IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 1

MODELO RELACIONAL

Tema

MODELO RELACIONAL

Palavras-chave

Objetivos

O aluno deverá ser capaz de:

- Conceituar Relação
- Identificar os componentes do modelo
- Conceituar integridade referencial

Estrutura de Conteúdo

UnidadeI – Lógico Relacional

1.1 Introdução ao Modelo Relacional

Proposto originalmente por E. F. Codd durante seu trabalho no Laboratório de Pesquisas da IBM em San Jose na segunda metade da década de 60, o Modelo Relacional se baseia em conceitos matemáticos para proporcionar uma representação confiável para Bancos de Dados. Desde sua implantação no primeiro SGBDR (Sistema de Gerenciamento de Bancos de Dados Relacionais) da IBM denominado System R, várias melhorias têm sido incluídas ao modelo, o que o torna ainda atual e o mais utilizado no mercado.

1.2 Estrutura relacional – domínios, relações, variáveis e valores

- **Relação:** também chamadas tabelas contêm informações sobre entidades ou relacionamentos existentes no domínio da aplicação utilizada como alvo para a modelagem. Informalmente uma relação pode ser considerada como uma tabela de valores, onde cada linha desta tabela representa uma coleção de valores de dados interrelacionados. Estes conjuntos de valores podem estar representando uma instância de uma entidade ou relacionamento da aplicação. Os nomes

fornecidos às tabelas e às suas colunas podem auxiliar na compreensão do significado dos valores armazenados

- **Atributo**: nome dado no Modelo Relacional a cada coluna da relação
- **Tupla**: nome dado no Modelo Relacional a cada linha da relação
- **Domínio**: consiste de um grupo de valores atômicos a partir dos quais um ou maisatributos (ou colunas) retiram seus valores reais.
- **Esquema de uma Relação:** consiste de um conjunto de atributos que descrevem as características dos elementos a ser modelados.
- **Instância de uma Relação:** consiste no conjunto de valores que cada atributo, definido no esquema, assume em um determinado instante, formando o conjunto de tuplas.
- **Chave:** conjunto de atributos de uma relação e que pode ser utilizado para a realização de qualquer operação que envolva atributos e valores de atributos.
- **Super Chave** é o maior conjunto de atributos para manipulação e/ou identificação univocamente de uma tupla em uma relação. Uma relação pode ter várias chaves para identificação unívoca de suas tuplas, onde cada uma é denominada de **Chave Candidata**.
- **Chave Primária:** chave candidata escolhida durante a fase de projeto lógico para ser suportada pelo SGBD e assim, é mantido automaticamente a restrição de unicidade. Quando uma chave primária for constituída por mais de um atributo da relação esta é denominada de **Chave Primária Composta**, caso contrário e denominada **Chave Primária Simples**.

- **Chave Estrangeira**: é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela. A chave estrangeira é o mecanismo que permite a implementação de relacionamentos em um banco de dados relacional

1.3 Regras de integridade relacional

Um dos objetivos primordiais de um SGBD é a integridade de dados. Dizer que os dados de um banco de dados estão íntegros significa dizer que eles refletem corretamente a realidade representada pelo banco de dados e que são consistentes entre si. Para tentar garantir a integridade de um banco de dados os SGBD oferecem o mecanismo de restrições de integridade. Uma restrição de integridade é uma regra de consistência de dados que é garantida pelo próprio SGBD. No caso da abordagem relacional, costuma-se classificar as restrições de integridade nas seguintes categorias:

- **Restrição de Domínio:** os valores de atributos devem ser coerentes com os domínios correspondentes. Cada Atributos de uma relação é definido com base em um domínio de valores (exemplos: Idade, seu domínio são os números inteiros positivos; Salários, seu domínio são os números reais positivos).
- **Restrição de Chave Primária (unicidade)**: cada valor de chave primária deve ser único dentro de seu escopo (a relação a que pertence).

- **Restrição de Entidade**: o valor de uma chave primária nunca deve ser nulo. O valor nulo não permite a identificação de uma tupla.
- **Restrição de Referência**: toda referência a uma tupla através de chave estrangeira deve ser verificada, ou seja, toda tupla referenciada deve previamente existir no Banco de Dados a menos que esta restrição seja explicitamente desprezada pelo usuário do Banco de Dados.
- **Restrições Semântica**, que podem ou não serem especificadas ou garantidas por um BD Relacional. se referem mais especificamente sobre valores ou características que determinados atributos podem assumir no contexto de uma determinada aplicação. As Restrições Semânticas consistem em definir intervalos de valores para os atributos, limites, condições de existência, e outras. Exemplos:- Salário: valores no intervalo de 150,00 a 2400,00 reais; Categoria: Senior se tempo de trabalho >20, Pleno se tempo de trabalho é >10, Júnior caso contrário.

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

1. Modelo Relacional

Apresentar aos alunos os componentes básicos do modelo relacional.

Conceitual Relação.

Explicar o que são atributos, tuplas, campos e valores

Conceituar Chave Primária, Estrangeira e Candidata

2. Integridade Referencial

Explicar a importância da integridade referencial.

Exemplificar o funcionamento da integridade a partir de um modelo de banco de dados

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Este banco pode ser bem simples com duas tabelas , por exemplo empregado e departamento, onde em empregado deverá existir uma chave estrangeira para departamento.

As tabelas podem ser pequenas com um máximo de 10 ou 15 linhas em empregado e 5 em departamento.

Para a prática o deverá ser apresentado aos alunos o programa cliente de banco e mostrado como realizar a conexão.

A seguir os alunos deverão ser orientados à acessar uma tabela do banco de dados e ver seu conteúdo.

O professor deverá mostrar na banco a aplicação dos conceitos apresentados (chave primária, colunas, tuplas, chave estrangeira, etc.)

Para exemplificar a integridade referencial o professor deverá:

1.Tentar excluir um departamento que tenha empregados e mostrar que o banco de dados não permite.

2. Inserir um novo empregado colocando como departamento um departamento que não existe na tabela de departamentos e mostrar que o banco também não permite esta operação

III. Discussão com a turma:

Qual a importância da integridade referencial?

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.
Laboratório com cliente de banco de dados
Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados
Cap 2 livro Sistema de banco de dados do Silberschatz.

Avaliação

Validar o aprendizado com as seguintes questões (feitas oralmente):

· O que é uma relação?

. Conceitue chave primaria

Conceitue chave estrangeira

Qual a diferença entre valor e campo?

.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 10

LINGUAGEM SQL – SUB CONSULTA

Tema

LINGUAGEM SQL – SUB CONSULTA

Palavras-chave

Objetivos

O aluno deverá ser capaz de:

- Escrever Comandos de Select utilizando subconsulta
- Distinguir subconsulta aninhada de correlata

Estrutura de Conteúdo

UnidadeII – Linguagem Sql

2.3.Comando Select

2.3.7 SubConsulta

Subconsulta é uma consulta dentro de um outro comando SQL que pode ser:

CREATE TABLE
CREATE VIEW
SELECT
INSERT
DELETE
UPDATE

Regras para Construção de Subconsultas

- A consulta interna tem que estar entre parênteses e sempre a mais interna é executada primeiro.
- Admite o aninhamento de n consultas internas ou em conjunto com operadores AND e OR

- Retorna uma ou várias linhas ou colunas
- Usualmente é usada na cláusula WHERE dos comandos SELECT, DELETE e UPDATE
- Pode ser usada na cláusula FROM do comando SELECT .
- Pode referenciar colunas da consulta mais externa
- Não pode conter a cláusula ORDER BY

Subconsultas Simples

Uma subconsulta simples é aquela que retorna no máximo uma linha e, neste caso, será tratada por um conjunto específico de operadores, diferentemente daquelas que retornam mais de uma linha.

Os operadores abaixo podem ser usados em condições de comparação com subconsultas simples, pois manipulam o retorno de apenas uma linha de comparação com a consulta principal.

Operador	**Descrição**
=	Igual a
<>	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual

Subconsulta simples em SELECT

Observe no exemplo abaixo, que existe um comando SELECT (consulta inicial) e um outro comando SELECT dentro deste primeiro (interno).

```
SELECT ult_nome
FROM c_empr
WHERE id_Depto = (SELECT id
                  FROM  c_Depto
```

WHERE Nome = 'Financeiro');

Subconsultas Multi-Linhas

Uma subconsulta multi-linhas é aquela que pode retornar à consulta principal mais que uma linha de dados para efeito de substituição. Isto, por conseguinte, implica numa construção ligeiramente diferente a nível de operadores, pois estes têm que ser capazes de manipular um conjunto (lista) com mais de um elemento.

Os operadores abaixo podem ser usados em condições de comparação com subconsultas multi-linhas, pois permitem a manipulação de um conjunto com mais de 1 elemento.

OPERADOR	DESCRIÇÃO
IN	Igual a qualquer elemento de
NOT IN	Diferente que qualquer elemento de
> ALL	maior que todos os elementos da lista
< ALL	menor que todos
<> ALL	diferente de todos (o mesmo que NOT IN)
= ANY	igual a *algum* dos elementos da lista (o mesmo que IN)
> ANY	maior que algum dos elementos da lista
< ANY	menor que algum dos elementos
<> ANY	diferente de algum dos elementos da lista (falso se igual a todos)

Além disso, podem ser usadas combinações com >= e <=, de forma análoga. Note que não é permitido o uso de = ALL.

Subconsultas correlatas

As sub-consultas que foram vistas até agora nos exemplos podem ser avaliadas uma vez só e depois substituídas no corpo da consulta principal. Já uma sub-consulta *correlata* [correlated subquery] depende dos valores da consulta principal onde ela está alinhada, por isso deve ser avaliada uma vez para cada linha do resultado externo.

Por exemplo, desejamos saber quais empregados ganham mais que a média salarial de seu departamento. A consulta abaixo nos dá este resultado.

```
Select id, ult_nome, salario
From c_empr e1
Where salario > (select avg(salario) from c_empr e2
                Where e1.id_depto = e2.id_depto);
```

Essa é uma sub-consulta correlata porque ela faz referência a uma tabela da consulta mais externa. A sub-consulta é avaliada repetidas vezes, uma para cada linha da retornada pela consulta externa.

Testes de existência

Um teste de existência é uma condição que envolve a palavra EXISTS e uma sub-consulta. A condição é verdadeira se a sub-consulta retorna alguma linha e é falsa se ela retorna zero linhas. Por exemplo, para saber quais os departamentos que possuem funcionários cujo cargo é igual a 'Almoxarife' execute o seguinte comando:

```
Select *
From c_depto d
Where exists (select *  from c_empr e
                    Where e.id_depto = d.id and e.cargo = 'Almoxarife');
```

O resultado da sub-consulta não importa, pois está apenas sendo testada a existência de um resultado. Nesse caso, a lista de colunas é sempre um asterisco (*).

