Gerenciamento de Dados e Informação

Fernando Fonseca
Ana Carolina
Robson Fidalgo

Centro de Informática
Universidade Federal de Pernambuco
CIn.ufpe.br

Introdução

- A tecnologia de BD tem evoluído para atender à crescente demanda de manipulação de aplicações e dados complexos
- SGBD convencionais (ex: relacional, de rede e hierárquico) são adequados para muitas aplicações comerciais
- Contudo, aplicações mais recentes têm requisitos e características não triviais que não são bem resolvidas pelos SGBD convencionais

Introdução

- Exemplos de limitações dos SGBD convencionais
  - Não oferecem suporte para implementar diretamente
    - Atributo composto
    - Atributo multivalorado
    - Especialização/Generalização
    - Tipos Complexos
    - Comportamento de objeto

Introdução

- Os SGBDOO surgiram para suprir estas limitações
  - The Object-Oriented Database System Manifesto (1989)
- Porém, os SGBDOO não foram bem aceitos pelo mercado* e pela academia**
  - * Grande esforço tecnológico e financeiro para migrar de SGBDR (dominante do mercado) para SGBDOO
  - ** Falta de padronização e base formal
    - Tentativa de padronização: ODMG

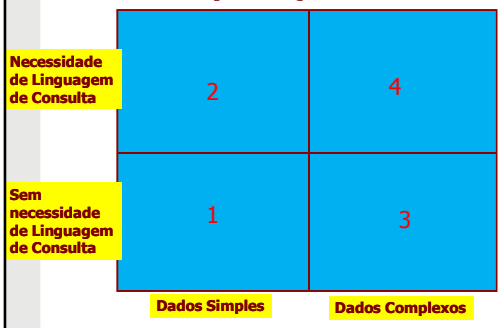
Introdução

- Para contornar a fraca aceitação dos SGBDOO surgiram os SGBDOR
  - Third Generation Database System Manifesto (1990)
  - SGBDOR → mantêm as vantagens do modelo relacional* e acrescentam características do modelo OO**
    - * Modelo eficiente
    - **Modelo rico
  - A tecnologia OR é uma camada de abstração construída sobre a tecnologia relacional
    - Permite incrementar o legado relacional com tecnologia OO

Introdução

- Sistemas de Banco de Dados Objeto-Relacionais podem ser vistos como uma tentativa de estender sistemas de banco de dados relacionais com a funcionalidade necessária para dar suporte a uma classe mais ampla de aplicações e, de certa forma, prover uma ponte entre os paradigmas relacional e orientado a objetos

Uma Classificação de Aplicações



7

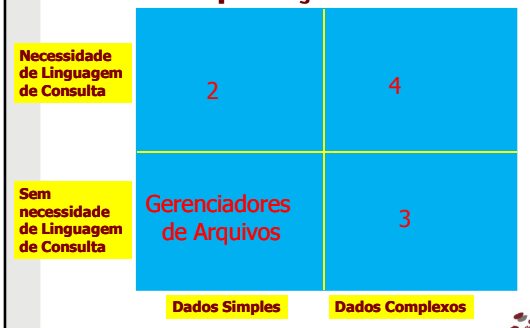
Uma Classificação de Aplicações

- Quadrante 1: Aplicações com dados simples, sem necessidade de linguagem de consulta
  - Operadores: get file
    put file
  - Bom desempenho
  - Exemplo: um editor de texto tradicional

Gerenciador de Arquivos (sistema operacional)

8

Uma Classificação de Aplicações



9

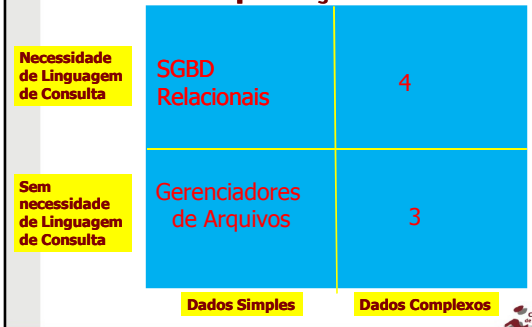
Uma Classificação de Aplicações

- Quadrante 2: Aplicações com dados simples e necessidade de linguagem de consulta
  - Linguagem de consulta
  - Ferramentas de interfaces
  - Desempenho (gerenciamento de transações consistente)
  - Segurança

