

[◀ VOLTAR](#)

Controle de transações e criação de relatórios

Conhecer os conceitos dos comandos DML. Para isso, é necessário controlar as transações realizadas por eles, transferi-las do database buffer (memória) para a base de dados (estrutura) e criar relatórios a partir de dados existente no banco de dados.

NESTE TÓPICO

- Controle de transações
- Estado do dado antes do COMMIT ou ROLLBACK
- Estado do dado após um comando COMMIT Marcar tópico
- Estado do dado após um comando ROLLBACK
- Criação de relatórios



Controle de transações

Até a presente aula os comandos apresentados pertencem à subdivisão DML – Data Manipulation Language – Linguagem de Manipulação de Dados, que são: Insert, Update e Delete. Eles provocam alterações no volume de uma base de dados. Esses volumes podem ter novas inclusões realizadas pelo comando Insert, atualizações diversas realizadas pelo comando Update e eliminações pelo comando Delete. Também são chamados de transações e é necessário administrá-las por meio de comandos.

Uma transação nada mais é que uma manipulação dos dados em uma tabela ou em um conjunto de tabelas. Para melhorar essas transações, elas são realizadas em uma memória chamada de database buffer e depois precisam ser gravadas na base de dados (fisicamente). Para que isso ocorra, existem comandos que controlam as transações.

São eles:

COMMIT => Confirma as manutenções (grava).

ROLLBACK => Desfaz as últimas alterações (até ultimo COMMIT).

SAVEPOINT => Coloca uma marca para possível ROLLBACK até aquele ponto.



ROLLBACK TO => Desfaz até o ponto especificado.

Vamos exemplificar utilizando a tabela criada na aula passada, a tabela funcionário.

Após a criação da tabela foram inseridas algumas linhas. O uso do comando que insere dados na tabela já é a realização de transações. Após a inserção dos registros é interessante validar essas transações, caso ocorra algum problema, ou seja, realizada uma transação não apropriada, os dados podem voltar ao estado anterior sem muito ou quase nenhum trabalho.

Vamos ao exemplo: em um momento da aula anterior a tabela funcionário foi criada (os comandos DDL, que trabalham a estrutura não fazem parte das transações), após sua criação, foi alimentada com dados. A partir desse momento começa a ser necessário o uso dos comandos que controlam as transações. Nessa mesma aula realizamos algumas atualizações, imagine que essas alterações não estavam autorizadas e alguém as executou. Se quiser voltar ao estado anterior, seria necessário refazer as inclusões novamente. É nesse momento que são usadas as instruções que controlam as transações.

Uso do commit:

Visualização dos dados atuais da tabela funcionário:



```
SQL> select * from funcionario;
MATRICULA NOME                SALARIO  DT_ADMIS  C
-----
1 Claudia          5000    10/10/10  A
2 Silvio           5600    07/12/00  B
3 Vanessa          4598,3  23/08/11  A
4 Ricardo           5350    01/04/00  A
4 linhas selecionadas.
SQL>
```

Caso seja realizada a transação equivocada que elimina todos os dados da tabela funcionário (delete from funcionario), esses dados não poderão mais ser recuperados, pois estão em memória e não foram gravados fisicamente. Com o uso do controle de transação, os dados podem ser recuperados de forma rápida. Seu uso é fácil, basta digitar o comando commit após a instrução de transação (insert, update, delete).

Uso do rollback:

Após a validação, qualquer movimentação imprevista pode ser desfeita com esse comando até encontrar a validação (commit). Voltando ao exemplo, se após o conjunto de comandos de inserção insert for realizado o commit, os dados podem ser deletados e, após a deleção, eles são recuperados, usando o comando rollback.

Praticamente:



1. COMMIT
2. Insert....
3. Insert ...
4. Delete..
5. ROLLBACK
6. Update?
7. Update
8. COMMIT

É importante destacar que todo dado modificado durante a transação só será efetivado quando aquela transação for confirmada (COMMIT).

Estado do dado antes do COMMIT ou ROLLBACK

Os comandos de manipulação de dados inicialmente afetam o database buffer. O usuário corrente (que efetuou a mudança) pode visualizar suas modificações por meio do comando SELECT. Outros usuários não enxergam essas modificações. As linhas afetadas nos comandos permanecem "bloqueadas". Nenhum outro usuário pode modificar essas linhas.

Estado do dado após um comando COMMIT

As mudanças são retiradas do database buffer e escritas em arquivos. Todos os usuários podem visualizar as modificações. Os "bloqueios" nas linhas afetadas são liberados.

Estado do dado após um comando ROLLBACK

As modificações são descartadas. Os arquivos continuam como estavam. Os "bloqueios" nas linhas afetadas pelas modificações são liberados.

Savepoint e Rollback to:

Esses são complementos de uso para controle das transações. O savepoint recebe um nome (por exemplo savepoint A) e o rollback pode ser realizado até um ou outro ponto de restauração (por exemplo rollback to savepoint A). Praticamente:

1. COMMIT
2. Insert....
3. Insert ...
4. Savepoint A
5. Delete..
6. ROLLBACK to savepoint A
7. Update?
8. Update
9. COMMIT

Nessa situação, só o comando delete será desfeito.

Criação de relatórios

Relatórios são pesquisas realizadas na base de dados e exibidas em tela, em nosso caso no SQL PLUS.



O comando de criar relatório é o Select.