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o conceito de subconsulta exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

Destacar as diferença de subconsulta que retornem uma única linha e subconsulta multilinhas.

Diferenciar subconsulta aninhadas de correlatas e explicar as diferenças em seu processamento, destacando os casos que um ou outra devem ser utilizadas

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de comando select utilizando subconsulta.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

· Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 11

LINGUAGEM SQL – OPERADORES E CONJUNHTO

Tema

LINGUAGEM SQL – OPERADORES DE CONJUNTO

Palavras-chave

Objetivos

O aluno deverá ser capaz de:

- Distinguir os operadores de conjunto
- Escrever comandos SQL utilizando operadores de conjunto

Estrutura de Conteúdo

UnidadeII – Linguagem Sql

2.3.8Operadores de Conjunto

Trabalhando com Conjuntos

Muitas vezes necessitamos que nossas consultas incluam, em uma mesma coluna, dados de mais de uma tabela. Para isto utilizamos os operadores de conjuntos UNION, UNION ALL, INTERSECT e MINUS.

Regras para Utilização dos Operadores de Conjuntos

- Os comandos SELECT participantes têm que ter o mesmo número de colunas,
- As colunas correspondentes têm que ser do mesmo tipo de dado,
- Linhas duplicadas são automaticamente descartadas, exceto com UNION ALL,
- Os nomes das colunas resultantes são os da primeira consulta,
- ALIAS de colunas só tem efeito se utilizados na primeira consulta,
- A cláusula ORDER BY só pode ser utilizada ao final do comando,

- Os operadores de conjuntos podem ser utilizados em subconsultas.

O Operador UNION

Resulta na combinação de todas as linhas de duas ou mais tabelas participantes do UNION, eliminando as linhas duplicadas resultantes.

Se quisermos, por exemplo, id dos empregados que trabalham nos departamentos 31, 32, 41 e 42 **ou** que emitiram faturas podemos dar o seguinte comando:

Select id from empregados where id_depto in (31,32,41,42)
Union
Select id_repr_vendas from faturas

O Operador UNION ALL

Resulta na combinação de todas as linhas de duas ou mais tabelas participantes do UNION, mantendo todas as linhas duplicadas.

O Operador INTERSECT

Resulta na interseção entre todas as linhas de duas ou mais tabelas participantes do INTERSECT, ou seja, apenas as linhas comuns entre ela

Se no comando do exemplo anterior desejássemos o id dos empregados que trabalham nos departamento 31,32,41,42 **e** que emitiram faturas o comando seria

Select id from empregados where id_depto in (31,32,41,42)
Intersect
Select id_repr_vendas from faturas

O Operador MINUS (Except)

Resulta nas linhas existentes na primeira tabela, mas que não existem na segunda. As linhas comuns também não são resultantes. Novamente se no exemplo anterior desejássemos saber os que não emitiram fatura o comando seria

Select id from empregados where id_depto in (31,32,41,42)
Minus
Select id_repr_vendas from faturas

Nota no Oracle este operador é denominado Minus, em outros SGBD´s Except

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Os operadores de conjunto devem ser apresentados como mais um recurso na construção de consultas. Poderá ser exemplificado para os alunos comando de select que utilizam operadores lógicos (and, or e not) na clausula where podem ser mapeados para comando com operadores de conjunto, discutindo-se ainda qual das formas é mais interessante nos diversos casos.

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de comando select utilizando operadores de conjunto.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 12

VISÕES

Tema

VISÕES

Palavras-chave

viewa

Objetivos

O aluno deverá ser capaz de:

- Criar e Manipular Visões

Estrutura de Conteúdo

UnidadeII ? Linguagem Sql

Criando Visões

> Uma View funciona de forma semelhante a uma tabela. É utilizada em comandos SELECT, INSERT, UPDATE e DELETE, para recuperação e manipulação de dados (com restrições), porém, não armazena estes dados.

Este objeto tem suas linhas e colunas calculadas dinamicamente através de um SELECT pré-estabelecido, cada vez que o solicitamos. Apenas a sua definição é armazenada no dicionário de dados.

Sintaxe do comando:

> CREATE [OR REPLACE] [FORCE I NOFORCE] VIEW nome_view
> [(alias1 [, alias2] [, ...])]
> AS subquery

Onde:

Cláusula	*Descrição*
OR REPLACE	Se existente, a View é recriada
FORCE	Cria a view mesmo que a tabela referenciada não exista
NOFORCE	Cria a View apenas se existir a tabela. É default
Nome_view	É o nome da View
aliasn	Especifica os nomes que atuarão como pseudocolunas da view, a

	partir de expressões e/ ou colunas oriundas da tabela referenciada.
subquery	É o comando SELECT que originará a View

Exemplo:

CREATE VIEW RESUMO_DEPTO AS
SELECT D.ID, D.NOME, COUNT(*) QTD_ EMP, SUM(SAL) TOT_SAL
FROM C_DEPTO D, C_EMPR E
WHERE D.ID = E.ID_DEPTO
GROUP BY D.ID, D.NOME

- Agora, podemos utilizar RESUMO_DEPTO como se fosse uma tabela.
- A diferença é que ela obtém os dados dinamicamente através da query especificada na própria definição da View.

Recuperando Dados Através da View

SELECT * FROM RESUMO_DEPTO;

Tipos de Views

Existem basicamente dois tipos de Views:

- Simples e
- Complexas.

O tipo simples é composto por apenas um SELECT, utiliza apenas uma tabela, suas colunas são formadas por colunas da tabela original, sem cálculos ou funções.

A View complexa é aquele onde há um join entre tabelas na subquery, conforme visto no exemplo.

Nota:

- Com uma VIEW simples será possível executarmos comandos INSERT, UPDATE e DELETE (além do SELECT).
- A manipulação dos dados através de uma VIEW não desabilita as constraints das tabelas as quais os mesmos se referem.
- Cada coluna definida para VIEWs deve ter um nome de coluna válido.
- Caso seja uma fórmula, deve possuir um alias.

Eliminando uma View

O comando DROP VIEW deleta uma View do Dicionário de Dados. Nenhum efeito ocorrerá sobre as Tabelas referenciadas, bastando para isso ter apenas o privilégio DROP VIEW ou DBA.

Vejamos a sintaxe do comando:

DROP VIEW nome_view;

Onde:

Cláusula	Decrição
nome_view	É o nome da view

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o conceito de Visão e os casos em que ela deve ser utilizada. Deve ser enfatizado que a partir de visões pode ser implemento o nível externo do banco de dados.

Devem ser apresentados os comandos de criação e eliminação de visões exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow. Deverá ser destacado a natureza ?dinâmica? da visão, mostrando que os se alterar os dados nas tabelas das quais ela se origina o conteúdo da visão se modifica.

Especial destaque deve ser dado as diferenças entre visões simples e complexas exemplificando no banco de dados como uma visão simples permite alteração nas linhas da tabela e uma visão complexas não. Pode ser interessante citar que para permitir alteração a partir de visões complexas é necessário realizar programação interna no SGBD utilizando gatilhos instead of.

Quanto as sequences deve ser mostrado ao aluno a sua criação e dar exemplos de quando a sua utilização pode ser interessante

.

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de criação de visões e sequences.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.
Laboratório com cliente de banco de dados
Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados
Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 14

INDICES

Tema

INDICES

Palavras-chave

indices

Objetivos

O aluno deverá ser capaz de:

- Identificar os diversos tipos de indices
- Criar Indices utilizando a linguagem SQL

Estrutura de Conteúdo

UnidadeIII ? Indices

3.1 Tipos de Indices

Índices: estruturas de dados auxiliares cujo único propósito é tornar mais rápido o acesso a registros baseado em certos campos, chamados campos de indexação.

Tipos:
? Índice primário - baseado na chave de ordenação.
? Índice secundário - baseado em qualquer campo não ordenado de um arquivo.

Índices Densos e Esparsos

? Índice denso:
? há uma **entrada no índice para cada valor** de chave que ocorre em um registro de dados
? a entrada aponta para o primeiro registro que contém aquele valor de chave

? Índice esparso:
? há um **entrada no índice apenas para alguns valores** de chave a entrada aponta para o primeiro registro que contém aquele valor de chave para localizar um registro com chave ***k***, procura-se a entrada ***e*** do índice com o maior valor de chave menor ou igual a ***k*** e pesquisa-se o arquivo a partir do registro apontado por ***e***

Índice Cluster e Não Cluster

? Índice de agrupamento (clustering) ? baseado no campo de ordenação não-chave de um arquivo.
? Índice Clusterizado ? as tuplas que tem um valor da chave de pesquisa estão armazenados no mesmo bloco ou em blocos contíguos, haverá, portanto, uma tendência de recuperar uma ou poucas paginas folhas do índice.

? Índice Não Clusterizado ? duas chaves consecutivas, armazenadas em uma mesma página folha do índice, podem apontar para páginas de dados totalmente distintas. Com isto uma varredura por todas as paginas folha do índice tende a realizar um número de acesso a paginas de dados igual ao número de chaves de pesquisa contidas nas folhas do índice.

Índice Multinível

?Índice de índice?.

? Primeiro nível: arquivo ordenado pela chave de indexação, valores distintos, entradas de tamanho fixo.

? Demais níveis: índice primário sobre o índice do nível anterior e assim sucessivamente até que no último nível o índice ocupe apenas um bloco.

Número de acessos a bloco: um a cada nível de índice, mais um ao bloco do arquivo de dados.

3.2 Definição de Índice em SQL.

Um Índice tem o propósito de acelerar o acesso aos dados de tabelas muito extensas, ou que são freqüentemente acessadas via join.

Nos SGBD, ao criarmos uma PRIMARY KEY ou UNIQUE KEY, automaticamente é definido um índice único de acesso àquelas chaves, porém outras colunas que acessamos constantemente poderão ser indexadas e para isso, temos que conhecer os comandos de criação e eliminação de Índices.

Tipos de Índices

- **Único** - Garante a unicidade do valor. O Índice criado na Primary Key ou Unique Key, é único, porém podemos criar restrições de unicidade em outras colunas da tabela.
- **Não_único** - Índices criados apenas com o propósito de acelerar a pesquisa, como em Chaves Estrangeiras (Foreing Key), onde a unicidade não é requerida.
- **Uma Coluna** - Apenas uma coluna será indexada.
- **Colunas Compostas ou Concatenadas** - Até 255 colunas podem ser concatenadas para formar apenas um índice e não necessitam ser adjacentes.
- **Função** ? É criado em função de um cálculo feito em uma ou mais colunas.

