
FIAP

Visões – Sequencias – Sinônimos – Índices (Parte 1)

Objetivos

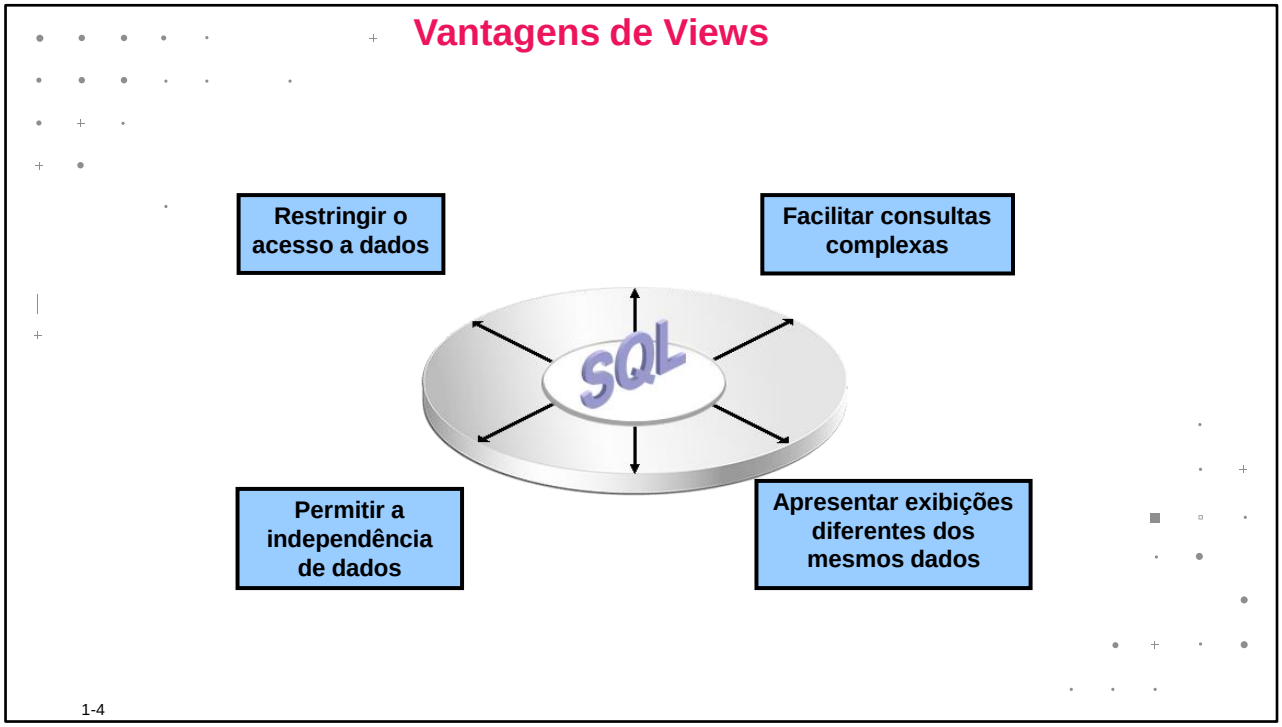
Ao concluir esta lição, você será capaz de:

- **Criar views simples e complexas**
- **Recuperar dados de views**

1-3

Objetivos

Esta lição apresenta os objetos view, sequência, sinônimo e índice. Você conhecerá os princípios básicos de como criar e usar views.



Vantagens de Views

As views restringem o acesso aos dados, pois podem exibir colunas selecionadas da tabela.

As views permitem fazer consultas simples para recuperar os resultados de consultas complicadas. Por exemplo, as views permitem aos usuários consultar informações de várias tabelas mesmo sem saber criar uma instrução de join.

As views permitem a independência de dados a usuários ad hoc e programas aplicativos. É possível usar uma view para recuperar dados de várias tabelas.

As views permitem o acesso de grupos de usuários a dados de acordo com critérios específicos.

Para obter mais informações, consulte "`CREATE VIEW`" no manual *Oracle SQL Reference*.