SGBD relacionais

10

Uma Classificação de Aplicações



11

Uma Classificação de Aplicações

- Quadrante 3: Aplicações com dados complexos, sem necessidade de linguagem de consulta
  - Necessidade de rotinas específicas para manipulação dos dados complexos
  - Grande integração com uma linguagem de programação
  - Desempenho na atualização de variáveis persistentes

SGBD orientados a Objetos

12

Uma Classificação de Aplicações

	Dados Simples	Dados Complexos
Necessidade de Linguagem de Consulta	SGBD Relacionais	4
Sem necessidade de Linguagem de Consulta	Gerenciadores de Arquivos	SGBD Orientados a Objetos

13

Uma Classificação de Aplicações

- Quadrante 4: Aplicações com dados complexos e necessidade de linguagem de consulta
  - Linguagem de Consulta estendida a objetos complexos (SQL3)
  - Ferramentas de visualização não convencionais
  - Otimizador de consultas

SGBD Objeto-relacionais

OBJETO: objetos complexos **RELACIONAL: ling. consulta**

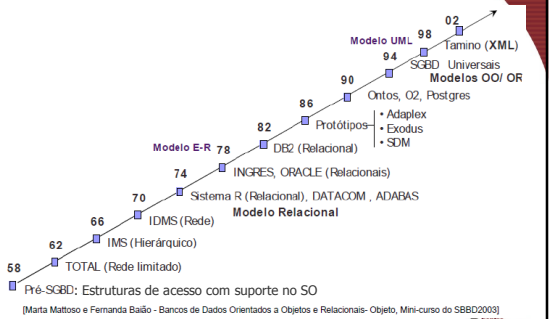
14

Uma Classificação de Aplicações

	Dados Simples	Dados Complexos
Necessidade de Linguagem de Consulta	SGBD Relacionais 100	SGBD Objeto-relacionais 150
Sem necessidade de Linguagem de Consulta	Gerenciadores de Arquivos	SGBD Orientados a Objetos 1

15

História dos SGBD



[Marta Mattoso e Fernanda Baião - Bancos de Dados Orientados a Objetos e Relacionais- Objeto, Mini-curso do SBBD2003]

16

Modelo e Linguagem OR

- O modelo de dados OR é uma extensão do modelo de dados relacional
  - A extensão permite que usuários estendam o BD a partir da criação de novos tipos e operações
- A linguagem OR é uma extensão de SQL
  - A linguagem SQL estendida oferece suporte para a definição de tipos de dados complexos e métodos, além da instanciação, manipulação e referência de objetos

17

Modelo e Linguagem OR

- SQL3 (também chamada de SQL:1999)
  - É a base para muitos SGBDO-R
    - EX: ORACLE, DB2, INFORMIX, PostgreSQL, ...
  - Oferece suporte para
    - Objetos complexos
    - Herança
    - Identidade de objetos (OID)
    - Referências a objetos
    - Definição de comportamento

18

Oracle OR – Visão Geral

- Conceitos básicos
  - Tipo de objetos
  - Métodos
  - Evolução de tipos
  - Herança de tipos
  - Tabela de objetos
  - Tabela de objetos com herança
  - Objetos de linha e objetos de coluna
  - Referência de objetos
  - Coleção de objetos

Oracle OR – Visão Geral

- Tipo de objetos
  - Tipo de objeto é um tipo abstrato de dados (TAD)
    - É um tipo de dado definido pelo usuário que encapsula propriedades (atributos) e comportamento (métodos)
    - Corresponde ao molde de um objeto
    - Não aloca espaço de armazenamento
      - Não pode armazenar dados

Oracle OR – Visão Geral

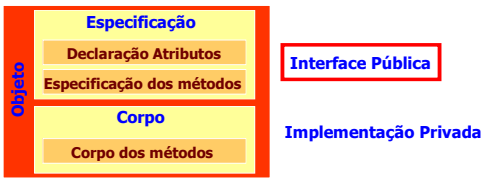
- Tipo de objetos (Cont.)
  - Permite capturar inter-relacionamento estrutural de objetos, estendendo a estrutura bidimensional relacional

CPF	NOME	ENDEREÇO			FONES
		DESCRIÇÃO	CIDADE	ESTADO	
444.444.444-44	Rita S. Lima	R. Sta. Ana, 10	Olinda	PE	2222-2222
					3333-3333
888.888.888-88	José R. Silva	Av. Recife, 20	Recife	PE	4444-4444
...	...	...	...	...	...