Utilizando Índices

Devemos utilizar índices sempre que:

- O meio de acesso tradicional (full table scan) mostrar-se ineficiente, provavelmente pelo tamanho da tabela e um número alto de consultas feitas a ela.
- A coluna tiver limites extensos de valores
- ***A coluna tiver muitos valores NULOS***
- Duas ou mais colunas são freqüentemente acessadas em conjunto numa cláusula WHERE ou condição.
- A tabela for muito grande e na maior parte das queries, é esperada a recuperação de até 4% das linhas.

Não devemos utilizar índices sempre que:

- A tabela for pequena (pode ser armazenada em poucos blocos Oracle).
- As colunas não são freqüentemente usadas em condições.
- A maior parte das queries recupera mais que 4% das linhas da tabela.
- A tabela sofre alta taxa de atualização.

Cuidados com Índices

É importante observar que Índices são objetos atualizados pelo Oracle, em todas as operações de INSERT, UPDATE e DELETE na tabela indexada, portanto, se por um lado aceleram a pesquisa aos dados através do comando SELECT, por outro lado, quanto mais Índice tiver a tabela, maior é o tempo de atualização da mesma.

Criando um Índice

Vejamos a sintaxe do comando abaixo:

CREATE INDEX [schema.]nome_indice ON tabela (coluna1 [, coluna2 [,...]])

Onde:

schema	E o schema onde será gerado o índice. O default é o schema do usuário que está criando o índice.
nome_índice	Nome dado ao objeto índice que será criado
tabela	Nome da tabela que será indexada
coluna n	A(s) coluna(s) que irão compor o índice.

Eliminando um Índice

Índices podem ser eliminados mas nunca alterados. Para alterá-los devemos efetuar a remoção e depois recriá-los.

O comando de eliminação de um índice é simples e exige apenas o privilégio DROP INDEX.

Vejamos a sintaxe do comando abaixo:

DROP INDEX [schema.]nome_índice
Onde:

Cláusula	*Descrição*
schema	E o schema a quem pertence o índice. O default é o schema do

	usuário que está removendo o índice.
nome_índice	Nome do índice que será removido.

Nota:

- Não podemos eliminar os Índices criados automaticamente pelas constraints PRIMARY KEY e UNIQUE KEY. Estes são automaticamente removidos quando eliminamos ou desabilitamos as constraints.

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o conceito de índice e descrever seus diversos tipos. Deve ser enfatizado a importância dos índices como ferramentas para melhorar a recuperação de dados em um banco de dados.

Devem ser apresentados os comandos de criação e eliminação de índices exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

.

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de criação de índices

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap14 Estruturas de Indexação de Arquivo de livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 15

TRANSAÇÃO

Tema

TRANSAÇÃO

Palavras-chave

transação

Objetivos

O aluno deverá ser capaz de:

- Conceituar Transação
- Identificar as propriedades da transação
- Distinguir os tipos de controle de transação

Realizar o controle de transação utilizando comandos de SQL

Estrutura de Conteúdo

UnidadeIV ? Transação

4.1 Conceito de transação

Um conjunto de várias operações em um BD pode ser vista pelo usuário como uma única unidade.

Exemplo:

> A transferência de fundos de uma conta corrente pra uma conta poupança é uma operação única do pontos de vista do cliente, porém, dentro do sistemas de banco de dados, ela envolve várias operações.

É essencial que todo o conjunto de operações seja concluído, ou que, no caso de uma falha, nenhuma delas ocorra. Essas operações, que forma uma única unidade lógica de trabalho são chamadas de transações.

Podemos formalizar transação como: um conjunto de operações sobre o BD que devem ser executados integralmente e sem falhas ou interrupções e que leva o banco de dados de um estado consistente para outro estado consistente.

O SGBD deve realizar o controle das Transações para que sejam executadas com segurança.

4.2 Estado da Transação

Ativo - Uma transação entra em estado ativo imediatamente após o início de sua execução, no qual poderá emitir operações READ (leitura) e WRITE (gravação).

Ex: BEGIN_TRANSACTION /*Inicio da transacao*/

begin /*Inicio*/

select * from c_depto; /*Leitura*/

insert into c_depto_aux (select * from c_depto); /*Gravação*/

Efetivação Parcial - Quando a transação termina, ela passa para o estado de efetivação parcial. Uma vez efetivada, a transação tem sua execução concluída com sucesso, e todas as suas mudanças serão gravadas permanentemente no banco de dados(COMMIT).

Ex: COMMIT_TRANSACTION /*Termino da transacao*/

insert into c_depto_aux (select * from c_depto); /*Gravação*/

commit; /*Termino da transacao*/

Falha - Uma transação poderá entrar em estado de falha se uma das verificações falhar ou se a transação for interrompida durante seu estado ativo. A transação deverá, então, ser revertida para desfazer(ROLLBACK) os efeitos de suas operações WRITE no banco de dados.

Ex: insert into c_depto_aux (select * from c_depto); /*Gravação*/

/*Durante a transação o usuario perdeu a conexao com o banco de dados.*/

/*O controle de transação devera dar o ROLLBACK implícito*/

rollback; /*Termino da transacao*/

Encerrado - Corresponde ao estado da transação quando deixa o sistema. As informações sobre a transação que foram mantidas no sistema em tabelas, enquanto a transação estava em execução, serão removidas quando a transação terminar.

Ex: END_TRANSACTION /* Termino da transacao*/

insert into c_depto_aux (select * from c_depto); /*Gravação*/

commit; /* ou */ rollback;

end; /*Termino*/

4.3 Propriedades da Transação

O SGBD deve realizar o controle das Transações para que sejam executadas com segurança. E assim garantir as propriedades da transação(ACID):

- Atomicidade: Ou todas as ações da transação acontecem, ou nenhuma delas acontece;
- Consistência: Se a transação é consistente e o BD começa consistente, ele termina consistente;
- Isolação: A execução de uma transação é isolada da execução de outras transações;
- Durabilidade: Se uma transação é concluída com sucesso (através de uma operação *commit* bem sucedida), então seus efeitos são persistentes(duráveis).

4.4 Execução Concorrente de Transações

Quando diversas transações são executadas de forma concorrente em um banco de dados, a propriedade do isolamento pode não ser preservada. É necessário que o sistema controle a interação entre transações concorrentes, esse controle é alcançado por mecanismos chamados de esquemas de controle de concorrência

Controle de Concorrência

É o gerenciamento das operações concorrentes no BD, garantindo que muitos usuários, ao tentar atualizar o mesmo dado, o façam de um modo controlado, para assegurar que os resultados das atualizações sejam corretos **(Isolamento)**.

O padrão SQL define quatro níveis de isolamento de transação em termos de três fenômenos que devem ser evitados entre transações simultâneas. Os fenômenos não desejados são:

? dirty read (leitura suja): A transação lê dados escritos por uma transação simultânea não efetivada (uncommitted).

? nonrepeatable read (leitura que não pode ser repetida) :A transação lê novamente dados lidos anteriormente, e descobre que os dados foram alterados por outra transação (que os efetivou após ter sido feita a leitura anterior). 2

? phantom read (leitura fantasma):A transação executa uma segunda vez uma consulta que retorna um conjunto de linhas que satisfazem uma determinada condição de procura, e descobre que o conjunto de linhas que satisfazem a condição é diferente por causa de uma outra transação efetivada recentemente.

Técnicas de Controle de Concorrência:

- Bloqueio (lock): é uma variável associada a um item de dados que descreve a condição do item em relação às possíveis operações que podem ser aplicadas a ele. Geralmente, há um bloqueio para cada item de dado no banco de dados e eles são usados como meio de sincronizar o acesso por transações concorrentes aos itens do banco de dados.

- Ordenação de Timestamp: Para cada transação do sistema associamos um TS(Ti). Se uma nova transação entrar no sistema receberá TS(Tj) onde TS(Ti) < TS(Tj). O protocolo de ordenação por timestamp garante que qualquer operação read/write em conflito seja executada em ordem de timestamp.

- Multiversão com ordenação de Timestamp: A cada transação Ti do sistema é associado um timestamp único e estático, denotado por TS(Ti) (associado antes do início da execução da transação). Para cada item de dado Q, uma seqüência de versões <Q1, Q2, ..., Qn> é associada.

- Multiversão com bloqueio: Tenta combinar as vantagens do controle de concorrência multiversão com as vantagens do bloqueio. Neste esquema as

transações de atualização executam um bloqueio enquanto as operações de leitura acessam as versões anteriores do dado .

4.5 Controle de Transação em Sql (Commit, Rollback, Savepoint)

O Oracle assegura a consistência dos dados baseado nas transações. As transações dão mais flexibilidade e controle quando da mudança do conteúdo das tabelas e asseguram a consistência dos dados em caso de falhas nos processos do usuário ou falhas no sistema.

Uma transação consiste de comandos DML (insert, update,delete,commit,rollback) que fazem uma mudança consistente nos dados. Por exemplo, uma transferência de valores entre duas contas bancárias implica no débito em uma conta e no crédito em outra no mesmo montante. Ambas as ações ou são realizadas ou são anuladas. O crédito não pode ser concretizado sem o correspondente débito.

Uma transação começa quando o primeiro comando SQL executável é encontrado e termina quando:

- Um comando COMMIT ou ROLLBACK aparece
- Um comando DDL (CREATE, ALTER, DROP, RENAME) ou um comando DCL (GRANT, REVOKE, AUDIT, DROP,RENAME) aparece
- O computador é desligado

Após uma transação terminar, o próximo comando SQL executável automaticamente inicia uma nova transação.

COMMIT: Encerra a transação corrente fazendo com que todas as modificações pendentes passem a ser definitivas.

ROLLBACK: Encerra a transação corrente fazendo com que todas as modificações pendentes sejam desprezadas.

Todas as modificações feitas durante a transação são temporárias até que a transação seja ?commited? (concretizada).

Situação do dado antes do COMMIT ou do ROLLBACK:

- As operações de manipulação de dados primeiramente afetam o buffer do banco de dados.
- O usuário corrente pode rever os resultados das operações de manipulação de dados usando o comando SELECT.
- Outros usuários NÃO podem ver os resultados das operações de manipulação de dados do usuário corrente.
- As linhas afetadas ficam bloqueadas; outros usuários não podem modificar os dados existentes nas linhas afetadas.

Situação do dado depois do COMMIT.

- As modificações são concretizadas no banco de dados. O conteúdo anterior do dado é definitivamente perdido.

- Todos os usuários podem ver os resultados da transação.
- Os bloqueios nas linhas afetadas são desfeitos; as linhas ficam disponíveis para que outros usuários possam executar novas alterações.

Situação do dado depois do ROLLBACK.

- As mudanças são desfeitas. O conteúdo anterior do dado é restabelecido.
- Os bloqueios nas linhas afetadas são desfeitos; as linhas ficam disponíveis para que outros usuários possam executar novas alterações.

