

# 10

## Visões, Índices, Sequências e Sinônimos

## Objetivos deste Capítulo

- Ao concluir este capítulo, você poderá:
  - Criar e recuperar dados de visões
  - Criar, manter e usar sequências
  - Criar e manter índices
  - Criar sinônimos públicos e privados

10-2

### Objetivos deste Capítulo

Este capítulo apresenta os objetos de banco de dados: visão, sequência, sinônimo e índice.

Você conhecerá os princípios básicos de como criar e usar visões, sequências e índices.

# Visão (View)

COD_FUNCIONARIO	NOME	SOBRENOME	EMAIL	TELEFONE	DATA_ADMISSAO	COD_CARGO	SA
100	Roberto	Carlos	RCARLOS	051.123.4567	17/06/97	AD_PRES	240
101	Nair	Martins	NMARTINS	051.123.4568	21/09/99	AD_VP	170
102	Leonardo	da Silva	LSILVA	051.123.4569	13/01/03	AD_VP	170
103	Alexandre	Honorato	AHONORATO	081.423.4567	03/01/00	IT_PROG	900
104	Pedro	Osterno	POSTERNO	081.423.4568	21/05/01	IT_PROG	600
107	Pilar	Reata	PRADATA	081.423.5567	07/02/09	IT_PROG	420
124					11/08	ES_GER	580
141					01/05	ES_AUX	350
142					11/07	ES_AUX	310
143					03/08	ES_AUX	260
144					07/08	ES_AUX	250
149					03/01/00	VE_GER	105
174	Paula	Schultz			11/05/06	VE_REP	110
176	Pamela	Sue			24/03/98	VE_REP	860
178				429263	24/05/99	VE_REP	700
200	Elias	Voorhees		123.4444	17/09/97	AD_ASST	440
201	William	Thacker	WTHACKER	+1.515.123.5555	17/02/06	MK_GER	130
202	Cosmo	Kramer	CKRAMER	+1.603.123.6666	17/08/07	MK_ANA	600
205	Sheila	Almeida	SALMEIDA	051.123.8080	07/06/04	CT_GER	120
206	Roberto	Nascimento	RNASCIMENTO	051.123.8181	07/06/04	CTPUB_GER	830

20 linhas retornadas em 0,03 segundos

10-3

## Visão

É o objeto de banco de dados usado para apresentar combinações ou subconjuntos lógicos de dados.

Uma visão é uma tabela lógica baseada em uma tabela ou em outra visão. Uma visão em si não contém dados, mas é semelhante a uma janela por meio da qual é possível exibir ou alterar dados de tabelas. As tabelas nas quais uma visão é baseada são denominadas *tabelas-base*. A visão é armazenada como um comando `SELECT`.

Quando você acessa dados usando uma visão, o Oracle executa as seguintes operações:

- Recupera a definição da visão armazenada no banco de dados.
- Verifica os privilégios de acesso da tabela-base da visão.
- Converte a consulta da visão em uma operação equivalente na(s) tabela(s) base subjacente(s). Em outras palavras, os dados são recuperados das tabelas-base ou ocorre uma atualização nessas tabelas.

## Vantagens das Visões

- Restringir o acesso aos dados
- Facilitar consultas complexas
- Permitir independência de dados
- Apresentar exibições diferentes dos mesmos dados

10-4

### Vantagens das Visões

As visões restringem o acesso aos dados, pois podem exibir apenas algumas colunas selecionadas da tabela.

As visões permitem que os usuários façam consultas simples para recuperar os resultados de consultas complicadas. Por exemplo, as visões permitem aos usuários consultar informações de várias tabelas mesmo sem saber criar um comando `SELECT` com `join`.

As visões permitem a independência de dados a usuários e programas aplicativos, pois mesmo que uma tabela-base seja alterada, se a estrutura de retorno da visão permanecer igual, as aplicações ou usuários que acessam ao dados através da visão não perceberão mudanças.

As visões permitem o acesso de alguns usuários aos dados de acordo com critérios específicos.

# Visões Simples e Complexas

Recurso	Visões Simples	Visões Complexas
Número de tabelas	Uma	Uma ou mais
Contêm funções	Não	Sim
Contêm agrupamento de dados	Não	Sim
Permitem a execução de operações DML na visão	Sim	Nem sempre

10-5

## Visões Simples e Complexas

Existem duas classificações para visões: simples e complexas. A diferença básica está relacionada às operações DML (`INSERT`, `UPDATE` e `DELETE`).

- Uma visão simples é aquela que:
  - É derivada de dados de uma única tabela
  - Não contém funções ou agrupamento de dados
  - Permite a execução de operações DML na visão
- Uma visão complexa é aquela que:
  - É derivada de dados de várias tabelas
  - Contém funções ou agrupamento de dados
  - Nem sempre permite a execução de operações DML na visão

# Criação de uma Visão

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW visão
  [(apelido[, apelido]...)]
  AS subconsulta
  [WITH CHECK OPTION [CONSTRAINT constraint]]
  [WITH READ ONLY [CONSTRAINT constraint]];
```

- A subconsulta pode conter uma sintaxe SELECT complexa.

10-6

## Criação de uma Visão

Você pode criar uma visão usando uma subconsulta no comando CREATE VIEW.

Na sintaxe:

OR REPLACE	recria a visão quando ela já existe
FORCE	cria a visão independentemente da existência das tabelas-base
NOFORCE	só cria a visão quando as tabelas-base existem (padrão)
<i>visão</i>	é o nome da visão
<i>apelido</i>	especifica nomes para as expressões selecionadas pela consulta da visão (O número de apelidos deve corresponder ao número de expressões selecionadas pela visão.)
<i>subconsulta</i>	é um comando SELECT completo (Você pode usar apelidos para as colunas na lista SELECT.)
WITH CHECK OPTION	especifica que apenas as linhas acessíveis à visão podem ser inseridas ou atualizadas
<i>constraint</i>	é o nome designado à constraint CHECK OPTION
WITH READ ONLY	garante que não seja possível executar operações DML na visão

# Criação de uma Visão

- Crie a visão `VW_FUNC80` com detalhes de funcionários do departamento 80:

```
CREATE VIEW   vw_func80
AS SELECT    cod_funcionario, sobrenome, salario
FROM         funcionario
WHERE        cod_departamento = 80;
View criada.
```

- Descreva a estrutura da visão usando o comando SQL\*Plus `DESCRIBE`:

```
DESCRIBE vw_func80
```

10-7

## Criação de uma Visão (continuação)

O exemplo do slide cria uma visão com o código, o sobrenome e o salário de cada funcionário do departamento 80.

Você pode exibir a estrutura da visão usando o comando SQL\*Plus `DESCRIBE`.

Nome	Nulo?	Tipo
-----		
COD_FUNCIONARIO	NOT NULL	NUMBER(6)
SOBRENOME	NOT NULL	VARCHAR2(25)
SALARIO		NUMBER(8,2)

A subconsulta que define uma visão pode conter a sintaxe `SELECT` complexa, inclusive joins, agrupamentos e subconsultas.

Se você não especificar um nome de constraint para a visão criada com `WITH CHECK OPTION`, o sistema designará um nome padrão no formato `SYS_Cn`.

Você pode usar a opção `OR REPLACE` para alterar a definição da visão sem eliminá-la e recriá-la, assim não é necessário conceder novamente com o comando `grant` privilégios de objeto concedidos anteriormente.

## Criação de uma Visão

- Crie uma visão usando apelidos de colunas na subconsulta:

```
CREATE VIEW    vw_sal50
AS SELECT     cod_funcionario CODIGO, sobrenome NOME,
              salario*13 SALARIO_ANUAL
FROM          funcionario
WHERE         cod_departamento = 50;
View criada.
```

- Selecione as colunas dessa visão pelos apelidos definidos

10-8

### Criação de uma Visão (continuação)

Você pode controlar os nomes das colunas incluindo apelidos de colunas na subconsulta.

O exemplo do slide cria uma visão que contém o número (COD\_FUNCIONARIO) com o apelido CODIGO, o nome (SOBRENOME) com o apelido NOME e o salário anual (SALARIO) com o apelido SALARIO\_ANUAL de todos os funcionários do departamento 50.

Como alternativa, você pode usar um apelido depois do comando CREATE e antes da subconsulta SELECT. O número de apelidos listados deve corresponder ao número de expressões selecionadas na subconsulta.

```
CREATE OR REPLACE VIEW    vw_sal50 (CODIGO, NOME, SALARIO_ANUAL)
AS SELECT cod_funcionario, sobrenome, salario*13
FROM      funcionario
WHERE     cod_departamento = 50;
View criada.
```



# Recuperando Dados de uma Visão

```
SELECT *  
FROM vw_sal50;
```

CODIGO	NOME	SALARIO_ANUAL
124	Brunni	75400
141	Lopes	45500
142	Cabral	40300
143	Miranda	33800
144	Chaves	32500

5 linhas retornadas em 0,01 segundos

10-9

## Recuperando Dados de uma Visão

Você pode recuperar dados de uma visão como faria com qualquer tabela. É possível exibir todo o conteúdo da visão ou apenas linhas e colunas específicas.

## Modificando uma Visão

- Modifique a visão `VW_FUNC80` usando a cláusula `CREATE OR REPLACE VIEW`. Adicione um apelido para cada nome de coluna:

```
CREATE OR REPLACE VIEW vw_func80
(codigo, nome, salario, cod_departamento)
AS SELECT  cod_funcionario, nome || ' '
           || sobrenome, salario, cod_departamento
FROM      funcionario
WHERE     cod_departamento = 80;
View criada.
```

- Os apelidos de colunas na cláusula `CREATE OR REPLACE VIEW` são listados na mesma ordem que as colunas na subconsulta.

10-10

### Modificando uma Visão

A opção `OR REPLACE` permite a criação de uma visão mesmo que já exista outra com o mesmo nome, substituindo a versão antiga pela versão de seu respectivo proprietário. Isso significa que é possível alterar a visão sem eliminar, recriar e conceder novamente os privilégios de acesso a visão para os usuários que o possuíam.

Ao criar apelidos de colunas na cláusula `CREATE OR REPLACE VIEW`, lembre-se de que os apelidos são listados na mesma ordem que as colunas na subconsulta.

## Criando uma Visão Complexa

- Crie uma visão complexa que contenha funções de agrupamento para exibir valores de duas tabelas:

- ```
CREATE VIEW vw_info_dept
(nome, menor_salario, maior_salario, media_salario)
AS SELECT      d.nome_departamento, MIN(e.salario),
              MAX(e.salario),AVG(e.salario)
FROM          funcionario e, departamento d
WHERE         e.cod_departamento = d.cod_departamento
GROUP BY      d.nome_departamento;
View criada.
```

## Criando uma Visão Complexa

O exemplo do slide cria uma visão complexa de nomes de departamentos, menores salários, maiores salários e salários médios por departamento. Observe que foram especificados nomes alternativos para a visão. Esse é um requisito quando alguma coluna da visão é derivada de uma função ou de uma expressão.

Você pode exibir a estrutura da visão usando o comando *SQL\*Plus* `DESCRIBE`. Para exibir o conteúdo da visão, execute o comando `SELECT`.

