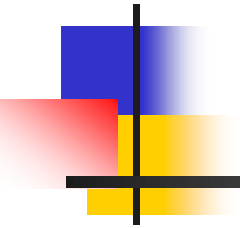
OCA ORACLE DATABASE

SQL EXPERT EXAM

(Exam 1Z0-047)



SQL Oracle: Introdução

andreviniciusnascimento@gmail.com

2012

Conteúdo

- Introdução ao SGBD Oracle
- Objetos do Banco de Dados
- Tipos de Dados Básicos

Introdução ao SGBD Oracle

- Banco de Dados Oracle
  - Principal produto da empresa Oracle.
  - SGBD Relacional extensivamente utilizado para diversos tipos de aplicações e em diversos tipos de plataformas.
  - Lawrence J. Ellison, inspirado pelo trabalho de Edgard F. Codd, fundou em 1977 a empresa Software Development Laboratories.
  - Em 1979 - Oracle V2.
  - Em 1983 - Oracle Corporation , Oracle 3.

Introdução ao SGBD Oracle

- Oracle Server
  - Banco de Dados Oracle (Oracle Database)
  - Instância Oracle (Oracle Server Instance)

Oracle Server = Banco de Dados + Instância

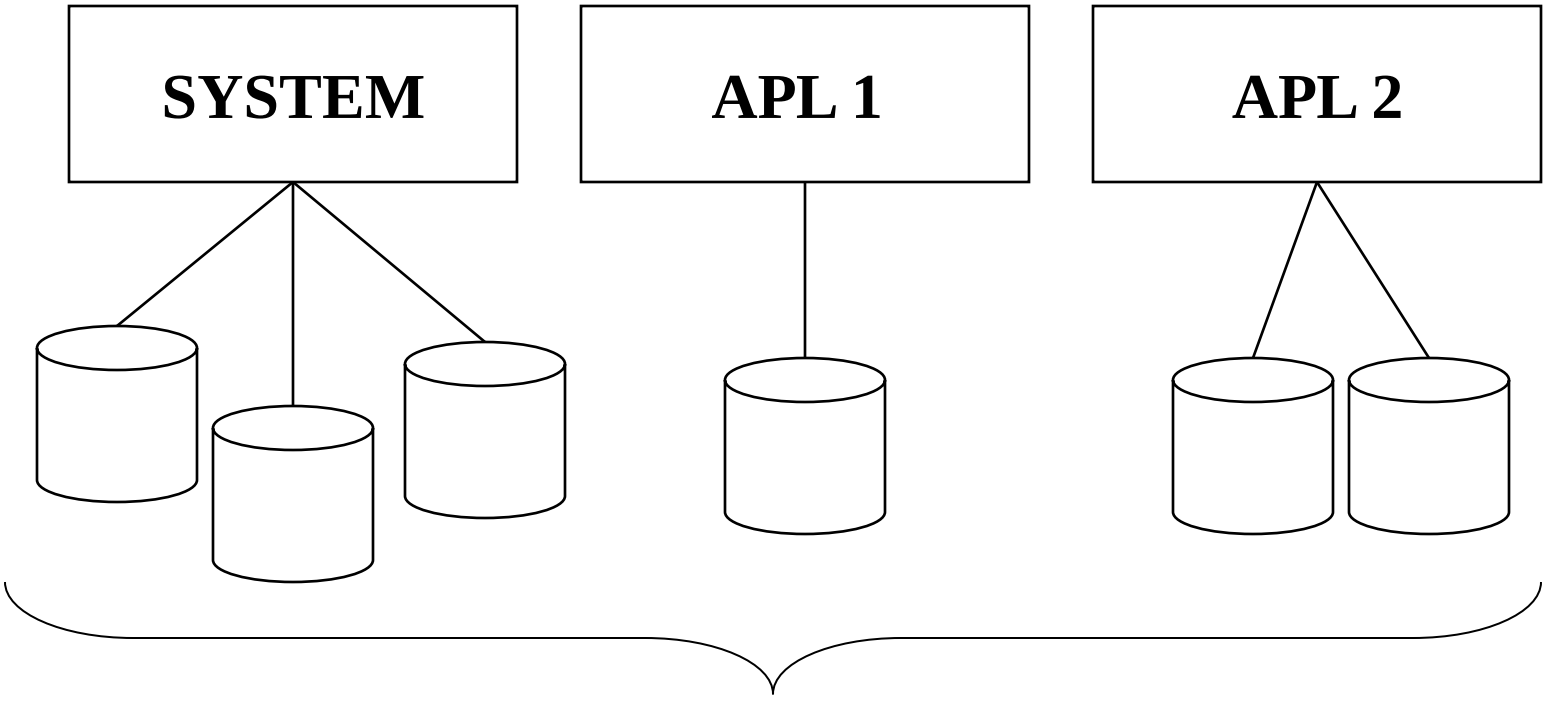
Introdução ao SGBD Oracle

- Banco de Dados Oracle
  - O termo banco de dados em Oracle refere-se ao conjunto de arquivos utilizados para armazenar informações.