Situações nas quais o COMMIT e o ROLLBACK são implícitos.

- Execução de um comando DDL, como um CREATE TABLE
    - COMMIT automático
- Saída normal do SQL*Plus, sem que tenha sido explicitado COMMIT ou ROLLBACK
    - COMMIT automático
- Término anormal do SQL*Plus ou queda do sistema
    - ROLLBACK automático
- Recomendação: Sempre explicitar o COMMIT e o ROLLBACK

O Comando SAVEPOINT

Pode-se criar um ponto de salvamento dentro de uma transação corrente através do uso do comando SAVEPOINT. Assim a transação é dividida em partes menores. Pode-se então descartar-se as alterações pendentes até o ponto de salvamento especificado, através do uso do comando ROLLBACK TO SAVEPOINT.

Um savepoint marca um ponto intermediário no processamento de uma transação.

A sintaxe do comando é:

- **SAVEPOINT** ***nome do ponto de salvamento***
    - Marca um ponto de salvamento dentro da transação.
- **ROLLBACK TO SAVEPOINT** ***nome do ponto de salvamento***

Selecionando o nível de isolamento em SQL

Sintaxe:

SET TRANSACTION [READ WRITE\READ ONLY] [WAIT\NO WAIT]
[[ISOLATION LEVEL] {SNAPSHOT[TABLE STABILIT]\
READ COMMITED [[NO] RECORD_VERSION]}]
[RESERVING nome_tabela, ... [FOR [SHARED/PROTECTED]
{READ\WRITE}], ... ;
Onde:

? READ WRITE/READ ONLY - define o motivo do acesso

? WAIT/NO WAIT - espera ou não em uma fila até o término das transações concorrentes
? ISOLATION LEVEL - como a transação age em relação as transações concorrentes
? SNAPSHOT - a transação retém uma visão estática do BD
? TABLE STABILITY - não pode escrever sobre esta visão
? READ COMMITED - a transação retém uma visão não estática do BD
? NO RECORD_VERSION - visão alterada apenas quando a última transação terminar
? RECORD_VERSION - visão alterada em cada transação encerrada
? RESERVING - reservar previamente as tabelas para evitar o deadlock
? Uma Transação pode atualizar uma Tabela Temporária se estiver especificada READ ONLY
Exemplo

ESTABELECER UMA TRANSAÇÃO QUE UTILIZA AS TABELAS PEÇA, FORNECEDOR E PROJETO

SET TRANSACTION
ISOLATION LEVEL READ COMMITTED
NO RECORD_VERSION WAIT
RESERVING Peça, Fornecedor FOR SHARED WRITE,
Projeto FOR PROTECTED WRITE;

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o conceito transação explicando seus estados e propriedades.

Exemplificar a importância do controle de transações concorrentes para o uso eficiente dobanco de dados.

Explicar o controle de transação em SQL, seja setando o nível de isolamento ou a partir dos camandos de begin transaction, commit, rollback e savepoint, exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

.

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios controle de transação

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados
Cap15 Transações do Livro do Silberchatz
Cap8 Sql99 do livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 16

REVISÃO

Tema

REVISÃO

Palavras-chave

Objetivos

O aluno deverá ser capaz de:

- Escrever comandos de SQL

Estrutura de Conteúdo

ALGEBRA RELACIONAL

A **álgebra relacional** é uma coleção de operações canônicas que são utilizadas para manipular as relações.

SELEÇÃO (σ): operação aplicada sobre uma relação de modo a selecionar um sub-conjunto de tuplas (com todos os seus atributos) que satisfaçam a um determinada condição (simples ou composta).
A operação **SELECT** é denotada por:

$$\sigma_{\text{<condição de seleção>}}(\text{<nome da relação>})$$

PROJEÇÃO (π atributos): operação aplicada sobre uma relação de modo a selecionar os atributos de uma relação de acordo com uma lista de atributos oferecida. Os atributos são exibidos na mesma ordem que aparecem na lista.
A forma geral do operador PROJECT é:

$$\pi_{\text{<lista de atributos>}}(\text{<nome da relação>})$$

JUNÇÃO é a operação utilizada para combinar tuplas relacionadas (via chave primária/chave estrangeira) de duas ou mais relações de modo a estabelecer virtualmente uma única tupla. Esta combinação é realizada de acordo com uma condição indicada.
A forma geral da operação **junção** entre duas tabelas **R** e **S** é a seguinte:

$$\mathbf{R} \;\; _{\text{<condição de junção>}} \; \mathbf{S}$$

LINGUAGEM SQL

Criação de Tabelas
CREATE TABLE [schema].nome_da_tabela

(nome_col1 tipo_col1 [default vl_default_col1] [restri_col1] [,

nome_col2 tipo_col2 [default vl_default_col2] [restri_col2] [,

nome_col3 tipo_col3 [default vl_default_col3] [restri_col3]

]]... [, restri_tab1 [,restri_tab2]);

Eliminação de Tabelas
Drop Table <tabela>

Inserção de Linhas

INSERT INTO schema.<nome tabela> VALUES (valor1, valor2, ..., valorn);

Atualização de Linhas

UPDATE [schema.] nome_tabela
SET coluna1 = expressão I subquery [, colunan = ...]
WHERE condição

Eliminação de Linhas

DELETE [FROM] [schema.]nome_tabela
WHERE condição

Comando Select

Sintaxe Básica

SELECT nome-col1, nome_col2, nome coln
FROM [schema].nome_da_ tabela;
WHERE <condição>
GROUP BY col1, col2
HAVING <condição>
ORDER BY col1, col2

Funções Grupo

- **AVG (x)**
  - Retorna o valor médio da coluna **x.**
  - Exemplo: AVG (salario)

    - Ignora os valores nulos.
- **MAX (x)**
    - Retorna o valor máximo da coluna **x.**
    - Exemplo: MAX (salario)
    - Ignora os valores nulos.
- **MIN (x)**
    - Retorna o valor mínimo da coluna **x.**
    - Exemplo: MIN (salario)
    - Ignora os valores nulos.
- **SUM (x)**
    - Retorna a soma da coluna **x.**
    - Exemplo: SUM (salario)
    - Ignora os valores nulos.
- **COUNT (x)**
    - Retorna o número de valores não nulos da coluna **x.**
    - Exemplo: COUNT (perc_comissao)
- **COUNT (*)**
    - Retorna o número de linhas de uma tabela.
    - Exemplo: COUNT (*)
    - Considera os valores nulos.

Junção

Sintaxe ANSI

select departamento.nome, funcionario.nome
from funcionario ***inner join*** *departamento*
on *funcionario.coddepartamento = departamento.coddepartamento*

Sintaxe Tradicional

select departamento.nome, funcionario.nome
from funcionário, departamento
where funcionario.coddepartamento = departamento.coddepartamento

Tipos Junção

Interior: tabela1 INNER JOIN tabela2 ON condição_de_junção
Exterior:

- Esquerda : tabela1 LEFT JOIN tabela2 ON condição_de_junção
- Direita: tabela1 RIGHT JOIN tabela2 ON condição_de_junção
- Total: tabela1 FULL JOIN tabela2 ON condição_de_junção

Subconsulta
Consulta dentro de outra consulta

SELECT col1, col2
FROM <tabela>
WHERE col3 in (SELECT col4
FROM <tabela>
WHERE <condição>);

Operadores de Conjunto
UNION : combinação de todas as linhas de duas ou mais tabelas participantes do UNION, eliminando as linhas duplicadas resultantes.

UNION ALL: combinação de todas as linhas de duas ou mais tabelas participantes do UNION, mantendo todas as linhas duplicadas.

INTERSECT: interseção entre todas as linhas de duas ou mais tabelas participantes do INTERSECT, ou seja, apenas as linhas comuns entre ela

MINUS : linhas existentes na primeira tabela, mas que não existem na segunda.

Criação de Visões
Sintaxe do comando:

CREATE [OR REPLACE] [FORCE I NOFORCE] VIEW nome_view
[(alias1 [, alias2] [, ...])]
AS subquery

Eliminação de Visões

DROP VIEW nome_view;

Criação de Índices

CREATE INDEX [schema.]nome_indice ON tabela (coluna1 [, coluna2 [,...]])

Eliminando um Índice

DROP INDEX [schema.]nome_índice

Controle de Transação

COMMIT: Encerra a transação corrente fazendo com que todas as modificações pendentes passem a ser definitivas.

ROLLBACK: Encerra a transação corrente fazendo com que todas as modificações pendentes sejam desprezadas.

SAVEPOINT: Marca um ponto intermediário no processamento de uma transação. A sintaxe do comando é:

- SAVEPOINT *nome do ponto de salvamento*
  - Marca um ponto de salvamento dentro da transação.
- ROLLBACK TO SAVEPOINT *nome do ponto de salvamento*

Selecionando o nível de isolamento em SQL

SET TRANSACTION [READ WRITE\READ ONLY] [WAIT\NO WAIT]
[[ISOLATION LEVEL] {SNAPSHOT[TABLE STABILIT]\
READ COMMITED [[NO] RECORD_VERSION]}]
[RESERVING nome_tabela, ... [FOR [SHARED/PROTECTED]
{READ\WRITE}], ... ;

Otimização de Comandos

EXPLAIN PLAN : O comando insere uma linha na PLAN_TABLE para cada passo do plano de execução. A sintaxe do comando é:
EXPLAIN PLAN [SET STATEMENT_ID = 'texto'] [INTO schema.tabela]
FOR comando;

Analisando a PLAN_TABLE

```
SQL> select id,
  2        lpad(' ',2 * level-2) || operation ||
  3        decode (id, 0, ' Custo = ' || COST) operacao,
  4        options,
  5        object_name
  6  from   plan_table
  7  connect by prior id = parent_id
  8  start with id = 0
  9  order by id
 10  /


 ID Operações                 OPTIONS          OBJECT_NAME
--- -------------------------- ---------------- ------------
  0 SELECT STATEMENT Custo = 1
  1   TABLE ACCESS           BY INDEX ROWID    CARGO
  2     INDEX                UNIQUE SCAN       PK_CARGO

3 rows selected.
```

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Deverá ser dada uma ênfase especial a revisão da linguagem SQL Os principais comandos deverão ser relembrados a partir de exemplos de acesso a um banco de dados no computador acoplado ao datashow

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor uma sugestão é passar para os alunos um conjunto de consultas a serem realizadas no banco de dados utilizado para as aulas. Estas consultas deverão abordar os principais aspectos da linguagem SQL com especial ênfase em funções de grupo, clausula group by e having, junções e subconsultas.