Views Simples e Complexas

Recurso	Views Simples	Views Complexas
Número de tabelas	Uma	Uma ou mais
Contêm functions	Não	Sim
Contêm grupos de	Não	Sim
Permitem a execução de operações DML	Sim	Nem sempre

1-5

Views Simples e Complexas

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

Uma view simples é aquela que:

- É derivada de dados de uma única tabela
- Não contém functions ou grupos de dados
- Permite a execução de operações DML

Uma view complexa é aquela que:

- É derivada de dados de várias tabelas
- Contém functions ou grupos de dados
- Nem sempre permite a execução de operações DML

Criando uma View

- **Incorpore uma subconsulta à instrução `CREATE VIEW`:**

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
 AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- **A subconsulta pode conter uma sintaxe `SELECT` complexa.**

1-6

Criando uma View

Você pode criar uma view incorporando uma subconsulta à instrução `CREATE VIEW`.

Na sintaxe:

`OR REPLACE`	recria a view quando ela já existe
`FORCE`	cria a view independentemente da existência das tabelas-base
`NOFORCE`	só cria a view quando as tabelas-base existem (Este é o default.)
`view`	é o nome da view
`alias`	especifica nomes para as expressões selecionadas pela consulta da view (O número de apelidos deve corresponder ao número de expressões selecionadas pela view.)
`subquery`	é uma instrução SELECT completa (Você pode usar apelidos para as colunas na lista SELECT.)
`WITH CHECK OPTION`	especifica que apenas as linhas acessíveis à view podem

	ser inseridas ou atualizadas
constraint	é o nome designado à constraint CHECK OPTION
WITH READ ONLY	garante que não seja possível executar operações DML na view

Criando uma View

- **Crie a view `EMPVU80` com detalhes de funcionários do departamento 80:**

```
CREATE VIEW  empvu80
 AS SELECT  employee_id, last_name, salary
    FROM    employees
    WHERE   department_id = 80;
View created.
```

- **Descreva a estrutura da view usando o comando `DESCRIBE`:**

```
DESCRIBE empvu80
```

1-7

Criando uma View (continuação)

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

Você pode exibir a estrutura da view usando o comando `DESCRIBE`.

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY		NUMBER(8,2)

Se você não especificar um nome de constraint para a view criada com `WITH CHECK OPTION`, o sistema designará um nome default no formato `SYS_C`*n*.

Você pode usar a opção `OR REPLACE` para alterar a definição da view sem eliminá-la e recriá-la, ou sem conceder novamente com grant privilégios de objeto concedidos anteriormente.

Criando uma View

- **Crie uma view usando apelidos de colunas na subconsulta:**

```
CREATE VIEW  salvu50
 AS SELECT  employee_id ID_NUMBER, last_name NAME,
            salary*12 ANN_SALARY
    FROM    employees
    WHERE   department_id = 50;
View created.
```

- **Selecione as colunas dessa view pelos apelidos definidos:**

1-8

Criando uma View (continuação)

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

O exemplo do slide cria uma view que contém o número (EMPLOYEE_ID) com o apelido ID_NUMBER, o nome (LAST_NAME) com o apelido NAME e o salário anual (SALARY) com o apelido ANN_SALARY de todos os funcionários do departamento 50.

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

```
CREATE OR REPLACE VIEW   salvu50 (ID_NUMBER, NAME,
ANN_SALARY)
  AS SELECT  employee_id, last_name, salary*12
     FROM employees
     WHERE       department_id = 50;
View created.
```

Recuperando Dados de uma View

```
SELECT *
FROM   salvu50;
```

ID_NUMBER	NAME	ANN_SALARY
124	Mourgos	69600
141	Rajs	42000
142	Davies	37200
143	Matos	31200
144	Vargas	30000

1-9

Recuperando Dados de uma View

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

Modificando uma View

- **Modifique a view `EMPVU80` usando a cláusula `CREATE OR REPLACE VIEW`. Adicione um apelido para cada nome de coluna:**

```
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT  employee_id, first_name || ' '
           || last_name, salary, department_id
   FROM    employees
   WHERE   department_id = 80;
View created.
```

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

1-10

Modificando uma View

A opção `OR REPLACE` permite a criação de uma view 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 view sem eliminar, recriar e conceder novamente privilégios de objeto com grant.