- Instância Oracle
  - Conjunto de estruturas de memória e processos que manipulam os arquivos de dados (banco de dados).

Introdução ao SGBD Oracle

- Estrutura Lógica



Introdução ao SGBD Oracle

- Usuários e Esquemas
  - O conceito de Esquema e Usuário em um banco de dados Oracle possui uma correspondência de 1 para 1.
  - Conjunto de Objetos Relacionados.
  - Representa uma organização lógica de objetos.

Objetos do Banco de Dados

- Objetos de Banco de Dados representam a base para aplicações baseadas em Banco de Dados.

- No SGBD Oracle, existem vários tipos de objetos que podem ser criados.

Objetos do Banco de Dados

- Os principais objetos tratados nesse curso são:
  - Tables
  - Constraints
  - Sequences
  - Views
  - Indexes
  - Synonyms
  - Users
  - Roles

Objetos do Banco de Dados

- Todos os objetos do banco de dados Oracle são classificados em uma dessas duas categorias:
  - Schema Objects (Objetos de Esquema)
  - Non-Schema Objects (Objetos não pertencentes a um Esquema)

Objetos do Banco de Dados

- Schema Objects
  - Objetos que podem pertencer a uma conta de usuário.
- Non-Schema Objects
  - Objetos que não podem pertencer a uma conta de usuário.

Objetos do Banco de Dados

- Schema Objects
  - Tables
  - Constraints
  - Sequences
  - Views
  - Indexes
  - Synonyms
- Non-Schema Objects
  - Users
  - Roles
  - Public Synonyms

Tipos de Dados

- O Oracle possui inúmeros tipos de dados. Muitos tipos de dados encaixam-se em uma das seguintes categorias gerais: Cadeias de Caracteres, Números e Datas. Além desses tipos, existem os tipos conhecidos como LOBs (Large Database Objects).

Tipos de Dados

- Embora os LOBs possam conter cadeias de caracteres, eles não podem ser incluidos em: PRIMARY KEY, DISTINCT, GROUP BY, ORDER BY ou JOINS.

Tipos de Dados

- O conjunto básico de tipos de dados que são utilizados pela maioria das aplicações é:
  - CHAR
  - VARCHAR2
  - NUMBER
  - DATE

Tipos de Dados

- CHAR(N)
  - Utilizado para cadeias de caracteres de tamanho fixo
  - N representa um inteiro que indica o tamanho máximo da cadeia de caracteres. N é opcional. Se omitido, o valor assumido é 1.
  - O tamanho máximo é 2000 bytes

Tipos de Dados

- VARCHAR2(N)
  - Utilizado para cadeias de caracteres de tamanho variado
  - N representa um inteiro que indica o tamanho máximo da cadeia de caracteres.
  - A especificação do N é obrigatória.
  - O tamanho máximo é 4000 bytes

Tipos de Dados

- Char é apropriado para atributos com tamanho fixo. Varchar2 é útil para colunas com cadeias de caracteres de tamanho variado.

- Em colunas do tipo char, se o tamanho da cadeia de caracteres for menor que o tamanho máximo permitido, o Oracle preenche o restante com espaços em branco.

Tipos de Dados

- NUMBER(P, S)
  - Utilizado para valores numéricos
  - P indica a precisão (Número total de dígitos)
  - S indica a escala (Número de dígitos à direita do ponto decimal)
  - O valor de P pode ser entre 1 e 38.
  - O valor de S pode ser de -84 a 127

Tipos de Dados

Original	Tipo	Valor Armazenado
123.89	NUMBER	123.89
123.89	NUMBER(3)	124
123.89	NUMBER(3,2)	excede precisão
123.89	NUMBER(4,2)	excede precisão
123.89	NUMBER(5,2)	123.89
123.89	NUMBER(6,1)	123.9
123.89	NUMBER(6,-2)	100
.01234	NUMBER(4,5)	.01234
.00012	NUMBER(4,5)	.00012
.000127	NUMBER(4,5)	.00013
.0000012	NUMBER(2,7)	.0000012
.00000123	NUMBER(2,7)	.0000012

Tipos de Dados

- DATE
  - Utilizado para valores válidos de data e hora
  - 01-01-4712AC 00:00:00 até 31-12-9999DC 23:59:59.