```
SELECT *
FROM    vw_info_dept;
```

[illegible]

7 linhas retornadas em 0.00 segundos

## Executar Operações DML em Visões

- Normalmente você pode executar comandos DML em visões simples.
- Você não poderá remover uma linha se a visão contiver:
  - Funções de agrupamento
  - Uma cláusula `GROUP BY`
  - A palavra-chave `DISTINCT`
  - A palavra-chave da pseudocoluna `ROWNUM`

10-12

### Executar Operações DML em Visões

Você poderá executar operações DML em dados por meio de uma visão se essas operações seguirem certas regras.

É possível remover uma linha de uma visão, a menos que ela contenha:

- Funções de agrupamento
- Uma cláusula `GROUP BY`
- A palavra-chave `DISTINCT`
- A palavra-chave da pseudocoluna `ROWNUM`

## Executar Operações DML em Visões

- Você não poderá modificar dados de uma visão se ela contiver:
  - Funções de agrupamento
  - Uma cláusula `GROUP BY`
  - A palavra-chave `DISTINCT`
  - A palavra-chave da pseudocoluna `ROWNUM`
  - Colunas definidas por expressões

10-13

### Executar Operações DML em Visões (continuação)

Você poderá modificar dados por meio de uma visão, a menos que ela contenha uma das condições mencionadas no slide anterior ou inclua colunas definidas por expressões (por exemplo, `SALARIO * 13`).

## Executar Operações DML em Visões

- Você não poderá adicionar dados por meio de uma visão se ela contiver:
  - Funções de agrupamento
  - Uma cláusula `GROUP BY`
  - A palavra-chave `DISTINCT`
  - A palavra-chave da pseudocoluna `ROWNUM`
  - Colunas definidas por expressões
  - Colunas `NOT NULL` nas tabelas-base que não estejam selecionadas pela visão

10-14

### Executar Operações DML em Visões (continuação)

Você poderá adicionar dados por meio de uma visão, a menos que ela contenha um dos itens listados no slide. Não será possível adicionar dados a uma visão se ela contiver colunas `NOT NULL` sem valores padrão na tabela-base. Todos os valores necessários devem estar na visão. Lembre-se de que você está adicionando valores diretamente à tabela subjacente *por meio* da visão.

## Cláusula WITH CHECK OPTION

- Você pode garantir que as operações DML executadas na visão se restrinjam ao domínio de dados abrangido pela visão usando a cláusula WITH CHECK OPTION:

```
CREATE OR REPLACE VIEW vw_func20
AS SELECT      *
   FROM        funcionario
   WHERE       cod_departamento = 20
   WITH CHECK OPTION CONSTRAINT vw_func20_ck;
View criada.
```

- Uma tentativa de alterar o número do departamento para um valor diferente de 20 não terá êxito porque violará a constraint WITH CHECK OPTION.

10-15

### Cláusula WITH CHECK OPTION

A cláusula WITH CHECK OPTION especifica que as instruções INSERT e UPDATE executadas por meio da visão não podem criar linhas que a visão não pode selecionar. Assim, ela permite a imposição de restrições de integridade e verificações de validação nos dados que estão sendo inseridos ou atualizados. Se houver uma tentativa de execução de operações DML em linhas não selecionadas pela visão, será exibido um erro com o nome da constraint, caso ele tenha sido especificado.

```
UPDATE vw_func20
SET cod_departamento = 10
WHERE cod_funcionario = 201;
```

ERRO na linha 1:

```
ORA-01402: violação da cláusula where da view WITH CHECK
OPTION
```

Nenhuma linha será atualizada porque, se for necessário alterar o número do departamento para 10, a visão não poderá mais ver esse funcionário. Portanto, com a cláusula WITH CHECK OPTION, a visão só poderá ver os funcionários do departamento 20 e não permitirá que o número de departamento desses funcionários seja alterado.

## Negando Operações DML

- Para garantir que não ocorram operações DML, adicione a opção `WITH READ ONLY` à definição da visão.
- Qualquer tentativa de executar uma operação DML nas linhas da visão resultará em erro do Oracle.

```
CREATE OR REPLACE VIEW vw_func10
(codigo, sobrenome, cargo)
AS SELECT      cod_funcionario, sobrenome, cod_cargo
FROM          funcionario
WHERE         cod_departamento = 10
WITH READ ONLY ;
View criada.
```

10-16

### Negando Operações DML

Para garantir que não ocorram operações DML em uma visão, crie essa visão com a opção `WITH READ ONLY`. O exemplo do slide cria a visão `VW_FUNC10` e impede a execução de operações DML nessa visão.

Qualquer tentativa de remover uma linha de uma visão criada com a opção `WITH READ ONLY` resultará em um erro:

```
DELETE FROM vw_func10
WHERE codigo = 200;
DELETE FROM vw_func10
*
```

ERRO na linha 1:

ORA-01752: não é possível a deleção a partir da view sem ter exatamente uma tabela preservada pela chave

Qualquer tentativa de inserir ou modificar uma linha usando a visão com uma constraint somente leitura resultará em um erro do servidor Oracle:

ORA-01733: coluna virtual não permitida aqui



## Removendo uma Visão

- Quando você remover uma visão do banco de dados você não perde dados, pois uma visão é uma consulta baseada em tabelas-base do banco de dados.