A sintaxe:

```
1. Select nome_coluna1,.....,nome_colunaN from nome_tabela;
```

Exemplificando:

Criar um relatório que mostre o nome do funcionário e seu salário.

```
1. Select nome, salario from funcionario;
```

É muito importante conhecer a estrutura da tabela antes de realizar qualquer instrução.

Caso exista a necessidade de um relatório para visualizar todas as colunas, não é necessário digitar uma por uma, basta utilizar um coringa: o asterisco:

```
1. Select * from funcionario;
```

Esse comando retorna todas as colunas existentes na tabela funcionário.

Uma variação desse comando é inibição de valores repetidos na instrução de seleção. Caso se queira exibir dados de uma coluna sem a repetição dos valores, podemos utilizar uma função que não permite a exibição de ocorrências iguais, essa função é a `distinct(nome_coluna)`.

Exemplificando:

No caso do relatório que mostra todas as ocorrências na tabela funcionário, percebe-se que existem valores repetidos na coluna comissão. Caso prefira, existem apenas os valores que não se repetem. A instrução funcionaria assim:

```
1. Select distinct (comissão) from funcionario;
```



Agora só aparecerão os dados diferentes que estão cadastrados na tabela funcionário. Essa função não pode ser aplicada com outra coluna, por exemplo:

```
1. Select nome, distinct(comissão) from funcionario;
```





```
SQL> select nome, distinct <comissao> from funcionario;
select nome, distinct <comissao> from funcionario
      *
ERRO na linha 1:
ORA-00936: expressão não encontrada

SQL> _
```

Como o nome do funcionário não possui repetição, não servirá de nada a função na coluna comissão.

Todo relatório pode ser classificado em sua exibição, a classificação ocorre em duas situações:

Ascendente, do menor valor para o maior, comando ASC (default, não precisa ser digitado).

Descendente, do maior valor para o menor, comando DESC.

A classificação ocorre com o comando order by nome_coluna opção.

Exemplo: Mostrar o nome dos funcionários ordenados pelo nome em ordem crescente:

1. Select nome from funcionario order by nome asc;



```
SQL> select nome from funcionario
2 order by nome asc;

NOME
-----
Claudia
Ricardo
Silvio
Vanessa

SQL>
```

Caso necessite do processo contrário, é só trocar a opção ao final da instrução;

1. Select nome from funcionario order by nome desc;



```
SQL> select nome from funcionario
2 order by nome desc;

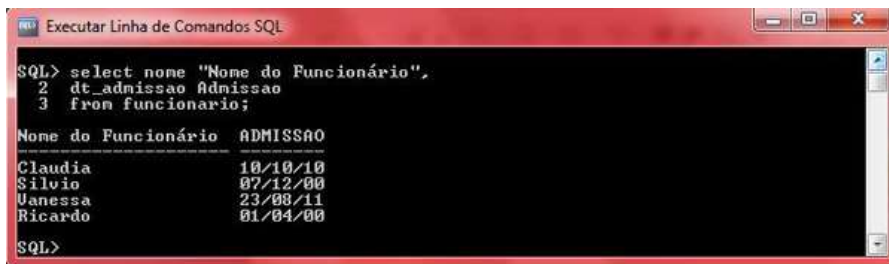
NOME
-----
Vanessa
Silvio
Ricardo
Claudia

SQL>
```

Os nomes das colunas podem ser trocados, ao ser criado um relatório. A esse processo dá-se o nome de apelido na coluna. Para executar esse processo, basta, após a digitação do nome da coluna que está na tabela, digitar seu apelido, se o apelido for com uma palavra, não há necessidade do uso de aspas.

Se houver mais de uma palavra, entretanto, o uso torna-se obrigatório.
Segue um exemplo:

1. `Select nome "Nome do Funcionário", dt_admissao Admissao from funcionario;;`



```
SQL> select nome "Nome do Funcionário",
2 dt_admissao Admissao
3 from funcionario;

Nome do Funcionário  ADMISSAO
-----
Claudia             10/10/10
Silvio              07/12/00
Vanessa             23/08/11
Ricardo             01/04/00
SQL>
```

Referências

BEIGHLEY, Lynn. *Use a Cabeça SQL*. Rio de Janeiro: Alta Books, 2008.

FANDERUFF, Damaris. *Dominando o Oracle 9i: Modelagem e Desenvolvimento*, São Paulo: Makron, 2003.

GRAVES, Mark. *Projeto de banco de dados com XML*. São Paulo: Pearson, 2003.

MORELLI, Eduardo Terra. *Oracle 9i Fundamental: SQL, PL/SQL e Administração*, São Paulo, Editora Érica, 2002.

PRICE, Jason. *Oracle Database 11g SQL*. (tradução: João Eduardo Nóbrega Tortello). Porto Alegre: Bookman, 2009.

SILVA, Robson. *Oracle Database 10g Express Edition*. São Paulo: Editora Érica, 2007.



Avalie este tópico



ANTERIOR

<

Manipulação de dados nas tabelas - inserindo dados - parte II

Biblioteca

(<https://www.uninove.br/conhec-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/>)

Portal Uninove

(<http://www.uninove.br>)

Mapa do Site

Índice

≡

Índice

Ajuda?

PRÓXIMO

(<https://ava.uninove.br/cursos/>)

Criação de relatórios utilizando filtros, operadores relacionais e operadores lógicos

© Todos os direitos reservados