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.
Laboratório com cliente de banco de dados
Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados
Cap 8 do livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 4

Operações de Seleção e Projeção

Tema

Algebra Relacional - Seleção e Projeção

Palavras-chave

Seleção, Projeção

Objetivos

O aluno deverá ser capaz de:

- Identificar as operações básicas da álgebra relacional
- Construir expressões algébricas de seleção e projeção

Estrutura de Conteúdo

UnidadeI ? Modelo Relacional

1.4 Algebra Relacional

A **álgebra relacional** é uma coleção de operações canônicas que são utilizadas para manipular as relações. Estas operações são utilizadas para selecionar tuplas de relações individuais e para combinar tuplas relacionadas de relações diferentes para especificar uma consulta em um determinado banco de dados. O resultado de cada operação é uma nova operação, a qual também pode ser manipulada pela álgebra relacional.

Os operadores da álgebra relacional podem ser divididos em dois grupos:

- Operadores de Conjuntos: são operadores típicos definidos pela álgebra para conjunto, tais como união, interseção, diferença e produto cartesiano.
- Operadores de Tabelas: são operadores especiais definidos especialmente para a manipulação de tuplas, tais como, Select, Project e Join.

1.4.1Operação e Seleção e Projeção

SELEÇÃO (σ): operação aplicada sobre uma relação de modo a selecionar um sub-conjunto de tuplas (com todos os seus atributos) que satisfaçam a um determinada condição (simples ou composta). O sub-conjunto selecionado forma uma relação resultante temporária. Esta condição aplica-se apenas em uma única relação sendo verificada individualmente para cada tupla da relação. Exemplo: Dado uma relação dos funcionários da empresa, selecionar aqueles que recebem salário > 1000,00 reais.

A operação **SELECT** é denotada por:

$$\sigma_{\text{<condição de seleção>}}(\text{<nome da relação>})$$

A letra grega σ é utilizada para representar a operação de seleção; **<condição de seleção>** é uma expressão *booleana* aplicada sobre os atributos da relação e **<nome da relação>** é o nome da relação sobre a qual será aplicada a operação **select**.

A relação resultante da operação SELECT tem os mesmos atributos da relação especificada em <nome da relação>.
A expressão Booleana especificada em <condição de seleção> é construída a partir de cláusulas da forma:

<nome de atributo> <operador de comparação> <valor constante>, ou
<nome de atributo> <operador de comparação> <nome de atributo>

Onde <nome de atributo> é o nome de um atributo da <nome da relação>, <operador de comparação> é normalmente um dos operadores relacionais {=, <, =, =, ?} e <valor constante> é um valor constante. As cláusulas podem ser utilizadas em conjunto com os operadores lógicos {AND, OR NOT}, seguindo a Lógica Booleana, para formar uma condição de seleção composta.
Exemplo, suponha que se deseja selecionar as tuplas de todos os empregados que ou trabalham no departamento 4 e faz mais de 2500 ou trabalha no departamento 5 e faz mais que 3000.
Neste caso, pode-se especificar a consulta da seguinte forma:
σ (NDEP = 4 AND SALÁRIO > 2500) OR (NDEP = 5 AND SALÁRIO > 3000) (EMPREGADO)
O operador SELECT é **comutativo**; isto é:
σ<cond1> (σ<cond2> (R))= σ<cond2> (σ<cond1> (R))
Assim, uma seqüência de SELECTs pode ser aplicado em qualquer ordem. Além disso, pode-se sempre trocar operadores SELECT em cascata com a conjuntiva AND; isto é:

σ<cond1> (σ<cond2> (...σ<condn> (R) ...))= σ<cond1> AND <cond2> AND ... AND <condn>(R)

PROJEÇÃO (π atributos): operação aplicada sobre uma relação de modo a selecionar os atributos de uma relação de acordo com uma lista de atributos oferecida. Os atributos são exibidos na mesma ordem que aparecem na lista. Como resultado tem-se uma relação onde não existem repetições de tuplas. Exemplo: Projetar apenas o nome, idade e salário de uma relação de funcionários. A relação resultante é uma tabela sem repetições. Caso na lista de atributos estiver incluída a chave primária da relação então tem-se certeza que não há repetições.
A forma geral do operador PROJECT é:

$$\pi_{\text{<lista de atributos>}}(\text{<nome da relação>})$$

A letra grega **π** representa a operação **project**, **<lista de atributos>** representa a lista de atributos que o usuário deseja selecionar e **<nome da relação>** representa a relação sobre a qual a operação **project** será aplicada.

Sequencialidade de Operações

As operações **project** e **select** podem ser utilizadas de forma combinada, permitindo que apenas determinadas colunas de determinadas tuplas possam ser selecionadas.

A forma geral de uma operação sequencializada é:

$$\pi_{\text{<lista de atributos>}} (\sigma_{\text{<condição de seleção>}} (\text{<nome da relação>}))$$

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

1. Algebra Relacional

Apresentar aos a definição de álgebra relacional e os seus tipos de operações

Explicar a diferença entre as operações de tabelas e as operações de conjunto

2. Operações de Seleção e Projeção

Explicar a operação e seleção e dar exemplo de seu uso.

Explicar a operação de projeção e exemplificar o seu uso

Exemplificar a utilização das duas operações em conjunto

Para melhor ilustrar estas operações o professor deverá apresentar aos alunos um esquema de banco de dados e realizar as operações sobre as tabelas mostrando as linhas que resultariam em cada operação

Após a explicação deverão ser apresentados aos alunos diversos exercícios que contemplem as operações vista de forma a fixar a aprendizagem

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Este banco pode ser bem simples com duas tabelas , por exemplo empregado e departamento, onde em empregado deverá existir uma chave estrangeira para departamento.

O professor poderá mostrar para os alunos através de comandos de SQL a execução das operações algébricas apresentadas

III. Discussão com a turma:

Exemplos de uso de operações de Seleção e Projeção

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 2 livro Sistema de banco de dados do Silberschatz.

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 5

Operações de Conjunto e Junção

Tema

Algebra Relacional - Operações de Conjunto e Junção

Palavras-chave

Conjunto, Modelo Relacional

Objetivos

O aluno deverá ser capaz de:

- Identificar as operações de conjunto
- Diferenciar os tipos de junção
- Construir expressões algébricas de operações de conjunto e junção

Estrutura de Conteúdo

UnidadeI ? Modelo Relacional

1.4 Algebra Relacional

1.4.2Operações de Conjunto

Operações que se aplicam a duas relações. Em termos das relações (A e B) usadas nas operações: - ambas podem estar fisicamente armazenadas no mesmo Banco de Dados; ou - estarem fisicamente armazenadas em diferentes Bancos de Dados; ou - uma estar armazenada e a outra ser originária (relação resultante) de uma operação de conjunto anteriormente executada; ou - ambas serem resultantes de operações de conjunto realizadas anteriormente.

UNIÃO (∪): o resultado da união de duas relações consiste no conjunto de todas as tuplas das duas relações, porém sem redundância de tuplas. Importante salientar que duas tuplas são semelhantes quantos todos os valores de atributos forem iguais em seus respectivos atributos. Exemplo: União de uma relação dos funcionários da matriz de São Paulo com a relação dos funcionários da filial de Campinas.

INTERSEÇÃO (∩): o resultado da interseção de duas relações consiste no conjunto de todas as tuplas que pertençam às duas relações. Exemplo: Interseção da relação dos funcionários com idade maior que 35 anos com a relação dos funcionários com salário acima de 1500,00 reais. O resultado será uma relação com os funcionários com idade maior que 35 anos e que recebem acima de 1500,00 reais

DIFERENÇA (-): a diferença entre duas relações é relação resultante formada pelas tuplas que pertencem a uma relação e que não pertencem a outra. Diferença entre as relações A e B (A - B) é o conjunto de tuplas que aparecem na relação A e não aparece na relação B. Diferença entre as relações B e A (B - A) é o conjunto de tuplas que aparecem na relação B e não aparecem na relação A. Exemplo: Diferença da relação dos funcionários com idade maior que 35 anos com a relação dos funcionários com salário acima de 1500,00 reais. O resultado será uma relação com os funcionários com idade maior que 35 anos e que não recebem acima de 1500,00 reais.

PRODUTO CARTESIANO (X): aplica-se a duas relações que não necessitam ser "compatíveis para união", resultando em uma relação que apresenta tuplas formadas pela combinação de todas as tuplas de uma relação com todas as tuplas da outra relação (*R(A1, A2, ..., An) X S(B1, B2, ..., Bm) = Relação Produto(A1, A2, ..., An, B1, B2, ...,Bm).* Exemplo: Produto Cartesiano de uma relação de todos os alunos do terceiro ano de seu curso com uma relação com todas as disciplinas do terceiro ano de seu curso. O resultado será uma relação de todos os alunos do terceiro ano e suas disciplinas.

DIVISÃO (/): V A operação DIVISÃO, indicada por /, é útil para um tipo especial de consulta que, às vezes, ocorre em aplicações de banco de dados. Um exemplo é "Recuperar os eleitores que participaram de todas as eleições. Iniciamente devesse produzir a relação dos eleitores e das eleições que participaram. A seguir dividir esta relaçção pela lista de todas as eleições, o resultado será uma lista dos eleitores que participaram de todas as eleições.

1.4.3 Junção

JUNÇÃO é a operação utilizada para combinar tuplas relacionadas (via chave primária/chave estrangeira) de duas ou mais relações de modo a estabelecer virtualmente uma única tupla. Esta combinação é realizada de acordo com uma condição indicada. Exemplo: Junção da relação de peças fornecidas com a relação de fornecedores (em comum as duas relações possuem o código do fornecedor). A relação resultante terá tuplas contendo o código do fornecedor, demais atributos do fornecedor e os atributos de peça, estando juntas apenas as peças e seus respectivos fornecedores. Tuplas cujos valores dos atributos join são null não aparecem no resultado.

A operação Junção denotada pelo operador (?join?) onde a condição for uma expressão explícita de comparação qualquer (=, <, <=, >, >=, <>) é denominada de **Theta Join** (θ condição), exemplo: fornecedor.código <> peça.código. Quando o operador de comparação for o de igualdade, teremos o **Equi Join** (= condição) (exemplo: fornecedor.código = peça.código) Como resultado de um Equi Join teremos tuplas apresentando pares de atributos com valores idênticos. O **Natural Join** (* condição) consiste em uma operação na qual do cruzamento de uma chave primaria e uma chave estrangeira, apenas a chave primária aparece na relação resultante. Quando os atributos

sobre os quais se aplicam o Natural Join apresentarem o mesmo nome em ambas as relações, a condição do Join pode ser totalmente omitida.
A forma geral da operação **junção** entre duas tabelas **R** e **S** é a seguinte:

R <condição de junção> **S**

O resultado da Junção é uma relação Q com n+m atributos Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m) nesta ordem; Q tem um tupla para cada combinação de tuplas ? uma de R e uma de S ? *onde quer que a combinação satisfaça a condição join*. Esta é a principal diferença entre Produto Cartesiano e Junção, na junçãoapenas combinações de tuplas que satisfazem a condição de junção é que aparecerá no resultado, já no produto cartesiano todas as combinações de tuplas são incluídas no resultado. cada combinação de tuplas.