```
DROP VIEW visão;
```

```
DROP VIEW vw_func80;  
View eliminada.
```

10-17

### Removendo uma Visão

Use o comando `DROP VIEW` para remover uma visão. O comando remove a definição da visão do banco de dados. A eliminação de uma visão não tem efeito sobre as tabelas nas quais a visão foi baseada. As visões ou outras aplicações baseadas em visões deletadas tornam-se inválidas. Apenas o proprietário ou um usuário com o privilégio `DROP ANY VIEW` pode remover uma visão.

Na sintaxe:

*visão*                      é o nome da visão

# Sequência

- Pode gerar números únicos automaticamente
- É um objeto compartilhável
- Pode ser usada para criar um valor de chave primária
- Substitui a necessidade de codificar na aplicação um gerador de números sequenciais únicos
- Pode ter seus valores gerados armazenados no cache de memória do banco de dados, aumentando assim a velocidade de acesso aos valores

10-18

## Sequência

Uma sequência é um objeto de banco de dados criado por um usuário que pode ser compartilhado por vários usuários para gerar números inteiros.

Você pode definir uma sequência para gerar valores únicos ou para reciclar e reutilizar os mesmos números.

As sequências são muito usadas para criar um valor de chave primária, que precisa ser único para cada linha de uma tabela. A sequência é gerada e incrementada (ou decrementada) por uma rotina interna do Oracle. Esse objeto pode poupar tempo reduzindo o tamanho do código da aplicação necessário para criar uma rotina de geração de sequência.

Os números de sequência são armazenados e gerados independentemente de tabelas. Portanto, é possível usar a mesma sequência para várias tabelas.

# Comando CREATE SEQUENCE

- Sintaxe:

```
CREATE SEQUENCE sequência
  [INCREMENT BY n]
  [START WITH n]
  [{MAXVALUE n | NOMAXVALUE}]
  [{MINVALUE n | NOMINVALUE}]
  [{CYCLE | NOCYCLE}]
  [{CACHE n | NOCACHE}];
```

10-19

## Comando CREATE SEQUENCE

Na sintaxe:

*sequência*  
INCREMENT BY *n*

START WITH *n*

MAXVALUE *n*

NOMAXVALUE

MINVALUE *n*

NOMINVALUE

é o nome do objeto gerador de números  
especifica o intervalo entre os números  
da sequência, em que *n* é um inteiro (Se  
esta cláusula for omitida, a sequência  
será incrementada em 1.)  
especifica o primeiro número da  
sequência a ser gerado (Se esta cláusula  
for omitida, a sequência começará com 1.)  
especifica o valor máximo que a  
sequência pode gerar  
especifica o valor máximo  $10^{27}$  para  
uma sequência em ordem crescente e -1  
para uma sequência em ordem  
decrescente (Esta é a opção padrão.)  
especifica o valor mínimo da sequência  
especifica o valor mínimo 1 para uma  
sequência em ordem crescente e -  
( $10^{26}$ ) para uma sequência em ordem  
decrescente (Esta é a opção padrão.)

## Comando CREATE SEQUENCE

- Crie uma sequência denominada SEQ\_COD\_DEPT a ser usada para a chave primária da tabela DEPARTAMENTO.
- Não use a opção CYCLE.

```
CREATE SEQUENCE seq_cod_dept  
    INCREMENT BY 10  
    START WITH 120  
    MAXVALUE 9999  
    NOCACHE  
    NOCYCLE;  
Sequência criada.
```

10-20

### Comando CREATE SEQUENCE (continuação)

CYCLE | NOCYCLE

especifica se a sequência continuará a gerar valores depois de atingir seu valor máximo ou mínimo (NOCYCLE é a opção padrão.)

CACHE *n* | NOCACHE

especifica quantos valores o Oracle pré-aloca e mantém na memória (Por padrão, o Oracle armazena 20 valores no cache.)

O exemplo do slide cria uma sequência denominada SEQ\_COD\_DEPT a ser usada para a coluna COD\_DEPARTAMENTO da tabela DEPARTAMENTO. A sequência começa em 120, não permite armazenamento em cache e não é cíclica.

Não use a opção CYCLE se a sequência for usada para gerar valores de chave primária, a menos que você tenha um mecanismo confiável que expurgue as linhas antigas mais rápido que os ciclos da sequência.

Uma sequência não é vinculada a uma tabela. Em geral, nomeie a sequência de acordo com o uso pretendido. No entanto, é possível usar a sequência em qualquer local, independentemente de seu nome.

## Pseudocolunas NEXTVAL e CURRVAL

- NEXTVAL retorna o próximo valor disponível da sequência. Quando referenciada, mesmo que por usuários diferentes, ela retorna um valor único.
- CURRVAL obtém o valor atual da sequência.
- É necessário executar NEXTVAL para a sequência antes que CURRVAL contenha um valor.

10-21

### Pseudocolunas NEXTVAL e CURRVAL

Depois que você cria uma sequência, ela gera números sequenciais para uso nas suas tabelas. Faça referência aos valores da sequência usando as pseudocolunas NEXTVAL e CURRVAL.

A pseudocoluna NEXTVAL é usada para extrair números sucessivos de uma sequência especificada. Qualifique NEXTVAL com o nome da sequência. Quando você fizer referência a *sequência*.NEXTVAL, um novo número da sequência será gerado e o número atual da sequência será inserido em CURRVAL.

A pseudocoluna CURRVAL é usada para fazer referência a um número da sequência que o usuário atual acabou de gerar. É preciso usar NEXTVAL para gerar um número de sequência na sessão do usuário atual antes que seja possível fazer referência a CURRVAL. Qualifique CURRVAL com o nome da sequência. Quando você fizer referência a *sequência*.CURRVAL, o último valor retornado para o processo desse usuário será exibido.

## Pseudocolunas `NEXTVAL` e `CURRVAL` (continuação)

### Regras de Uso de `NEXTVAL` e `CURRVAL`

Você pode usar `NEXTVAL` e `CURRVAL` nos seguintes contextos:

- Lista `SELECT` de um comando `SELECT` que não faz parte de uma subconsulta
- Lista `SELECT` de uma subconsulta em um comando `INSERT`
- Cláusula `VALUES` de um comando `INSERT`
- Cláusula `SET` de um comando `UPDATE`

Você não pode usar `NEXTVAL` e `CURRVAL` nos seguintes contextos:

- Lista `SELECT` de uma visão
- Comando `SELECT` com a palavra-chave `DISTINCT`
- Comando `SELECT` com as cláusulas `GROUP BY`, `HAVING` ou `ORDER BY`
- Subconsulta em um comando `SELECT`, `DELETE` ou `UPDATE`
- Expressão `DEFAULT` em um comando `CREATE TABLE` ou `ALTER TABLE`

# Usando uma Sequência

- Insira um novo departamento chamado "Suporte" na localidade de código 2500:

```
INSERT INTO departamento(cod_departamento,  
                           nome_departamento, cod_localidade)  
VALUES      (seq_cod_dept.NEXTVAL,  
             'Suporte', 2500);  
1 linha criada.
```

- Exiba o valor atual da sequência SEQ\_COD\_DEPT:

```
SELECT  seq_cod_dept.CURRVAL  
FROM    dual;
```

10-23

## Usando uma Sequência

O exemplo do slide insere um novo departamento na tabela DEPARTAMENTO. Ele usa a sequência SEQ\_COD\_DEPT para gerar um novo número de departamento como mostrado a seguir.

Você pode exibir o valor atual da sequência:

```
SELECT seq_cod_dept.CURRVAL  
FROM   dual;  
CURRVAL  
-----  
120
```

Agora, suponha que você queira admitir funcionários para a equipe do novo departamento. O comando INSERT a ser executado para todos os novos funcionários pode incluir o seguinte código:

```
INSERT INTO funcionario (cod_funcionario,  
                          cod_departamento, ...)  
VALUES (seq_funcionario.NEXTVAL, seq_cod_dept.CURRVAL,  
        ...);
```

Esse exemplo supõe que uma sequência chamada SEQ\_FUNCIONARIO já tenha sido criada para gerar novos números de funcionários.

# Guardando Valores de Sequência em Cache

- O armazenamento de valores da sequência no cache da memória permite um acesso mais rápido a esses valores.
- Podem haver falhas na geração de números consecutivos (buracos), essas falhas ocorrem quando:
  - É efetuado um rollback
  - Ocorre uma falha do sistema
  - Uma sequência é usada em outra tabela

10-24

## Guardando Valores de Sequência no Cache

É possível armazenar sequências no cache da memória para permitir o acesso mais rápido aos valores da sequência. O cache é preenchido na primeira vez em que você faz referência à sequência. Cada solicitação do próximo valor da sequência é recuperada da sequência armazenada no cache. Depois que o último valor da sequência é usado, a próxima solicitação dessa sequência armazena outro cache de sequências na memória.

### Falhas na Geração de Números Consecutivos

Embora os geradores de sequência gerem números consecutivos sem falhas ou buracos, essa geração de números ocorre independente de que se faça um commit ou rollback após a execução do comando. Portanto, se você executar rollback de um comando que obtenha um número da sequência, esse número será perdido.

Outro evento que pode causar buracos na geração consecutiva é uma falha do sistema. Se a sequência armazenar valores no cache da memória, esses valores serão perdidos caso ocorra uma falha do sistema.

Como as sequências não estão diretamente vinculadas a tabelas, é possível usar a mesma sequência para várias tabelas. Se você fizer isso, cada tabela poderá conter buracos de sequenciamento.



# Modificando uma Sequência

- Altere o valor de incremento, o valor máximo, o valor mínimo, a opção de ciclo ou a opção de cache:

```
ALTER SEQUENCE seq_cod_dept  
            INCREMENT BY 20  
            MAXVALUE 999999  
            NOCACHE  
            NOCYCLE;  
Sequência alterada.
```

10-25

## Modificando uma Sequência

Se o limite `MAXVALUE` definido para a sequência for atingido, não serão alocados outros valores da sequência e você receberá um erro indicando que essa sequência excede `MAXVALUE`. Para continuar a usar a sequência, você poderá modificá-la usando o comando `ALTER SEQUENCE`.

### Sintaxe

```
ALTER SEQUENCE sequência  
    [INCREMENT BY n]  
    [{MAXVALUE n | NOMAXVALUE}]  
    [{MINVALUE n | NOMINVALUE}]  
    [{CYCLE | NOCYCLE}]  
    [{CACHE n | NOCACHE}];
```

Na sintaxe, *sequência* é o nome do objeto gerador de sequência.

## Diretrizes para Modificar uma Sequência

- Você precisa ser o proprietário da sequência ou ter o privilégio `ALTER` na sequência.
- Somente os futuros números da sequência são afetados.
- É necessário eliminar e recriar a sequência para reiniciá-la em um número diferente.
- É executada uma validação.
- Para remover uma sequência, use o comando `DROP SEQUENCE`:

```
DROP SEQUENCE seq_cod_dept;  
Sequência eliminada.
```

10-26

### Diretrizes para Modificar uma Sequência

Você precisa ser o proprietário da sequência ou ter o privilégio `ALTER` na sequência para modificá-la. Você precisa ser o proprietário da sequência ou ter o privilégio `DROP ANY SEQUENCE` para removê-la.

Somente os futuros números da sequência serão afetados pelo comando `ALTER SEQUENCE`.

Não é possível alterar a opção `START WITH` usando `ALTER SEQUENCE`. É necessário eliminar e recriar a sequência para reiniciá-la em um número diferente.

É executada uma validação. Por exemplo, não é possível impor um novo `MAXVALUE` menor que o número atual da sequência.