Tipos de Dados

- Existem outros tipos de dados relacionados com data e hora:
  - TIMESTAMP (n)
  - TIMESTAMP (n) WITH TIME ZONE
  - TIMESTAMP (n) WITH LOCAL TIME ZONE
  - INTERVAL Year(n) TO MONTH
  - INTERVAL DAY(n1) TO SECOND (n2)

Tipos de Dados

- LOBs
  - BLOB (Binary Large Object). Aceita objetos grandes binários como imagens e vídeos.
  - CLOB (Character Large Object). Aceita elementos de texto grandes.
  - NCLOB. Aceita dados CLOB em Unicode.

Tipos de Dados

- LOBs podem ser utilizados como os outros tipos de dados.
- Uma tabela pode ter várias colunas do tipo LOB.
- LOBs não podem fazer parte de chave primária e não podem ser utilizados com DISTINCT, GROUP BY, ORDER BY ou JOINS.

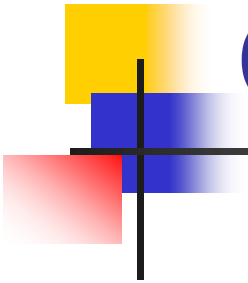
OCA ORACLE DATABASE

SQL EXPERT EXAM

SQL Oracle: Tabelas e Restrições

andreviniciusnascimento@gmail.com

2012



Conteúdo

- Criando Tabelas
- Namespaces
- Alterando Tabelas
- Removendo Tabelas
- Restrições de Chave
- Restrições de Integridade Referencial
- Restrições de Domínio
- Restrições de Tuplas e Relações

Criando Tabelas

- O comando para criação de tabelas (relações) é o CREATE TABLE, cuja sintaxe simplificada é:

```
CREATE TABLE <NOME DA TABELA> (
   <NOME_COLUNA1> <TIPO_DE_DADO>,
   <NOME_COLUNA2> <TIPO_DE_DADO>,
   <NOME_COLUNA3> <TIPO_DE_DADO>,
   ...
)
```

Criando Tabelas

- Exemplos:

```
CREATE TABLE PROPRIETARIO (
   CD_PROPRIETARIO NUMBER(9),
   NM_PROPRIETARIO VARCHAR2(50),
   DT_NASCIMENTO      DATE
)
```

Criando Tabelas

- Nomes de Tabelas e colunas
  - Deve começar com uma letra
  - Deve ter entre 1-30 caracteres
  - Deve conter apenas A-Z, a-z, 0-9, _ , $, e #
  - Não pode duplicar o nome de um outro objeto pertencente ao mesmo usuário
  - Não deve ser uma palavra reservada do servidor Oracle.

Criando Tabelas

- Em geral, os nomes são case insensitive e são tratados como uppercase independente de como são criados ou referenciados.

- A exceção a essa regra acontece quando o nome do objeto é colocado entre aspas " ". Nesses casos, eles passam a ser case sensitive.

Criando Tabelas

- Exemplo:

```
CREATE TABLE “Aplicacao Financeira” (
... )

Select * from “Aplicacao Financeira”
```

Criando Tabelas

- Na definição dos atributos de uma tabela, podemos indicar as restrições NULL ou NOT NULL:

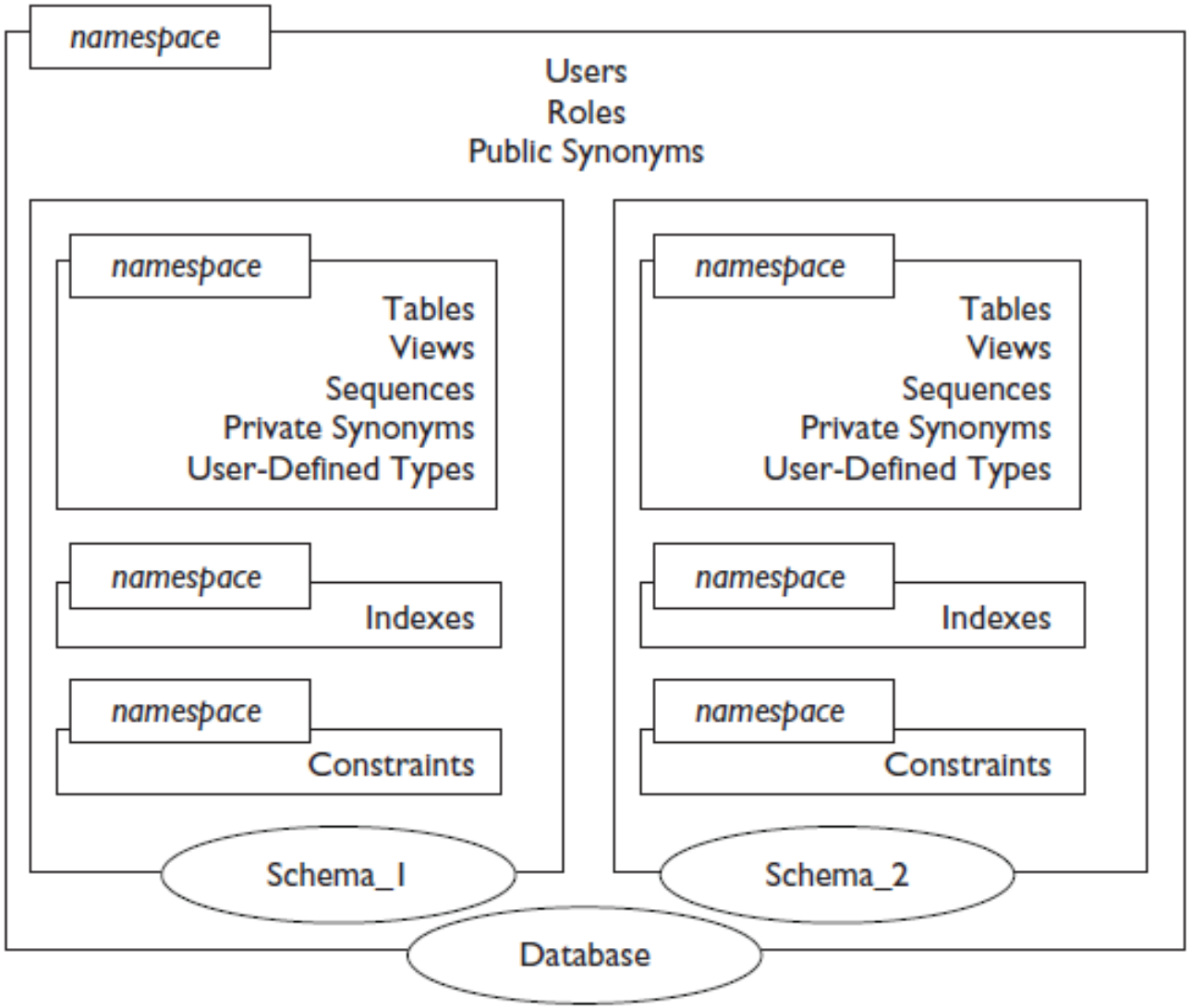
```
CREATE TABLE TIPO_CAO (
    CD_RACA          NUMBER(9)      NOT NULL,
    NM_RACA          VARCHAR2(50)   NOT NULL,
    PORTE            VARCHAR2(10)   NULL,
    PAIS_DE_ORIGEM VARCHAR2(40)   NULL
)
```

Criando Tabelas

- Quando não indicamos restrições de nulidade, o Oracle assume como padrão NULL.

```
CREATE TABLE TIPO_CAO (
    CD_RACA          NUMBER(9)      NOT NULL,
    NM_RACA          VARCHAR2(50)   NOT NULL,
    PORTE            VARCHAR2(10),
    PAIS_DE_ORIGEM VARCHAR2(40)
)
```

Namespaces


namespace
Users
Roles
Public Synonyms
namespace
Tables
Views
Sequences
Private Synonyms
User-Defined Types
namespace
Indexes
namespace
Constraints
Schema_1
namespace
Tables
Views
Sequences
Private Synonyms
User-Defined Types
namespace
Indexes
namespace
Constraints
Schema_2
Database

Alterando Tabelas

- No SGBD Oracle podemos executar os seguintes procedimentos com a finalidade de alterar a estrutura de uma tabela (relação):
  - Adicionar uma coluna
  - Remover uma coluna
  - Modificar as características de uma coluna

Alterando Tabelas

- Para adicionar uma coluna a uma tabela, utilizamos o comando ALTER TABLE ... ADD

```
ALTER TABLE <NOME_DA_TABELA> ADD (
      <DEFINICAO DE COLUNA>,
      <DEFINICAO DE COLUNA>,
      ...
)
```

Alterando Tabelas

- Exemplos:

```
CREATE TABLE EMPREGADO (
     CD_EMPREGADO NUMBER(9)
)

ALTER TABLE EMPREGADO ADD (NOME VARCHAR2(50))

ALTER TABLE EMPREGADO ADD (NOME VARCHAR2(50),
                           TELEFONE NUMBER(10))
```