Outras formas explícitas de variações da operação de ?join? são:

- LEFT OUTER JOIN : Exemplo - A LEFT OUTER JOIN B permite a junção da relação A com a relação B, sendo colocadas nas relação resultante todas as tuplas da relação A mesmo que não tenham correspondentes na relação B.
- RIGHT OUTER JOIN : Exemplo - A RIGHT OUTER JOIN B permite a junção da relação A com a relação B, sendo colocadas nas relação resultante todas as tuplas da relação B mesmo que não tenham correspondentes na relação A.
- FULL OUTER JOIN : Exemplo - A FULL OUTER JOIN B permite a junção da relação A com a relação B, sendo colocadas nas relação resultante todas as tuplas da relação A e da relação B mesmo que não tenham correspondentes.

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

1.Operações de Conjunto:

Apresentar as operações de conjuntos união, intersecção, diferença, produto cartesiano e divisão exemplificando seu uso a partir de um esquema de banco de dados de exemplo

2. Junção

Apresentar os diversos tipos de junção, natural, interior e exterior.

Explicar a diferença entre junção interior e exterior, exemplificando a partir de uma esquema de banco de dados

Para melhor ilustrar estas operações o professor deverá apresentar aos alunos um esquema de banco de dados e realizar as operações sobre as tabelas mostrando as linhas que resultariam em cada operação

Exercícios

Após as explicações deverão ser apresentados aos alunos diversos exercícios que contemplem as operações vista de forma a fixar a aprendizagem

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Este banco pode ser bem simples com duas tabelas , por exemplo empregado e departamento, onde em empregado deverá existir uma chave estrangeira para departamento.

O professor poderá mostrar para os alunos através de comandos de SQL a execução das operações junção (interior e exterior) e das operações de conjunto union, minus e intersection

III. Discussão com a turma:

Exemplos de uso de operações de Operações de Conjunto e Junção

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.
Laboratório com cliente de banco de dados
Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados
Cap 2 livro Sistema de banco de dados do Silberschatz.

Avaliação

Avaliação da solução dos alunos para os exercícios passados

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 6

DDL - TABELAS

Tema

SQL - TABELAS

Palavras-chave

DLL, DRL, DML

Objetivos

O aluno deverá ser capaz de:

- Distinguir os SubConjuntos da Linguagem Sql

Escrever Comandos para Criar alterar e eliminar tabelas

Estrutura de Conteúdo

UnidadeII ? Linguagem Sql

2.1.Comandos DDL

SQL quer dizer Structured Query Language e é o padrão mundial de acesso às bases de dados relacionais.

A Linguagem SQL é dividida nas seguintes partes:

- **DDL - Create, Alter, Drop, Rename, Truncate:** Permitem a criação e definição de objetos como tabelas, views e outros objetos no banco de dados.
- **DRL - Select:** É o comando mais comum do SQL. Utilizado amplamente para recuperação dos dados de uma base.
- **DML - Insert, Delete, Update:** Comandos de manipulação dos dados. Usados nas aplicações que mantém a base de informações com inserções, atualizações e deleções de dados.
- **DCL - Grant, Revoke:** São utilizados para atribuir ou remover direitos de acesso a objetos do banco de dados e suas estruturas.

2.2 Criação de Tabelas

Elementos que Compõem uma Tabela

A tabela é a forma básica de armazenamento de informações em um sistema gerenciador de banco de dados relacional e por isso deve conter um conjunto de elementos, alguns opcionais, na sua composição e que são:

- Nome
- Colunas
- Tipos de Dados
- Restrições (Constraints)

Nome:

Em geral é herdado do nome da entidade que a originou no modelo conceitual, apenas colocando-o no plural, em maiúsculas, sem espaços ou caracteres especiais, sendo estes substituídos por sublinhado (underscore).

Exemplo

Entidade	**Tabela**
EMPREGADO	EMPREGADOS
ITEM NOTA FISCAL	ITENS_NOTAS_FISCAIS

Colunas

Tal como o nome da tabela, origina-se do modelo conceitual, sendo mapeados dos atributos da entidade, sem acentos e caracteres especiais, em minúsculas e, preferencialmente, utilizando nomes os mais curtos possíveis de forma significativa.

Exemplo

Atributo	**Coluna**
matrícula do empregado	matric_emp
Data de validade	data_valid

Tipos de Dados

Cada coluna de uma tabela, tem que ter um tipo de dado definido e único, obedecendo a lista de tipos definidos pelo RDBMS. No Oracle existe uma diversidade bastante grande de tipos e alguns são descritos na tabela abaixo:

Tipos de Dados	**Descrição**
VARCHAR2(tam)	Caracter de tamanho variável, podendo atingir de 1 até 4000 bytes, sendo limitado a *tam.*
CHAR(tam)	Caracter de tamanho fixo, podendo *tam* ser especificado no máximo de 2000 caracteres.
NUMBER	Número de ponto flutuante podendo atingir a magnitude entre 1.0×10^{-130} e $9.9....9 \times 10^{125}$ dígitos significativos.
NUMBER(I,d)	Numérico com precisão, podendo atingir em i de 1 a 38 dígitos significativos inteiros e d representa o valor máximo em decimais, que pode variar de ?84 a 127.
DATE	Data e hora, podendo ser representados entre 01 de janeiro de 4712 AC até 31 de dezembro de 4712 DC.
LONG	Caracter de tamanho variável, podendo atingir até 2 Gb.
ROWID	String base 64 representando o endereço único de uma linha numa tabela, através da pseudo coluna ROWID

A Criacão de Tabelas - Sintaxe

A estrutura básica do comando é:

Sintaxe

```
CREATE TABLE [schema].nome_da_tabela
      (nome_col1 tipo_col1 [default vl_default_col1] [restri_col1] [,
       nome_col2 tipo_col2 [default vl_default_col2] [ restri_col2 ] [,
       nome_col3 tipo_col3 [default vl_default_col3] [restri_col3]
        ]]...   [,  restri_tab1   [,restri_tab2] );
```

Onde:

Palavra Chave	**Descrição**
Schema	É o schema onde a tabela será criada. Por

	default é o próprio schema do usuário que está criando.
nome_da_tabela	Nome que será atribuído a tabela
Nome_coln	Tipo de dado atribuído à coluna n
default vl_default_coln	Valor default a ser atribuído à coluna, podendo ser um literal, ou uma expressão, função, de mesmo tipo que o atribuído a coluna. Este valor é automaticamente atribuído ao campo caso nenhum valor seja especificado.
restri_coln	Define uma restrição de integridade automática a qual os campos da coluna devem obedecer.
restri_tabn	Define uma restrição de integridade automática a qual toda a tabela deve obedecer

Restrições (Constraints)

Um conjunto de Restrições pode ser garantido automaticamente pelo SGBDe, quando implementadas em tempo de criação ou alteração da tabela.

Estas restrições são:

Restrição (Constraint)	**Descrição**
NOT NULL (Obrigatório)	Especifica se a coluna é obrigatória para todas as linhas da tabela, não podendo conter nenhum valor NULO.
UNIQUE (Chave Alternada)	A coluna ou combinação de diversas colunas, tem que ser única para todas as linhas da tabela, não permitindo repetições.
PRIMARY KEY (Chave Primária)	É a chave primária de identificação unívoca da tabela. Pode ser uma ou uma combinação de colunas.
FOREIGN KEY (Chave Estrangeira)	É uma coluna que garante a integridade de uma relação entre duas tabelas, sendo referenciada por uma chave primária da outra tabela.

CHECK	Especifica uma condição que tem que ser verdadeira.

2.2 Alterando Tabelas

A estrutura de uma tabela pode ser alterada pelo comando:
Alter Table < tabela> [modificação].

Entre as modificações que pode ser realizadas temos:

- Acrescentar ou eliminar uma coluna (ADD/DROP Column)
- Adicionar um Constraint (ADD Constraint)
- Habilitar e Desabilitar Constraint (Disable/Enable Constraint)

2.3 Eliminando Tabelas
As tabelas são eliminadas pelo comando:

Drop Table <tabela>

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

1.Linguagem SQL:

Apresentar as partes que compõem a linguagem SQL exemplificando o seu uso.

2. Criação, Alteração e Eliminação de Tabelas

Apresentar os comando Create ,Alter de Drop exemplificando o seu uso

Para melhor ilustrar estas operações o professor deverá apresentar aos alunos um esquema de banco de dados e realizar as operações de DDL de tabelas.

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser emininentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de criação de tabelas, alteração de seus esquema e eliminação de tabelas.

III. Discussão com a turma:

Exemplos de uso de comandos de DDL

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.
Laboratório com cliente de banco de dados
Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados
Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 7

Comandos DML

Tema

INSERT, UPDATE E DELETE

Palavras-chave

DML, SQL

Objetivos

O aluno deverá ser capaz de:

- Escrever Comandos para Inserir linhas
- Escrever comandos para alterar linhas

Escrever comandos para eliminar linhas

Estrutura de Conteúdo

UnidadeII ? Linguagem Sql

2.2.Comandos DML

Os comandos de DML nos permitem incluir, alterar ou eliminar linhas das tabelas

2.2.1 Comando Insert

O comando INSERT insere linhas em uma tabela. A forma mais simples do comando INSERT insere somente uma linha , dados os valores conforme a sintaxe abaixo:

```
insert into     schema.<nome_tabela> (coluna1, coluna2, ..., colunan)
                values      (valor1, valor2, ..., valorn);
```

Onde:

Cláusula	**Descrição**
schema	E o schema que contém a tabela
nome_tabela	O nome da tabela a ser atualizada
Coluna n	A coluna que queremos inserir
Valor n	É o novo valor associado à coluna a ser inserida

Vejamos um rápido exemplo de inserção na tabela de departamento, dentro do uma Linha a Linha.

```
INSERT  INTO  C_depto   (id, nome ,id_Regiao )
            VALUES ( 200, 'Meu depto 2', 2);
```

Nota:

- Observe que o comando informa as 3 colunas da tabela C_Depto e, em seguida, define 3 valores que a elas serão atribuídas (na mesma ordem)
- Assim: a coluna *id* receberá o valor numérico 200,
  a coluna *nome* receberá o valor alfanumérico 'Meu depto 2? e
  a coluna *id_regiao* receberá e valor 2.