```
ALTER SEQUENCE seq_cod_dept  
    INCREMENT BY 20  
    MAXVALUE 90  
    NOCACHE  
    NOCYCLE;
```

```
ALTER SEQUENCE seq_cod_dept  
*
```

ERRO na linha 1:

ORA-04009: MAXVALUE não pode ser criado para ser menor que o valor atual

# Índice

- É um objeto de esquema do banco de dados
- É usado pelo Oracle para acelerar a recuperação de linhas
- Pode reduzir a leitura/gravação de disco por fornecer um caminho de acesso mais rápido aos dados da tabela
- É um objeto independente da tabela que indexa
- É usado e mantido automaticamente pelo Oracle

10-27

## Índice

Um índice é um objeto de esquema do banco de dados Oracle que pode acelerar a recuperação de linhas. Para isso o índice usa um ponteiro para os dados na tabela. É possível criar índices de forma explícita ou automática. Se você não tiver um índice criado para uma coluna de uma tabela, para executar uma consulta, o Oracle terá que fazer uma varredura integral da tabela (Full Table Scan).

Um índice permite acesso direto e rápido a linhas de uma tabela. Seu objetivo é reduzir a necessidade de leitura/gravação de disco usando um caminho indexado para localizar os dados com rapidez. O Oracle usa e mantém automaticamente o índice. Depois que um índice é criado, não é exigida qualquer atividade direta do usuário.

Do ponto de vista lógico e físico, os índices são independentes da tabela que indexam. Isso significa que é possível criá-los ou eliminá-los a qualquer momento sem causar efeitos nas tabelas-base ou em outros índices.

Quando você elimina uma tabela, os índices correspondentes também são eliminados.

# Criação de Índices

- Automaticamente: Um índice de chave exclusiva é criado automaticamente quando você define uma constraint de chave primária ou `UNIQUE` em uma definição de tabela.
- Manualmente: Os usuários podem criar índices não exclusivos em colunas para acelerar o acesso às linhas. Também conhecidos como índice de performance.

10-28

## Criação de Índices

É possível criar dois tipos de índice.

**Índice de chave exclusiva:** O Oracle cria automaticamente este índice quando você define uma constraint de chave primária ou `UNIQUE` para a coluna de uma tabela. O nome do índice é o mesmo fornecido à constraint.

**Índice não exclusivo:** É um índice que um usuário pode criar. Por exemplo, você pode criar um índice de uma coluna que é chave estrangeira para uma join em uma consulta a fim de aumentar a velocidade de recuperação.

Você pode criar um índice de chave exclusiva de forma manual, mas é recomendável criar uma constraint `UNIQUE`, que gera implicitamente um índice de chave exclusiva.

# Criação de Índices

- Crie um índice em uma ou mais colunas:

```
CREATE [UNIQUE] INDEX índice  
ON tabela (coluna[, coluna]...);
```

- Aumente a velocidade de acesso da consulta à  
coluna SOBRENOME da tabela FUNCIONARIO:

```
CREATE INDEX funcionario_sobrenome_idx  
ON          funcionario(sobrenome);  
Índice criado.
```

10-29

## Criação de Índices (continuação)

Crie um índice em uma ou mais colunas executando o comando `CREATE INDEX`.

Na sintaxe:

|               |                                             |
|---------------|---------------------------------------------|
| <i>índice</i> | é o nome do índice                          |
| <i>tabala</i> | é o nome da tabela                          |
| <i>coluna</i> | é o nome da coluna da tabela a ser indexada |

## CREATE INDEX com o Comando

```
CREATE TABLE NOVA_FUNC
(cod_funcionario NUMBER(6)
        PRIMARY KEY USING INDEX
        (CREATE INDEX func_cod_idx ON
        NOVA_FUNC(cod_funcionario)),
nome      VARCHAR2(20),
sobrenome VARCHAR2(25));
Tabela criada.
```

10-30

### CREATE INDEX com o Comando CREATE TABLE

No exemplo do slide, a cláusula `CREATE INDEX` é usada com o comando `CREATE TABLE` para criar um índice de chave primária explicitamente. Você pode nomear os índices no momento da criação da chave primária para diferenciá-lo do nome da constraint `PRIMARY KEY`.

Se você não der um nome a chave primária ou ao índice da chave primária, o Oracle irá atribuir um nome automaticamente como `SYS_CXXXX`, onde `XXXX` é um número criado automaticamente.

## **CREATE INDEX com a Instrução CREATE TABLE (continuação)**

Também é possível usar um índice existente para a coluna de chave primária; por exemplo, quando você estiver esperando uma carga de uma grande quantidade de dados e quiser acelerar a operação. Você pode desativar as constraints enquanto executa a carga e, em seguida, ativá-las. Nesse caso, a existência de um índice exclusivo na chave primária fará com que os dados sejam verificados durante a carga. Sendo assim, você pode primeiro criar um índice não exclusivo na coluna designada como `PRIMARY KEY` e depois criar a coluna `PRIMARY KEY` e especificar que ela deve usar o índice existente. Os exemplos abaixo ilustram esse processo:

### **Etapa 1: Crie a tabela**

```
CREATE TABLE NOVA_FUNC2
(  cod_funcionario NUMBER(6),
   nome             VARCHAR2(20),
   sobrenome        VARCHAR2(25)
);
```

### **Etapa 2: Crie o índice**

```
CREATE INDEX func_cod_idx2 ON
           nova_func2(cod_funcionario);
```

### **Etapa 3: Crie a Chave Primária**

```
ALTER TABLE nova_func2 ADD PRIMARY KEY (cod_funcionario) USING
INDEX func_cod_idx2;
```

# Índices Baseados em Função

- Um índice baseado em função utiliza expressões.
- A expressão do índice é criada a partir de colunas de tabela, constantes, funções SQL e funções definidas pelo usuário.

