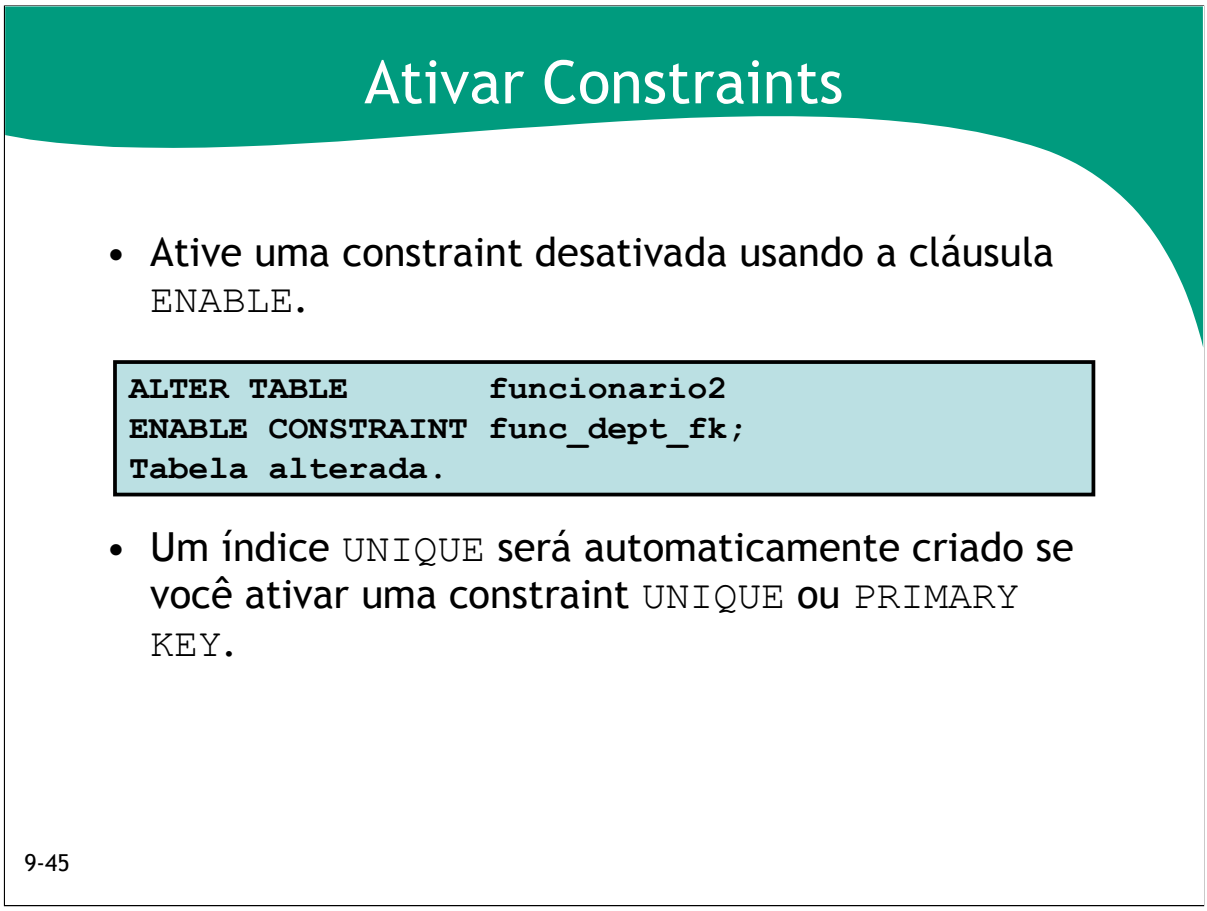
onde:

tabela	é o nome da tabela
constraint	é o nome da constraint

Você pode usar a cláusula `DISABLE` nos comandos `CREATE TABLE` e `ALTER TABLE`.

A cláusula `CASCADE` desativa constraints de integridade dependentes.

A desativação de uma constraint de chave exclusiva ou primária remove o índice exclusivo.



Ativar Constraints

Você pode ativar uma constraint que foi criada e estava desativada usando o comando `ALTER TABLE` com a cláusula `ENABLE`.

Sintaxe

```
ALTER   TABLE      tabela
ENABLE  CONSTRAINT constraint;
```

onde:

tabela	é o nome da tabela
constraint	é o nome da constraint

Se você ativar uma constraint, ela será aplicada a todos os dados da tabela. Todos os dados da tabela devem estar de acordo com a constraint.

Se você ativar uma chave `UNIQUE` ou `PRIMARY KEY`, um índice `UNIQUE` ou `PRIMARY KEY` será criado automaticamente. Caso já exista um índice, ele poderá ser usado por essas chaves.

Você pode usar a cláusula `ENABLE` nos comandos `CREATE TABLE` e `ALTER TABLE`.

Ativar Constraints (continuação)

A ativação de uma constraint de chave primária que tenha sido desativada com a opção `CASCADE` não ativará as chaves estrangeiras dependentes da chave primária.

Para ativar uma constraint de chave `UNIQUE` ou `PRIMARY KEY`, você deve ter os privilégios necessários para criar um índice na tabela.