Alterando Tabelas

- Para remover uma coluna de uma tabela, utilizamos o comando ALTER TABLE ... DROP

```
ALTER TABLE <NOME_DA_TABELA>
DROP COLUMN <NOME_DA_COLUNA>
```

Ou

```
ALTER TABLE <NOME_DA_TABELA> DROP (
      <NOME_DA_COLUNA>,
      <NOME_DA_COLUNA>,
      ...)
```

Alterando Tabelas

- Exemplos:

```
ALTER TABLE EMPREGADO DROP COLUMN NOME

ALTER TABLE EMPREGADO DROP (NOME,
                            TELEFONE)
```

- Obs: Não é possível eliminar todas as colunas de uma tabela

Alterando Tabelas

- Em alguns cenários, é bem mais interessante marcar uma coluna para remoção ao invés de remover. Para isso, utilizamos o comando ALTER TABLE ... SET UNUSED

```
ALTER TABLE <NOME_DA_TABELA>
SET UNUSED COLUMN <NOME_DA_COLUNA>

ALTER TABLE <NOME_DA_TABELA> SET UNUSED (
       <NOME_DA_COLUNA>,
       <NOME_DA_COLUNA>,
       ...
)
```

Alterando Tabelas

- Exemplos:

```
ALTER TABLE EMPREGADO SET UNUSED COLUMN NOME

ALTER TABLE EMPREGADO SET UNUSED (NOME,
                                  TELEFONE)
```

- Em um momento mais apropriado, pode-se remover, efetivamente, as colunas marcadas

```
ALTER TABLE EMPREGADO DROP COLUMNS UNUSED
```

Alterando Tabelas

- Para modificarmos uma coluna de uma tabela, utilizamos o comando ALTER TABLE ... MODIFY

```
ALTER TABLE <NOME_DA_TABELA>
MODIFY <DEFINICAO_DA_COLUNA>

ALTER TABLE <NOME_DA_TABELA> MODIFY (
      <DEFINICAO_DA_COLUNA>,
      <DEFINICAO_DA_COLUNA,
      ...
)
```

Alterando Tabelas

- Exemplos:

```
ALTER TABLE EMPREGADO MODIFY
   (CD_EMPREGADO NUMBER(14))

ALTER TABLE EMPREGADO MODIFY (
      CD_EMPREADO NUMBER(15),
       NOME VARCHAR2(60)
)
```

Removendo Tabelas

- Para remover uma tabela (relação) utilizamos o comando DROP TABLE

```
DROP TABLE <NOME_DA_TABELA>
```

- Exemplo:

```
DROP TABLE EMPREGADO
```

Removendo Tabelas

- Se existirem referências para a tabela a ser removida , as mesmas podem ser removidas automaticamente através da cláusula CASCADE CONSTRAINTS

```
DROP TABLE <NOME_DA_TABELA> CASCADE
  CONSTRAINTS
```

- Exemplo:

```
DROP TABLE EMPREGADO CASCADE CONSTRAINTS
```

Restrições

- A linguagem SQL provê diversos mecanismos para que possamos expressar restrições de integridade

- Restrições de Chave
- Restricões de Integridade Referencial
- Restrições de Domínio
- Restrições de Tuplas e Relações

Restrições de Chave

- Na linguagem SQL podemos expressar a restrição de chave primária através da restrição PRIMARY KEY
- Também podemos expressar a restrição de chave candidata através da restrição UNIQUE
- Ambas as restrições podem ser declaradas no comando CREATE TABLE ou no comando ALTER TABLE.

Restrições de Chave

- Declarando uma restrição de chave primária no comando CREATE TABLE

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO  NUMBER(9) PRIMARY KEY,
    NOME          VARCHAR2(50),
    TELEFONE      NUMBER(10)
)
```

Restrições de Chave

- Declarando uma restrição de chave primária nomeada no comando CREATE TABLE

```
CREATE TABLE EMPREGADO (
  CD_EMPREGADO     NUMBER(9) CONSTRAINT
                   PK_EMPREGADO PRIMARY KEY,
    NOME           VARCHAR2(50),
    TELEFONE       NUMBER(10)
)
```

Restrições de Chave

- Declarando uma restrição de chave primária nomeada no comando CREATE TABLE

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO     NUMBER(9),
    NOME             VARCHAR2(50),
    TELEFONE         NUMBER(10),
    CONSTRAINT PK_EMPREGADO PRIMARY KEY
    (CD_EMPREGADO)
)
```

Restrições de Chave

- Com essa variação, é possível declarar uma chave primária composta por mais de um atributo