Inserindo sem Referenciar as Colunas
Vamos analisar um outro comando INSERT, semelhante ao comando acima e que tem a seguinte sintaxe:

insert into schema.<nome tabela> values (valor1, valor2, ..., valorn);

Exemplo
INSERT INTO C_Depto VALUES (201, 'Meu Depto 3', 3);

Nota:

- Desta vez não foram especificadas as colunas que receberão os valores. Neste caso o comando utilizará todas as colunas da tabela na ordem em que foram criadas.
- Você pode utilizar o comando DESCRIBE do SQL*Plus para verificar as colunas de uma tabela e sua ordem.

lnserindo com Valores Nulos

Caso alguma coluna deva ficar com valor NULO em uma inserção, basta omitir o nome da mesma na lista de colunas. Vejamos um exemplo onde isto ocorre:

INSERT INTO C_DEPTO (id, nome) VALUES (60, ?Exemplo?);

2.2.2 Comando Update

Para alterarmos dados já existentes em nossas tabelas utilizaremos o comando UPDATE.

Veja a sintaxe do comando abaixo:

UPDATE [schema.] nome_tabela

SET coluna1 = expressão I subquery [, colunan = ...]
WHERE condição

Onde:

Cláusula	Descrição
schema	É o schema que contém a tabela
Nome_tabela	O nome da tabela a ser atualizada
colunan	A coluna que queremos alterar
expressão	É o novo valor associado à coluna a ser alterada
subquery	Um comando SELECT que retornará o novo valor da coluna
condição	A condição que deverá satisfazer as colunas que serão alteradas

Vejamos um exemplo que provoca um aumento de 50% em todos os salários da empresa.

```
UPDATE C_EMPR
  SET SALARIO = SALARIO * 1.5;
```

Restringindo a Atualização

Podmeos limitar as linhas recuperadas, indicando ao comando UPDATE que altere apenas parte da tabela de acordo com meu interesse.

Vamos alterar o comando acima para que altere apenas os salários menores ou iguais a 1500:

```
UPDATE   C_EMPR

SET SALARIO = SALARIO * 1.5
WHERE     SALARIO <= 1500;
```

Atualização Múltipla

Posso também alterar várias colunas ao mesmo tempo, evitando a execução múltipla do comando.

Vejamos um exemplo no qual os empregados de id 20 a 23 tem seu salário alterado para 100 e passam a ser subordinados ao empregado 10:

```
UPDATE    C_EMPR
```

SET ID_GERENTE = 10,

SALARIO = 100

WHERE ID IN (20,21, 22, 23) ? ;

2.2.3 Comando Delete

Para excluirmos linhas em uma tabela utilizamos o comando DELETE.

Veja a sintaxe do comando abaixo:

DELETE [FROM] [schema.]nome_tabela

WHERE condição

Onde:

Cláusula	**Descrição**
schema	E o schema que contém a tabela
Nome_tabela	O nome da tabela a ser deletada
condição	A condição que deverá satisfazer as colunas que serão deletadas

Exemplo
DELETE FROM C_NIVEL_SALARIAL;

Nota:

- Com este comando, todas as linha da tabela NIVEL SALARIAL foram removidas.

Restringindo a Deleção
Para executar uma exclusão selecionada utilize a cláusula WHERE:

DELETE FROM C_NIVEL_SALARIAL

WHERE NIVEL <= 3;

Nota:

- Neste exemplo excluímos todas as linhas da tabela c_nível_salarial que possuíam ID com valor igual ou menor a 3.

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

1.Comandos de DML:

Apresentar os comandos de DML (insert update delete) exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de inserção, alteração e eliminação de linhas nas tabelas do banco de dados.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso de comandos de DML

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 3

Select uma Tabela

Tema

Comando Select

Palavras-chave

Comando Select

Objetivos

O aluno deverá ser capaz de:

- Escrever Comandos de select para recuperar dados de uma tabelas
- Utilizar os operadores lógicos e de comparação para restringir a consulta

Estrutura de Conteúdo

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o comando Select exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

Destacar o uso dos diversos operadores como in, between . like etc... sempre exemplificando o uso a partir de consultas realizadas em um banco de dados

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de consultas as tabelas do banco de dados.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.
Laboratório com cliente de banco de dados
Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados
Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 2

Select uma Tabela - Ordenação

Tema

Alias, Expressões , Order By e Distinct

Palavras-chave

Alias, Expressões , Order By e Distinct

Objetivos

O aluno deverá ser capaz de:

- Escrever Comandos de select utilizando funções e calculos
- Eliminar linhas duplicadas no comando de select

Ordenar o resultado da consulta

Estrutura de Conteúdo

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o comando Select exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

Destacar o uso dos alias de colunas ao realizar cálculos ou utilizar funções, bem como mostrar como ordenar o resultado e eliminar as linhas duplicadas.. sempre exemplificando o uso a partir de consultas realizadas em um banco de dados

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de consultas as tabelas do banco de dados.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 8

LINGUAGEM SQL – GROUP BY

Tema

LINGUAGEM SQL – GROUP BY

Palavras-chave

Objetivos

O aluno deverá ser capaz de:

- Escrever Comandos de select utilizando funções de grupo
- Escrever comandos utilizando as clausulas group by e having

Estrutura de Conteúdo

UnidadeII – Linguagem Sql

2.3.Comando Select

2.3.5 Funções de Grupo, cláusulas GROUP BY e HAVING

Funções de Grupos

Funções de grupo operam sobre conjuntos de linhas. Elas retornam resultados baseados sobre um grupo de linhas, antes que um resultado por linha tenha retornado como uma função de linha única. Como padrão todas as linhas de um tabela são trilhadas como um grupo. A clausula GROUP BY da declaração do SELECT é usada para agrupar as linhas em menores grupos.

- Funções de Grupo
  - **AVG (x)**
    - Retorna o valor médio da coluna **x.**
    - Exemplo: AVG (salario)
    - Ignora os valores nulos.
  - **MAX (x)**
    - Retorna o valor máximo da coluna **x.**

- Exemplo: MAX (salario)
- Ignora os valores nulos.

- **MIN (x)**
  - Retorna o valor mínimo da coluna **x.**
  - Exemplo: MIN (salario)
  - Ignora os valores nulos.
- **SUM (x)**
  - Retorna a soma da coluna **x.**
  - Exemplo: SUM (salario)
  - Ignora os valores nulos.
- **COUNT (x)**
  - Retorna o número de valores não nulos da coluna **x.**
  - Exemplo: COUNT (perc_comissao)
- **COUNT (*)**
  - Retorna o número de linhas de uma tabela.
  - Exemplo: COUNT (*)
  - Considera os valores nulos.

DISTINCT faz uma função de grupo considerar valores não duplicados; **ALL** considera todos os valores. Se omitida a consulta considera ALL como **default**;

Aqrupando os Resultados da Querv

Outro recurso do comando SELECT é a possibilidade de agruparmos nossos dados, utilizando a cláusula GROUP BY.

Sintaxe:

* SELECT nome da coluna [, nome da coluna]
FROM nome da tabela

WHERE condição

GROUP BY expressão

onde

expressão especifica as colunas cujos valores determinam a base para o grupo de linhas; **group by** produz uma linha sumarizada para cada grupo de linhas selecionado.

Após utilizarmos a cláusula GROUP BY dividir as linhas de uma tabela em um grupo menor. Funções de grupo devem ser usadas para resumir informações por cada grupo.

Restringindo Dados Agrupados

Quando utilizamos a cláusula GROUP BY podemos restringir nossa seleção de dados em dois momentos: antes ou depois do agrupamento (e seus cálculos: médias, totais, etc.).

Decidiremos quando executar esta limitação de acordo com o momento em que a mesma pode ocorrer. Se não utilizarmos para tal decisão o resultado dos cálculos efetuados pela cláusula GROUP BY, podemos limitar nosso resultado através da cláusula WHERE, caso contrário devemos utilizar a cláusula HAVING.

A Cláusula HAVING

A cláusula HAVING tem função semelhante a cláusula WHERE, que é ser o elemento de declaração do Join entre resultados agrupados. Quando temos um comando GROUP BY, o SQL checa se existe pelo menos uma função de agrupamento assim como um elemento agrupador dos dados, caso contrário retornará um erro.

O elemento agrupador, por sua vez, tem que estar referenciado na cláusula GROUP BY e caso o resultado de uma função de agrupamento deva ser submetido a comparação, utiliza-se a cláusula HAVING.

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o comando Select utilizando funções de grupo exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

Destacar o uso das clausulas group by e having sempre exemplificando o uso a partir de consultas realizadas em um banco de dados

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de comando select utilizando funções de grupo e as clausulas group by e having.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 9

LINGUAGEM SQL – JUNÇÃO

Tema

LINGUAGEM SQL – JUNÇÃO

Palavras-chave

Objetivos

O aluno deverá ser capaz de:

- Escrever Comandos de JUNÇÃO INTERIOR
- Escrever Comandos de JUNÇÃO EXTERIOR
- Escrever Comandos de AUTO JUNÇÃO
- Distinguir as sintaxes ANSI e NÃO ANSI de junção

Estrutura de Conteúdo

UnidadeII – Linguagem Sql

2.3.Comando Select

2.3.6 Junção

Junções de tabelas

Um comando SELECT pode fazer uma consulta que traz dados de duas ou mais tabelas. Esse é um processo chamado de junção [join]. As tabelas têm uma coluna em comum que é usado para fazer as junções.

Sintaxe da Junção

Existem duas sintaxes diferentes para junção de tabelas.

Uma delas é a tradicional, a segunda é a Sintaxe ANSI.

Na sintaxe tradicional, na lista do FROM as duas (ou mais) tabelas são especificadas, separadas por vírgulas. Na cláusula WHERE deve haver uma condição ligando as duas, a

condição de junção [join condition]. Na lista de colunas do SELECT podem ser incluídos colunas de qualquer uma das tabelas. Veja um exemplo:

select departamento.nome, funcionario.nome
from funcionário, departamento
where funcionario.coddepartamento = departamento.coddepartamento

A outra forma de sintaxe que pode ser usada é a sintaxe do padrão ANSI SQL. O exemplo anterior, com a sintaxe ANSI, ficaria:

select departamento.nome, funcionario.nome
from funcionario ***inner join*** *departamento*
on *funcionario.coddepartamento = departamento.coddepartamento*

Nessa sintaxe, o *tipo de junção* entre as tabelas deve ser especificado entre elas (veremos os diferentes tipos abaixo) e a condição de junção é especificada com a palavra ON.