O exemplo acima pode ser feita em estrutura OR, mas não pode ser feita em estrutura Relacional

Oracle OR – Visão Geral

- Tipo de objetos (Cont.)
  - São especificados a partir de
    - Atributos → propriedades do objeto (opcional)
    - Métodos → procedimentos ou funções (opcional)



Oracle OR – Visão Geral

- Tipo de objetos (Cont.)
  - Exemplo de especificação da interface pública de um objeto

Sintaxe resumida:

```
CREATE [OR REPLACE] TYPE nome_tipo  AS OBJECT (
   [lista de atributos]
   [lista de métodos]
);
```


Oracle OR – Visão Geral

- Tipo de objetos (Cont.)
  - EX. :

Especificação

```
CREATE OR REPLACE TYPE tp_pessoa AS OBJECT (
  id            NUMBER,                          } Atributos
  pri_nome      VARCHAR2(20),
  ult_nome      VARCHAR2(25),
  email         VARCHAR2(25),
  fone          VARCHAR2(12),
MEMBER PROCEDURE exibir_detalhes ( SELF IN OUT tp_pessoa), MAP
MEMBER FUNCTION get_id RETURN NUMBER               } Métodos
) NOT FINAL;
```

← Para permitir criar um subtipo

Não é possível inserir dados em tp_pessoa. Pois, um tipo de objeto é um molde, não podendo armazenar dados.

Oracle OR – Visão Geral

- Tipo de objetos (Cont.)
  - EX :

```
CREATE OR REPLACE TYPE tp_ponto AS OBJECT(
   x NUMBER,
   y NUMBER,
   MEMBER FUNCTION getX RETURN NUMBER,
   MEMBER FUNCTION getY RETURN NUMBER,
   MEMBER PROCEDURE setX(newx NUMBER),
   MEMBER PROCEDURE setY(newy NUMBER),
   MEMBER PROCEDURE desenhar
) NOT FINAL NOT INSTANTIABLE;
```

para criar um tipo abstrato

Oracle OR – Visão Geral

- Tipo de objetos (Cont.)
  - Pode ser usado da mesma forma que é usado um tipo primitivo
    - EX:

Para definir o tipo de um atributo de uma tabela

```
CREATE TABLE tb_contatos (
  contato        tp_pessoa,
  dt_contato     DATE );

CREATE TABLE tb_domicilio (
  local          tp_ponto,
  endereco       VARCHAR2 (80) );
```

Para definir o tipo de um atributo de um TAD

```
CREATE TYPE tp_contatos AS OBJECT (
  contato        tp_ pessoa,
  dt_contato     DATE );

CREATE TYPE tp_domicilio AS OBJECT (
  local          tp_ponto,
  endereco       VARCHAR2 (80) );
```


Oracle OR – Visão Geral

- Métodos
  - São funções ou procedimentos que são declarados na definição de um tipo de objeto
  - Exigem o uso de **parênteses** (mesmo sem parâmetros)
    - O uso de ( ) é para diferenciar o método de um procedimento ou função comum
  - Podem ser
    - MEMBER
    - MAP ou ORDER
    - Construtor

Oracle OR – Visão Geral

- Um tipo de objeto sempre possui um construtor, pode possuir zero ou mais métodos membro e pode possuir um método map ou um método order, porém não os dois.

realizar comparações objeto-a-objeto

fornece a base para comparar objetos, mapeando as instâncias dos objetos em um dos tipos escalares DATE, NUMBER, VARCHAR2

possuem acesso aos dados da instância do objeto

Método que cria uma nova instância para o objeto, atribuindo valores aos seus atributos

Oracle OR – Visão Geral

- Métodos (Cont.)
  - Construtor
    - Criado implicitamente (pelo Oracle) ao criar um tipo de objeto ou explicitamente pelo programador
    - Deve ser exatamente igual ao nome do tipo
    - Pode haver mais de um construtor
  - EX:

```
INSERT INTO tb_contatos VALUES (
 tp_pessoa (65, 'Ruth', 'Araújo', 'ruth@gmail.com', '81-3333-3333'), '24 Jun 2006'
);
```