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO          NUMBER(9),
    NOME                  VARCHAR2(50),
    TELEFONE              NUMBER(10),
    CONSTRAINT PK_EMPREGADO PRIMARY KEY
    (CD_EMPREGADO, NOME)
)
```

Restrições de Chave

- As restrições de chave primária também podem ser declaradas através do comando ALTER TABLE ADD CONSTRAINT

```
ALTER TABLE EMPREGADO ADD CONSTRAINT
  PK_EMPREGADO PRIMARY KEY (CD_EMPREGADO)

ALTER TABLE EMPREGADO ADD CONSTRAINT
  PK_EMPREGADO PRIMARY KEY (CD_EMPREGADO,
  NOME)
```

Restrições de Chave

- Outras variações

```
ALTER TABLE EMPREGADO ADD PRIMARY KEY
   (CD_EMPREGADO)

ALTER TABLE EMPREGADO MODIFY CD_EMPREGADO
   CONSTRAINT PK_EMPREGADO PRIMARY KEY
```

Restrições de Chave

- As restrições de chave candidata seguem o padrão das definições de chave primária, alterando apenas a cláusula UNIQUE

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO  NUMBER(9) PRIMARY KEY,
    NOME                    VARCHAR2(50),
    CPF                     NUMBER(15) UNIQUE
)
```

Restrições de Integridade Referencial

- Na linguagem SQL podemos expressar a restrição de integridade referencial através da restrição FOREIGN KEY

- Assim como as restrições de chave, as restrições de integridade referencial podem ser declaradas no comando CREATE TABLE ou no comando ALTER TABLE.

Restrições de Integridade Referencial

- Considere o seguinte esquema relacional

EMPREGADO (#CD_EMPREGADO, NOME, @CD_DEPARTAMENTO)

DEPARTAMENTO(#CD_DEPARTAMENTO, NOME_DEPARTAMENTO)

Chave Primária
@ Chave Estrangeira

Restrições de Integridade Referencial

- Definição da Tabela Departamento

```
CREATE TABLE DEPARTAMENTO (
    CD_DEPARTAMENTO      NUMBER(9),
    NOME                 VARCHAR2(50),
    CONSTRAINT PK_DEPARTAMENTO PRIMARY KEY
                                 (CD_DEPARTAMENTO)
)
```

Restrições de Integridade Referencial

- Definição da Tabela Empregado

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO          NUMBER(9) PRIMARY KEY,
    NOME                  VARCHAR2(50),
    CD_DEPARTAMENTO       NUMBER(9),
    CONSTRAINT FK_EMPREGADO FOREIGN KEY
    (CD_DEPARTAMENTO) REFERENCES
    DEPARTAMENTO(CD_DEPARTAMENTO)
)
```

Restrições de Integridade Referencial

- Utilizando a cláusula REFERENCES

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO  NUMBER(9) PRIMARY KEY,
    NOME          VARCHAR2(50),
    CD_DEPARTAMENTO   NUMBER(9) REFERENCES
                                  DEPARTAMENTO
)
```

Restrições de Integridade Referencial

- Através do comando ALTER TABLE

```
ALTER TABLE EMPREGADO ADD CONSTRAINT
  FK_EMPREGADO
FOREIGN KEY (CD_DEPARTAMENTO)
REFERENCES DEPARTAMENTO (CD_DEPARTAMENTO)
```

Restrições de Domínio

- A linguagem SQL provê diversos mecanismos para restringir os valores possíveis para determinado atributo

- NULL e NOT NULL
- DEFAULT
- CHECK

NULL e NOT NULL

- As restrições NULL e NOT NULL são declaradas no comando CREATE TABLE e indicam, respectivamente, que o valor de um atributo pode ser NULO e que o valor de um atributo não pode ser NULO

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO  NUMBER(9) PRIMARY KEY,
    NOME          VARCHAR2(50) NOT NULL
)
```

DEFAULT

- A cláusula DEFAULT é utilizada para atribuir um valor padrão quando um valor para determinado atributo não for especificado

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO  NUMBER(9) PRIMARY KEY,
    NOME          VARCHAR2(50) NOT NULL,
    ESTADO        CHAR(2) DEFAULT('SE')
)
```

CHECK

- Uma restrição do tipo CHECK é utilizada para verificar os valores inseridos para uma coluna ou tupla

```
CREATE TABLE EMPREGADO (
    CD_EMPREGADO  NUMBER(9) PRIMARY KEY,
    NOME          VARCHAR2(50) NOT NULL,
    SEXO          CHAR(1)
                  CHECK(SEXO = 'M' OR
                        SEXO = 'F')
)
```