```
CREATE INDEX upper_dept_nome_idx  
ON departamento(UPPER(nome_departamento));
```

Índice criado.

```
SELECT *  
FROM departamento  
WHERE UPPER(nome_departamento) = 'VENDAS';
```

10-32

## Índices Baseados em Função

Os índices baseados em função com as palavras-chave `UPPER(coluna)` ou `LOWER(coluna)` permitem pesquisas sem distinção entre maiúsculas e minúsculas. Por exemplo, o índice:

```
CREATE INDEX upper_sobrenome_idx ON funcionario  
(UPPER(sobrenome));
```

facilita o processamento de consultas como:

```
SELECT * FROM funcionario  
WHERE UPPER(sobrenome) = 'CARLOS';
```

O Oracle usa o índice apenas quando essa função específica é usada em uma consulta. Por exemplo, talvez o comando abaixo use o índice, mas, sem a cláusula `WHERE`, o servidor Oracle poderá executar uma varredura integral de tabela:

```
SELECT *  
FROM funcionario  
WHERE UPPER(sobrenome) IS NOT NULL  
ORDER BY UPPER(sobrenome);
```

O parâmetro de inicialização `QUERY_REWRITE_ENABLED` deve ser definido como `TRUE` para que seja usado um índice baseado em função.



# Diretrizes para a Criação de Índices

| Crie um índice quando:              |                                                                                                                                        |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> | Uma coluna conter uma grande faixa de valores                                                                                          |
| <input checked="" type="checkbox"/> | Uma coluna conter um grande número de valores nulos                                                                                    |
| <input checked="" type="checkbox"/> | Uma ou mais colunas forem usadas em conjunto com frequência em uma cláusula <code>WHERE</code> ou em uma condição de <code>join</code> |
| <input checked="" type="checkbox"/> | A tabela for grande e for esperado que a maioria das consultas recupere menos de 2% a 4% das linhas da tabela                          |
| Não crie um índice quando:          |                                                                                                                                        |
| <input type="checkbox"/>            | As colunas não forem usadas com frequência como uma condição na consulta                                                               |
| <input type="checkbox"/>            | A tabela for pequena ou for esperado que a maioria das consultas recupere mais de 2% a 4% das linhas da tabela                         |
| <input type="checkbox"/>            | A tabela for atualizada com frequência                                                                                                 |
| <input type="checkbox"/>            | As colunas indexadas forem referenciadas como parte de uma expressão                                                                   |

10-33

## Diretrizes para a Criação de Índices

Um número maior de índices em uma tabela não resulta em consultas mais rápidas. Para cada operação DML submetida a `commit` em uma tabela com índices, é necessário atualizar esses índices. Quanto mais índices estiverem associados a uma tabela, maior será o esforço do Oracle para atualizar todos os índices após uma operação DML.

Portanto, você só deverá criar índices se:

- A coluna conter uma grande faixa de valores
- A coluna conter um grande número de valores nulos
- Uma ou mais colunas forem usadas em conjunto com frequência em uma cláusula `WHERE` ou em uma condição de `join`
- A tabela for grande e for esperado que a maioria das consultas recupere menos de 2% a 4% das linhas

Lembre-se de que, para impor a exclusividade, você deverá definir uma constraint `UNIQUE` na definição da tabela. Depois, um índice exclusivo será criado automaticamente.

# Removendo um Índice

- Para remover um índice do banco de dados, use o comando `DROP INDEX`:

```
DROP INDEX índice;
```

- Remova o índice `FUNCIONARIO_SOBRENOME_IDX` do banco de dados:

```
DROP INDEX funcionario_sobrenome_idx;  
Índice eliminado.
```

- Para eliminar um índice, você precisa ser o proprietário dele ou ter o privilégio `DROP ANY INDEX`.

10-34

## Removendo um Índice

Não é possível modificar índices. Para alterar um índice, elimine-o e, depois, recrie-o.

Remova um índice do banco de dados executando o comando `DROP INDEX`. Para eliminar um índice, você precisa ser o proprietário dele ou ter o privilégio `DROP ANY INDEX`.

Na sintaxe, *índice* é o nome do índice.

Se você eliminar uma tabela, os índices e as constraints serão eliminados automaticamente, mas as visões e as sequências permanecerão.

# Sinônimos

- Simplifique o acesso a objetos criando um sinônimo (outro nome para um objeto).
- Com sinônimos, você pode:
  - Criar uma referência mais fácil a uma tabela pertencente a outro usuário
  - Reduzir nomes longos de objetos