Junção interior

O exemplo acima é uma junção interior de tabelas [inner join]. Esse tipo de junção conecta as duas tabelas e retorna apenas as linhas que satisfazem a condição de junção. No exemplo, isso significa que, se existirem funcionários para os quais não há departamento relacionado eles não serão incluídos no resultado. Igualmente, se existirem departamentos que não têm empregados(como 'Contabilidade') , eles não aparecem no resultado.
Uma junção interior é chamada de equijoin quando as colunas são comparadas usando o =, e as duas colunas aparecem no resultado, mostrando dados redundantes, já que elas têm o mesmo valor. Uma junção interior é chamada junção natural quando a coluna usada para junção aparece apenas uma vez no resultado, vinda de uma ou outra tabela.
Na sintaxe ANSI, junções interiores são indicadas da forma:

tabela1 INNER JOIN tabela2 ON condição_de_junção

Apelidos de tabela

Para simplicar a qualificação de colunas, pode-se usar um apelido [alias] de tabela, um nome colocado imediatamente após o nome da tabela, na lista do FROM. Esse nome representa a tabela nas qualificações. Por exemplo, a consulta anterior pode ser reescrita da forma:

select departamento.nome, funcionario.nome
from funcionário f ***inner join*** *departamento d*
on *f.coddepartamento = d.coddepartamento*

Junção cruzada ou irrestrita

Uma *junção cruzada* [cross join] de tabelas, também chamada *junção irrestrita* de duas tabelas gera um resultado formado por *todas* as combinações possíveis de uma linha da primeira tabela com uma linha da segunda. Não existe uma condição de junção. Esse resultado é chamado *produto cartesiano* das duas tabelas. Na sintaxe ANSI, junções cruzadas são indicadas com CROSS JOIN, por exemplo:

```
select  c_depto.id, c_depto.nome,c_regiao.nome
from c_depto cross join c_regiao;
```

Junção exterior

Uma junção exterior [outer join] mostra todas as linhas de uma tabela, mesmo quando elas não satisfazem a condição de junção.

A sintaxe trandicional do outer join no Oracle é:

SELECT nome da tabela1.nome da coluna, nome da tabela2 .nome da coluna ....
FROM nome da tabela1, nome da tabela2
WHERE nome da tabela1.nome da coluna (+) = nome da tabela2.nome da coluna
onde

(+) é o símbolo do outer join, que pode ser colocado em quaisquer dos lados da cláusula where mas não em ambos os lados. Este símbolo deve ser colocado seguindo o nome da coluna que pode não ter correspondente.

Vejamos um exemplo. Quisermos mostrar todos os dados dos departamentos e das regiões que estão relacionados e os dados dos departamentos que não estão relacionados podemos dar o seguinte comando:

*Select * from c_depto, c_região*
Where c_depto.Id_regiao = c_região.ID (+);

Este join que realizamos acima é denominado LEFT OUTER JOIN o que indica que todas as linhas da tabela à esquerda (no caso, 'c_depto') são incluídas no resultado. Nesse caso, são mostrados todos os departamentos, mesmo aqueles que não estão associados a uma região.. Quando um departamento não está associado a alguma regiao , as colunas da tabela 'c_regiao' irá mostrar o valor NULL . A tabela 'c_depto' é chamada de a *tabela exterior* e 'c_regiao' é a *tabela interior* da junção.

Se fosse usado RIGHT OUTER JOIN, a tabela à direita ('c_regiao') mostraria todas as linhas e a tabela à esquerda, apenas as relacionadas, para tal bastaria inverter o simbolo do outer join de lado (+) veja o exemplo abaixo

*Select * from c_depto, c_região*
Where c_depto.Id_regiao (+) = c_região.ID;

As mesma consulta acima em sintaxe ANSI seriam:

Para o LEFT OUTER JOIN

*select **
*from c_depto **left join** c_regiao*
***on** c_depto.id_regiao = c_regiao.id;*

Para o RIGHT OUTER JOIN

*select **
*from c_depto **right join** c_regiao*
***on** c_depto.id_regiao = c_regiao.id;*

Na sintaxe ANSI dispomos ainda do FULL OUTER JOIN onde todas as linhas de ambas as tabelas são incluídas, mesmo as que não estão relacionadas com a outra tabela.

*select **
*from c_depto **full join** c_regiao*
***on** c_depto.id_regiao = c_regiao.id;*

Auto-junções

Uma *auto-junção* [self join] é uma junção da tabela com ela mesma. Na tabela c_empr, por exemplo, cada empregado está subordinado a outro. A coluna 'id_gerente' indica o código do gerente do empregado. Para mostrarmos uma lista de todos os gerentes , cada um com seus subordinados podemos usar:

Select G.id, G.ult_nome,G.cargo, S.id, S.ult_nome, S.cargo
From c_empr G, c_empr S
Where G.id = S.id_gerente

Nesse caso, é obrigatório usar um apelido de tabela para distinguir as duas "cópias" da tabela que estão sendo relacionadas: 'G' no exemplo representa uma linha da tabela 'c_empr' enguanto Gerente e 'S' representa outra linha, de um subordinado, que estão sendo comparadas entre si.

A sintaxe ANSI deste mesmo comando seria

Select G.id, G.ult_nome,G.cargo, S.id, S.ult_nome, S.cargo
From c_empr G inner join c_empr S on G.id = S.id_gerente

Procedimentos de Ensino

I Aula expositiva:

- Utilizar datashow para apresentação da parte teórica;

Apresentar o comando de junção exemplificando seu uso utilizando um banco de dados no computador acoplado ao datashow.

Destacar as diferença de uso entre a junção exterior e a junção interior sempre exemplificando o uso a partir de consultas realizadas em um banco de dados.

Grande destaque deverá ser dado as diferenção entre as sintaxes ANSI e Não ANSI.

II Aula de laboratório

Para a aula de laboratório deverá ser utilizada um cliente de banco de dados e um SGBD com um banco criado.

Sugerimos o uso do Oracle Express por ser livre e se constituir em um SGBD estável e renomado no mercado

A aula deverá ser eminentemente prática onde o professor deverá apresentar aos alunos diversos exercícios de comando de junção interior e exterior, inclusive caso de auto junção que utilizem inner join e outer join.

Sugerimos que em todas as aulas o professor utilize o mesmo banco de dados.

Esta banco de dados deve ter umas 8 tabelas com 15 a 20 linhas em cada tabela.

As tabelas deverão ter pk, fk e as demais restrições normalmente encontradas em banco de dados de produção

III. Discussão com a turma:

Exemplos de uso dos comandos

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais

IMPLEMENTAÇÃO DE BANCO DE DADOS - CCT0835

Semana Aula: 13

SEQUENCES

Tema

SEQUENCES

Palavras-chave

SEQUENCES

Objetivos

CRIAR E GERENCIAR SEQUENCES

Estrutura de Conteúdo

2.5.2 Criando uma sequência (sequence)

Alguns SGBD como oOracle possui internamente uma máquina geradora de números seqüenciais que pode perfeitamente ser usado paraproduzir números únicos, consecutivos e incrementados conforme determinado.

A esses números chamamos de Sequences e são muito utilizados para fazer o papel de ChavesPrimáriasem tabelas onde não existe uma coluna mais apropriada, ou qualquer outra aplicação que haja necessidade de números únicos.

A sintaxe do comando de criação de uma Sequence é:

CREATE SEQUENCE[schema.]sequence_name
[INCREMENT BY n]
[START WITH n]
[MAXVALUE nINOMAXVALUE]
[MINVALUE nINOMINVALUE]
[CYCLE I NOCYCLE]
[CACHE lNOCACHE]
[ORDER I NOORDER]
Onde:

Cláusula	***Descrição***
schema	É o schema onde será gerada a sequence. O default é o schema do usuário que está criando a sequence.
INCREMENT BY	Especifica o intervalo entre os números seqüenciais. Pode ser positivo ounegativo e a omissão significa incremento positivo de 1 em 1.
START WITH	É o primeiro número da seqüência
MAXVALUE	Especifica o valor máximo que a sequence pode alcançar.

NOMAXVALUE	É o default. Especifica um valor com precisão de 27 dígitos positivos ou negativos.
MINVALUE	Especifica um valor mínimo para a sequence.
NOMINVALUE	É o default. Especifica o valor mínimo de 1 para uma sequence ascendente e precisão negativa de 26 dígitos para sequences descendentes.
CYCLE	Especifica que após atingido o valor limite, a sequence recomeçará no MINVALUE ou MAXVALUE
NOCYCLE	É o default. Especifica que a sequence não pode gerar mais números após atingir o limite.
CACHE	Especifica quantos números a sequence pré-aloca e mantém em memória para acesso mais rápido.
NOCACHE	Valores não são pré-alocados em memória.
ORDER	Garante a geração das sequences em ordem de requisição. É o modo default para bancos de dados em modo exclusivo. Caso esteja usando Parallel Server esta opção pode ser necessária para manter a ordem de geração.
NOORDER	Não garante a geração em seqüência dos números tal como foram requisitados.

Exemplo:

CREATE SEQUENCE cod_func_seq
INCREMENT BY 1
START WITH 1
MAXVALUE 1000;

Gerando Sequences com NEXTVAL

Existe uma pseudo-coluna chamada NEXTVAL que é utilizada para extrair valores de uma sequence qualquer, sempre que for referenciada.

Extraindo Valores de uma Sequence com NEXTVAL
SELECT cod_func_seq.NEXTVAL
FROM DUAL;
Nota:

·Esta operação retornaria o valor 1, visto que DUAL é uma tabela do sistema (dummy) utilizada apenas para esses tipos de situações e o comando NEXTVAL retorna o próximo seqüencial. Não se deve esquecer de prefixar NEXTVAL com o nome da sequence.
·A repetição do comando acima retornaria o valor 2 e assim sucessivamente.

Verificando a Sequence com CURRVAL

Para sabermos qual foi o último número gerado pela sequence, devemos utilizar a pseudo-coluna CURRVAL, a qual é sempre alimentada após um NEXTVAL, com o último valor gerado.

SELECT cod_func_seq.CURRVAL
FROM DUAL ;

Nota:

- ***·No nosso exemplo, poderia ser retornado o valor 2 pois fizemos anteriormente, duas referências a NEXTVAL.***

Alterando e Eliminando uma Sequence

Sequences podem ser submetidas a alteração e eliminação tal qual uma tabela, através dos comandos ALTER SEQUENCE e DROP SEQUENCE.

Veja a sintaxe do comando:

ALTER SEQUENCE [schema.]sequence_name
[INCREMENT BY n]
[MAXVALUE n I NOMAXVALUE]
[MINVALUE nI NOMINVALUE]
.....

A Sintaxe para eliminação de uma sequence:

DROP SEQUENCE [schema.]sequence_name;

Procedimentos de Ensino

Estratégias de Aprendizagem

Indicação de Leitura Específica

Recursos

Projetor multimídia com computador acoplado.

Laboratório com cliente de banco de dados

Servidor de banco de dados.

Aplicação: articulação teoria e prática

Site do SGBD utilizado como servidor de banco de dados

Cap 8 livro Sistema de banco de dados do Navathe

Avaliação

Solicitar aos alunos que apresentem suas soluções para os exercícios passados.

Considerações Adicionais