CHECK

- Podemos declarar uma restrição do tipo CHECK que faz referência a mais de uma coluna

```
CREATE TABLE EMPREGADO (
     CD_EMPREGADO          NUMBER(9) PRIMARY KEY,
     NOME                  VARCHAR2(50) NOT NULL,
     SEXO                  CHAR(1),
     ESTADO                CHAR(2)
)
ALTER TABLE EMPREGADO ADD CONSTRAINT CK_EMPREGADO
   CHECK ((SEXO = 'M' OR SEXO = 'F') AND (ESTADO =
   'SE' OR ESTADO = 'AL'))
```

Removendo Restrições

- Para remover uma restrição utilizamos o comando ALTER TABLE ... DROP CONSTRAINT

```
ALTER TABLE <NOME_DA_TABELA> DROP
  CONSTRAINT <NOME_DA_CONSTRAINT>
```

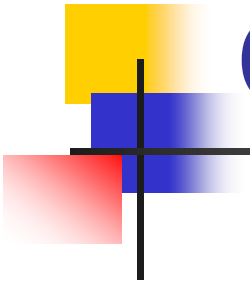
- Exemplo:

```
ALTER TABLE EMPREGADO DROP CONSTRAINT
  PK_EMPREGADO
```

OCA ORACLE DATABASE
SQL EXPERT EXAM

SQL Oracle:
Manipulação de Dados

andreviniciusnascimento@gmail.com
2012



Conteúdo

- Comandos SQL
- INSERT
- UPDATE
- DELETE
- Controle Transacional

Comandos SQL

- Todos os comandos SQL são categorizados em um dos seis tipos de comandos abaixo:
  - DDL
  - DML
  - TCL (TRANSACTION CONTROL)
  - SESSION CONTROL
  - SYSTEM CONTROL
  - EMBEDDED SQL (Comandos integrados a uma L3G)

Comandos SQL

- DDL (Data Definition Language)
  - CREATE
  - ALTER
  - DROP
  - RENAME
  - TRUNCATE
  - GRANT
  - REVOKE
  - FLASHBAK
  - PURGE
  - COMMENT

Comandos SQL

- DML (Data Manipulation Language)
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - MERGE

Comandos SQL

- TCL (Transaction Control Language)
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

INSERT, UPDATE e DELETE

- A parte de Manipulação de dados da linguagem SQL apresenta, além dos comandos para recuperação de dados, comandos para inserção, alteração e remoção de dados

- Esses comandos são conhecidos, respectivamente, como: INSERT, UPDATE e DELETE

INSERT, UPDATE e DELETE

- Para os exemplos que seguem, vamos considerar a tabela EMPREGADO definida como:

```
CREATE TABLE EMPREGADO (
  MATRICULA         CHAR(5),
  NOME              VARCHAR2(50),
  SEXO              CHAR(1),
  SALARIO           NUMBER(17,2),
  DEPARTAMENTO      CHAR(2),
  DT_NASC           DATE
 )
```

INSERT

- O comando INSERT é utilizado para inserir valores em uma relação

- A sintaxe básica para o comando INSERT é

```
INSERT INTO <TABLE_NAME> (col1, col2 ...)
VALUES (val1, val2 ...)
```

INSERT

- Quando o comando INSERT é submetido para execução, os seguintes passos são executados antes do comando retornar um resultado:
  - Confirmação e Validação da Tabela
  - Confirmação e Validação das Colunas
  - Avaliação das expressões na cláusula VALUES
  - Verificação de compatibilidade dos tipos de dados
  - Aplicação de restrições

INSERT

```
INSERT INTO EMPREGADO (MATRICULA, NOME,
  SEXO, SALARIO, DEPARTAMENTO)
VALUES (96345, 'DAVID AUGUSTO','M',
  1200.00, 'TI')

INSERT INTO EMPREGADO (MATRICULA, NOME)
  VALUES (97897, 'MARIA SILVA')
```

INSERT

- A lista de colunas pode ser omitida. Nesse caso, todos os valores devem ser fornecidos e na ordem em que foram definidos.

```
INSERT INTO EMPREGADO VALUES (96345,
  'DAVID AUGUSTO','M', 1200.00, 'TI')
```

INSERT

- Podemos utilizar o resultado de uma consulta como valores para o comando INSERT através da variação INSERT ... SELECT

```
INSERT INTO EMPREGADO_FILIAL_SUL
SELECT * FROM EMPREGADO
WHERE SEXO = 'F'

INSERT INTO EMPREGADO_FILIAL_NORTE
   (MATRICULA, NOME)
SELECT MATRICULA, NOME FROM EMPREGADO;
```