Observação: Ao designar 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 View Complexa

Crie uma view complexa que contenha functions de grupo para exibir valores de duas tabelas:

```
CREATE VIEW   dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT      d.department_name, MIN(e.salary),
               MAX(e.salary),AVG(e.salary)
   FROM        employees e, departments d
   WHERE       e.department_id = d.department_id
   GROUP BY    d.department_name;
View created.
```

1-11

Criando uma View Complexa

O exemplo do slide cria uma view complexa de nomes de departamentos, salários mínimos, salários máximos e salários médios por departamento.
Observe que foram especificados nomes alternativos para a view. Esse é um requisito quando alguma coluna da view é derivada de uma function ou de uma expressão.
Você pode exibir a estrutura da view usando o comando `DESCRIBE`. Para exibir o conteúdo da view, execute a instrução `SELECT`.

```
SELECT *
FROM   dept_sum_vu;
```

NAME	MINSAL	MAXSAL	AVGSAL
Accounting	8300	12000	10150
Administration	4400	4400	4400
Executive	17000	24000	19333.3333
IT	4200	9000	6400
Marketing	6000	13000	9500
Sales	8600	11000	10033.3333
Shipping	2500	5800	3500

7 rows selected.

Regras para Executar Operações DML em uma View

- **Em geral, é possível executar operações DML em views simples.**
- **Você não poderá remover (DELETE) uma linha se a view contiver:**
  - **Functions de grupo**
  - **Uma cláusula `GROUP BY`**
  - **A palavra-chave `DISTINCT`**
  - **A palavra-chave da pseudocoluna `ROWNUM`**

1-12

Executando Operações DML em uma View

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

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

Functions de grupo

Uma cláusula `GROUP BY`

A palavra-chave `DISTINCT`

A palavra-chave da pseudocoluna `ROWNUM`

Regras para Executar Operações DML em uma View

Você não poderá modificar (UPDATE) dados de uma view se ela contiver:

- **Functions de grupo**
- **Uma cláusula `GROUP BY`**
- **A palavra-chave `DISTINCT`**
- **A palavra-chave da pseudocoluna `ROWNUM`**
- **Colunas definidas por expressões**

1-13

Executando Operações DML em uma View (continuação)

Você poderá modificar dados por meio de uma view, a menos que ela contenha uma das condições mencionadas no slide anterior ou inclua colunas definidas por expressões (por exemplo, `SALARY * 12`).

Regras para Executar Operações DML em uma View

Você não poderá adicionar (INSERT) dados por meio de uma view se ela contiver:

- **Functions de grupo**
- **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 view**

1-14

Executando Operações DML em uma View (continuação)

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

Para obter mais informações, consulte "`CREATE VIEW`" no manual *Oracle SQL Reference*.

Usando a Cláusula WITH CHECK OPTION

- Você pode garantir que as operações DML executadas na view se restrinjam ao domínio dessa view usando a cláusula **WITH CHECK OPTION:**

```
CREATE OR REPLACE VIEW empvu20
AS SELECT    *
   FROM      employees
   WHERE     department_id = 20
   WITH CHECK OPTION CONSTRAINT empvu20_ck ;
View created.
```

- Qualquer tentativa de alterar o número do departamento relativo a uma linha da view não terá êxito porque violará a constraint **WITH CHECK OPTION.**

1-15

Usando a Cláusula `WITH CHECK OPTION`

Você pode executar verificações de integridade referencial por meio de views. Pode também impor constraints no nível do banco de dados. É possível usar a view para proteger a integridade dos dados, mas o uso é muito limitado.
A cláusula `WITH CHECK OPTION` especifica que as instruções `INSERT` e `UPDATE` executadas por meio da view não podem criar linhas que a view não pode selecionar. Assim, ela permite a imposição de constraints 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 view, será exibido um erro com o nome da constraint, caso ele tenha sido especificado.

```
UPDATE empvu20
SET department_id = 10
WHERE employee_id = 201;
```

causa:

```
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause
violation
```

Observação: Nenhuma linha será atualizada porque, se for necessário alterar o número do departamento para 10, a view não poderá mais ver esse funcionário. Portanto, com a cláusula `WITH CHECK OPTION`, a view 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 view.**
- **Qualquer tentativa de executar uma operação DML nas linhas da view resultará em erro do servidor Oracle.**

Negando Operações DML

Para garantir que não ocorram operações DML em uma view, crie essa view com a opção `WITH READ ONLY`. O exemplo do próximo slide modifica a view `EMPVU10` para impedir a execução de operações DML nessa view.