```
CREATE [PUBLIC] SYNONYM sinônimo  
FOR      objeto;
```

10-35

## Sinônimos

Os sinônimos são objetos de banco de dados que permitem chamar uma tabela usando outro nome. É possível criar sinônimos para atribuir um nome alternativo a uma tabela.

Para fazer referência a uma tabela pertencente a outro usuário, é preciso inserir o nome do esquema dessa tabela seguido de um ponto como prefixo do nome da tabela. A criação de um sinônimo elimina a necessidade de qualificar o nome do objeto com o esquema e fornece um nome alternativo para uma tabela, uma visão, uma sequência, um procedimento outros objetos. Esse método pode ser útil especialmente com nomes longos de objetos, como visões.

Na sintaxe:

|                              |                                                     |
|------------------------------|-----------------------------------------------------|
| <code>PUBLIC</code>          | cria um sinônimo acessível a todos os usuários      |
| <code><i>sinônimo</i></code> | é o nome do sinônimo a ser criado                   |
| <code><i>Objeto</i></code>   | identifica o objeto para o qual o sinônimo é criado |

O objeto não pode estar contido em um Pacote.

O nome de um sinônimo privado deve ser diferente do de outros objetos pertencentes ao mesmo usuário.

# Criando e Removendo Sinônimos

- Crie um nome abreviado para a visão VW\_INFO\_DEPT:

```
CREATE SYNONYM d_soma  
FOR vw_info_dept;  
Sinônimo criado.
```

- Elimine um sinônimo:

```
DROP SYNONYM d_soma;  
Sinônimo eliminado.
```

10-36

## Criando um Sinônimo

O exemplo do slide cria um sinônimo relativo à visão VW\_INFO\_DEPT para agilizar a referência.

O administrador do banco de dados pode criar um sinônimo público acessível a todos os usuários. Este exemplo cria um sinônimo público denominado DEPT para a tabela DEPARTAMENTO da usuária Alice:

```
CREATE PUBLIC SYNONYM dept  
FOR alice.departamento;  
Sinônimo criado.
```

## Removendo um Sinônimo

Para remover um sinônimo, use o comando DROP SYNONYM. Somente o administrador do banco de dados pode eliminar um sinônimo público.

```
DROP PUBLIC SYNONYM dept;  
Sinônimo eliminado.
```

# Exercício 10

10-37

## Exercício 10

1. Crie uma visão denominada `VW_FUNCIONARIO` com base nos códigos, sobrenomes e código do departamento da tabela `FUNCIONARIO`. Atribua o nome `FUNCIONARIO` a coluna com o sobrenome do funcionário.
2. Verifique se a visão funciona. Exiba o conteúdo da visão `VW_FUNCIONARIO`.

| COD_FUNCIONARIO | FUNCIONARIO | COD_DEPARTAMENTO |
|-----------------|-------------|------------------|
| 100             | Carlos      | 90               |
| 101             | Martins     | 90               |
| 102             | da Silva    | 90               |
| 103             | Honorato    | 60               |
| 104             | Osterno     | 60               |
| 107             | Barata      | 60               |
| 108             | Bruni       | 50               |

3. Usando a visão `VW_FUNCIONARIO`, crie uma consulta para exibir todos os nomes de funcionário e código de departamento.

| FUNCIONARIO | COD_DEPARTAMENTO |
|-------------|------------------|
| Carlos      | 90               |
| Martins     | 90               |
| da Silva    | 90               |
| Honorato    | 60               |
| Osterno     | 60               |
| Barata      | 60               |
| Bruni       | 50               |

## Exercício 10

- O departamento 50 precisa de acesso aos dados de seus funcionários. Crie uma visão denominada `DEPT50` que contenha os códigos e os sobrenomes, bem como os códigos de departamento de todos os funcionários do departamento 50. Foi solicitada a atribuição dos nomes `COD_FUNC`, `FUNCIONARIO` e `COD_DEPT` às colunas da visão. Para fins de segurança, não permita que um funcionário seja alterado para outro departamento por meio da visão.
- Exiba a estrutura e o conteúdo da visão `DEPT50`.

| Table               | Column                   | Tipo De Dados | Tamanho | Precisão | Escala | Chave Primária | Anulável |
|---------------------|--------------------------|---------------|---------|----------|--------|----------------|----------|
| <code>DEPT50</code> | <code>COD_FUNC</code>    | Number        | -       | 6        | 0      | -              | -        |
|                     | <code>FUNCIONARIO</code> | Varchar2      | 25      | -        | -      | -              | -        |
|                     | <code>COD_DEPT</code>    | Number        | -       | 4        | 0      | -              | ✓        |

| <code>COD_FUNC</code> | <code>FUNCIONARIO</code> | <code>COD_DEPT</code> |
|-----------------------|--------------------------|-----------------------|
| 124                   | Brunni                   | 50                    |
| 141                   | Lopes                    | 50                    |
| 142                   | Cabral                   | 50                    |
| 143                   | Miranda                  | 50                    |
| 144                   | Chaves                   | 50                    |

5 linhas retornadas em 0,01 segundos

- Teste a visão. Tente alterar Miranda para o departamento 80.
- Você precisa de uma sequência que possa ser usada com a coluna de chave primária da tabela `DEPARTAMENTO`. A sequência deve começar com o valor 200 e ter o valor máximo 1.000. Incremente a sequência em 10. Nomeie-a como `SEQ_DEPARTAMENTO_COD`.
- Para testar a sequência, crie um script para inserir duas linhas na tabela `DEPARTAMENTO`, mas não faz commit. Nomeie o script como `cap_10_08.sql`. Certifique-se de utilizar a sequência criada no exercício anterior para a coluna `cod_departamento`. Adicione dois departamentos: Treinamento e Qualidade na localidade 1700 e com o gerente 100. Confirme as adições e observe os códigos gerados.
- Execute um rollback das inserções do exercício anterior e execute novamente o script `cap_10_08.sql`. Consulte os dados da tabela `DEPARTAMENTO` e observe os códigos gerados novamente. Faça novamente rollback.
- Crie um índice não exclusivo na coluna `NOME` da tabela `DEPARTAMENTO`.
- Crie um sinônimo para a tabela `FUNCIONARIO`. Nomeie o sinônimo como `FUNC`.