UPDATE

- O comando UPDATE é utilizado para modificar valores de uma tupla

- A sintaxe básica para o comando UPDATE é

```
UPDATE <TABLE_NAME>
SET <ATRIBUIÇÕES>
WHERE <CONDIÇÃO>
```

UPDATE

- Modificar o salário do EMPREGADO de matricula 94234 para 1500

```
UPDATE EMPREGADO
SET SALARIO = 1500
WHERE MATRICULA = 94234
```

UPDATE

- Modificar o salário do EMPREGADO de matricula 96765 para 1500 e o departamento para ‘FI’

```
UPDATE EMPREGADO
SET SALARIO = 1500, DEPARTAMENTO = ‘FI’
WHERE MATRICULA = 96765
```

DELETE

- O comando DELETE é utilizado para remover tuplas de uma relação

- A sintaxe básica para o comando DELETE é

```
DELETE FROM <TABLE_NAME>
WHERE <CONDIÇAO>
```

DELETE

- Remover todos os empregados do sexo masculino

```
DELETE FROM EMPREGADO
WHERE SEXO = 'M'

DELET EMPREGADO -- FROM OPCIONAL
WHERE SEXO = 'M'
```

Controle Transacional

- O Controle Transacional no SGBD Oracle é realizado através dos seguintes comandos:
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

Controle Transacional

- COMMIT
  - Salva as modificações no banco de dados desde que a sessão começou, ou desde o commit mais recente.
- ROLLBACK
  - Desfaz alterações realizadas no banco de dados desde o último commit.

Controle Transacional

- SAVEPOINT
  - Provê uma marca em uma sessão.
  - Permite que um comando ROLLBACK desfaça modificações até determinado ponto.

COMMIT

- O comando COMMIT é utilizado para salvar mudanças realizadas pelos comandos INSERT, UPDATE e DELETE.
- O comando COMMIT realiza mudanças permanentes no banco de dados. Após um COMMIT, as mudanças não podem ser desfeitas com um ROLLBACK.

COMMIT

- Existem dois tipos de evento COMMIT
  - COMMIT EXPLÍCITO – Ocorre quando um comando COMMIT é executado.
  - COMMIT IMPLÍCITO – Ocorre quando certos eventos do banco de dados ocorrem.

COMMIT EXPLÍCITO

- Ocorre quando um comando COMMIT é executado.

```
COMMIT;

COMMIT WORK; -- WORK é opcional.
                    -- Compatibilidade ANSI
```

COMMIT IMPLÍCITO

- Ocorre quando certos eventos acontecem no banco de dados.
  - Imediatamente antes e imediatamente depois de uma tentativa de execução de um comando DDL, como CREATE, ALTER, DROP, GRANT ou REVOKE. Mesmo que o comando DDL falhe (erro de execução), o COMMIT imediatamente antes é executado.
  - A saída normal de muitos utilitários Oracle, como o SQL*Plus.

ROLLBACK

- Desfaz as alterações realizadas no banco de dados desde o último commit.

```
COMMIT;
INSERT INTO TB_FUNCIONARIO (…
DELETE FROM…
UDATE TB_FUNCIONARIO…
ROLLBACK;
```

ROLLBACK

- Também pode ocorrer um ROLLBACK implícito.
  - Quando um programa termina de forma inesperada.

SAVEPOINT

- Representa uma marcação em uma transação que permite que o comando ROLLBACK desfaça modificações desde a marcação.

```
SAVEPOINT MARCA_01;

ROLLBACK WORK TO MARCA_O1;
```

SAVEPOINT

```
COMMIT;
UPDATE TB_FUNCIONARIO SET SALARIO = SALARIO* 1.1;
SAVEPOINT SP_1;
UPDATE TB_FUNCIONARIO SET SALARIO = SALARIO* 1.2;
ROLLBACK WORK TO SP_1;
COMMIT;
```

SAVEPOINT

- Regras para SAVEPOINTs
  - Todos os SAVEPOINTS devem incluir um nome.
  - Não se deve duplicar nomes de SAVEPOINTS em um mesma transação. Um novo SAVEPOINT com o mesmo nome de um SAVEPOINT anterior apaga o primeiro.
  - Na ocorrência de um COMMIT (explícito ou implícito), todos os SAVEPOINTS são apagados da memória. Qualquer referência posterior produz um código de erro.

SAVEPOINT

- ROLLBACK WORK TO SAVEPOINT
  - Se um comando ROLLBACK referenciar um SAVEPOINT que não existe, será produzido um erro informando que o comando ROLLBACK tentou referenciar um SAVEPOINT que não foi estabelecido.
  - Após o erro, o status do banco de dado continua inalterado (uncommitted state).