Negando Operações DML

```
CREATE OR REPLACE VIEW empvu10
    (employee_number, employee_name, job_title)
AS SELECT     employee_id, last_name, job_id
   FROM      employees
   WHERE     department_id = 10
   WITH READ ONLY ;
View created.
```

1-17

Negando Operações DML (continuação)

Qualquer tentativa de remover uma linha de uma view com uma constraint somente leitura resultará em um erro:

```
DELETE FROM empvu10
WHERE employee_number = 200;
DELETE FROM empvu10
       *
ERROR at line 1:
ORA-01752: cannot delete from view without exactly one
key-preserved table
```

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

```
01733: virtual column not allowed here.
```

Removendo uma View

Você pode remover uma view sem perder dados, pois uma view é baseada em tabelas subjacentes do banco de dados.

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
View dropped.
```

1-18

Removendo uma View

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

Na sintaxe:

`view` é o nome da view

Exercício: Visão Geral da Parte 1

Este exercício aborda os seguintes tópicos:

- **Criando uma view simples**
- **Criando uma view complexa**
- **Criando uma view com uma constraint de verificação**
- **Tentando modificar dados da view**
- **Removendo views**

1-19

Exercício : Visão Geral da Parte 1

A Parte 1 do exercício desta lição contém várias atividades que permitem criar, utilizar e remover views.

Faça as questões de 1 a 6 no final desta lição.

Sumário

Nesta lição, você aprendeu a:

- **Criar, usar e remover views**

Exercício

Parte 1

1. A equipe do departamento de recursos humanos deseja ocultar alguns dados da tabela `EMPLOYEES`. Ela deseja uma view denominada `EMPLOYEES_VU` com base nos números e nomes de funcionário, bem como nos números de departamento da tabela `EMPLOYEES`. Também deseja atribuir `EMPLOYEE` como o cabeçalho do nome do funcionário.
2. Verifique se a view funciona. Exiba o conteúdo da view `EMPLOYEES_VU`.

	EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
...			
104	203	Mavris	40
105	204	Baer	70
106	205	Higgins	110
107	206	Gietz	110

3. Usando a view `EMPLOYEES_VU`, crie uma consulta para o departamento de recursos humanos a fim de exibir todos os nomes de funcionário e números de departamento.

	EMPLOYEE	DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
...		
104	Mavris	40
105	Baer	70
106	Higgins	110
107	Gietz	110

Exercício

4. O departamento 50 precisa de acesso aos dados de seus funcionários. Crie uma view denominada `DEPT50` que contenha os números e os sobrenomes, bem como os números de departamento de todos os funcionários do departamento 50. Foi solicitada a atribuição dos labels `EMPNO`, `EMPLOYEE` e `DEPTNO` às colunas da view. Para fins de segurança, não permita que um funcionário seja redesignado a outro departamento por meio da view.
5. Exiba a estrutura e o conteúdo da view `DEPT50`.

```
Nome     Nulo?    Tipo
-------- -------- ------------
EMPNO    NOT NULL NUMBER(6)
EMPLOYEE NOT NULL VARCHAR2(25)
DEPTNO             NUMBER(4)
```

	EMPNO	EMPLOYEE	DEPTNO
1	120	Weiss	50
2	121	Fripp	50
3	122	Kaufling	50
4	123	Vollman	50
...			
42	196	Walsh	50
43	197	Feeney	50
44	198	OConnell	50
45	199	Grant	50

6. Teste a view. Tente redesignar Matos ao departamento 80 (transfira Matos para o departamento 80).

OBRIGADO

profalexandre.barcelos@fiap.com.br

https://www.linkedin.com/in/alexandrebarcelos

FIAP


FIAP