Exclusão de uma Tabela

- A estrutura e os dados da tabela são excluídos.
- As transações pendentes são submetidas a commit.
- Todos os índices são excluídos.
- Todas as constraints são excluídas.
- Não é possível fazer rollback do comando `DROP TABLE`.

```
DROP TABLE departamento80;
Tabela eliminada.
```

9-47

Exclusão de uma Tabela

O comando `DROP TABLE` remove a tabela e todos os seus registros do banco de dados. Quando você remove uma tabela, além de todos os dados dessa tabela e todos os índices associados a ela também são removidos.

Sintaxe

`DROP TABLE` *`tabela`*

Onde, *`tabela`* é o nome da tabela.

Todos os dados são da tabela são excluídos.

As visões e os sinônimos que apontam para a tabela permanecem, mas ficam inválidos.

As transações pendentes são submetidas a commit.

Somente o dono da tabela ou um usuário com o privilégio `DROP ANY TABLE` pode remover uma tabela.

A instrução `DROP TABLE`, depois de executada, é irreversível. O Oracle não questiona a ação quando você executa o comando `DROP TABLE`. Se você for o dono ou tiver um privilégio de alto nível na tabela, essa tabela será removida imediatamente. Como ocorre com todas as instruções DDL, pós o `DROP TABLE` um `commit` é executado automaticamente logo em seguida.

Exercício 9

9-48

Exercício 9

1. Crie a tabela `DEPT` com base na tabela a seguir. Salve o comando no script `cap_09_01.sql` e o execute. Verifique se a tabela foi criada.

Nome da Coluna	CODIGO	NOME
Tipo de Chave	Primary key	
Tipo de dados	NUMBER	VARCHAR2
Tamanho	7	25

Table	Column	Tipo De Dados	Tamanho	Precisão	Escala	Chave Primária	Anulável
DEPT	CODIGO	Number	-	7	0	1	-
	NOME	Varchar2	25	-	-	-	✓

2. Preencha a tabela `DEPT` com dados da tabela `DEPARTAMENTO`. Inclua somente as colunas necessárias.

3. Crie a tabela `FUNC` com base na tabela a seguir. Salve o comando no script `cap_09_03.sql` e o execute. Verifique se a tabela foi criada.

Nome da Coluna	CODIGO	NOME	SOBRENOME	COD_DEPT
Tipo de dados	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Tamanho	7	25	25	7

Table	Column	Tipo De Dados	Tamanho	Precisão	Escala	Chave Primária	Anulável
EMP	CODIGO	Number	-	7	0	-	✓
	NOME	Varchar2	25	-	-	-	✓
	SOBRENOME	Varchar2	25	-	-	-	✓
	COD_DEPT	Number	-	7	0	-	✓

Exercício 9 (continuação)

4. Crie a tabela FUNCIONARIO2 com base na estrutura da tabela FUNCIONARIO. Inclua somente as colunas COD_FUNCIONARIO, NOME, SOBRENOME, SALARIO e COD_DEPARTAMENTO. Nomeie as colunas da nova tabela como CODIGO, NOME, SOBRENOME, SALARIO e COD_DEPT, respectivamente.
5. Modifique a tabela FUNC para que aceite sobrenomes mais longos de funcionários. Confirme a modificação.

Table	Column	Tipo De Dados	Tamanho	Precisão	Escala	Chave Primária	Anulável
EMP	CODIGO	Number	-	7	0	-	✓
	NOME	Varchar2	25	-	-	-	✓
	SOBRENOME	Varchar2	50	-	-	-	✓
	COD_DEPT	Number	-	7	0	-	✓

06. Elimine a coluna NOME da tabela FUNCIONARIO2 . Confirme a modificação verificando a descrição da tabela.

Table	Column	Tipo De Dados	Tamanho	Precisão	Escala	Chave Primária	Anulável
FUNCIONARIO2	CODIGO	Number	-	6	0	-	✓
	SOBRENOME	Varchar2	25	-	-	-	-
	SALARIO	Number	-	8	2	-	✓
	COD_DEPT	Number	-	4	0	-	✓

07. Na tabela FUNCIONARIO2 , marque a coluna COD_DEPT como UNUSED. Confirme a modificação verificando a descrição da tabela.

Table	Column	Tipo De Dados	Tamanho	Precisão	Escala	Chave Primária	Anulável
FUNCIONARIO2	CODIGO	Number	-	6	0	-	✓
	SOBRENOME	Varchar2	25	-	-	-	-
	SALARIO	Number	-	8	2	-	✓

08. Elimine todas as colunas UNUSED da tabela FUNCIONARIO2 .

09. Adicione uma constraint PRIMARY KEY em nível de tabela para a tabela FUNC na coluna CODIGO. A constraint deve ser nomeada durante a criação. Nomeie-a como emp_cod_pk.

Exercício 9 (continuação)

10. Adicione uma chave estrangeira na tabela FUNC referenciando a tabela `DEPT` que garante que o funcionário não foi designado para um departamento inexistente. Nomeie a constraint como `func_dept_cod_fk`.
11. Modifique a tabela `FUNC`. Adicione uma coluna `COMISSAO` do tipo de dados `NUMBER`, precisão 2, escala 2. Adicione uma constraint à coluna `COMISSAO` que garanta um valor de comissão maior do que zero.
12. Exclua as tabelas `FUNC` e `DEPT`;