Invocando o método construtor padrão

Oracle OR – Visão Geral

- Métodos (Cont.)
  - MEMBER
    - São os métodos mais comuns
      - Implementam as operações das instâncias do tipo
    - São invocados através da qualificação de objeto
      - Objeto.método()
  - MAP ou ORDER
    - São funções para comparar objetos
      - São mutuamente exclusivos!
    - Métodos ORDER não podem ser definidos em subtipos e são menos eficientes do que métodos MAP

Oracle OR – Visão Geral

- Métodos (Cont.)
  - ORDER
    - Exige como parâmetro um **objeto** do mesmo tipo
    - Compara o objeto **corrente** (SELF) com o objeto do **parâmetro** (x)
    - Usa a **lógica interna** do objeto para efetuar a comparação entre dois objetos diferentes (mas do mesmo tipo), **devolvendo** um inteiro correspondente ao tipo de ordem

Inteiro retornado	Interpretação
Positivo	SELF > X
Negativo	SELF < X
Zero	SELF = X

Oracle OR – Visão Geral

- Métodos (Cont.)
  - MAP
    - Não exige parâmetro
    - Compara vários objetos (ex: ORDER BY)
    - Faz uma comparação de tipos padrão, usando os atributos do objeto como fatores da comparação
      - Permite comparar objetos mapeando suas instâncias em um dos tipos escalares (ex: DATE, NUMBER, VARCHAR2) ou tipo SQL (ex: CHARACTER ou REAL)

Oracle OR – Visão Geral

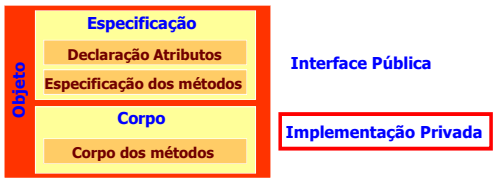
- Métodos
  - MAP (Cont.)
    - Retorna um dos atributos do objeto
    - É chamado implicitamente quando há comparação de objetos
      - DISTINCT, GROUP BY, UNION e ORDER BY
    - Só podem ser declarados em um subtipo se houver um método MAP declarado no supertipo

Oracle OR – Visão Geral

Caso não seja especificado um método MAP ou ORDER, o Oracle só pode concluir se dois objetos são iguais, nada podendo dizer sobre a ordem de relacionamento entre estes objetos

Oracle OR – Visão Geral

- Métodos (Cont.)



Oracle OR – Visão Geral

- Métodos (Cont.)
  - Implementação privada do corpo de um objeto

Sintaxe resumida:

```
CREATE [OR REPLACE] TYPE BODY nome_tipo AS
  [lista de subprogramas (procedimento, função ou construtor)]
  [lista de funções MAP ou ORDER]
END ;
```

Oracle OR – Visão Geral

- Métodos (Cont.)
  - EX: Corpo dos métodos para um tipo pessoa - MAP

```
CREATE OR REPLACE TYPE BODY tp_pessoa AS

MEMBER PROCEDURE exibir_detalhes ( SELF IN OUT tp_pessoa) IS
 BEGIN
   DBMS_OUTPUT.PUT_LINE(TO_CHAR(id) || ' ' || pri_nome || ' ' || ult_nome);
   DBMS_OUTPUT.PUT_LINE(email || ' '  || fone);
 END;

MAP MEMBER FUNCTION personTOInt RETURN INTEGER IS
 p INTEGER := id;
 BEGIN
   RETURN p;
 END;
END;
```


Oracle OR – Visão Geral

- Métodos (Cont.)
  - EX: Corpo dos métodos para um tipo apartamento - ORDER

```
CREATE OR REPLACE TYPE BODY tp_apartamento IS
ORDER MEMBER FUNCTION comparaOrdemApto(X IN tp_apartamento)
RETURN INTEGER IS
BEGIN
RETURN SELF.numero - X.numero;
END;
END;
```


Oracle OR – Visão Geral

- Métodos (Cont.)
  - EX:

```
CREATE OR REPLACE TYPE BODY tp_ponto AS
  MEMBER FUNCTION getX RETURN NUMBER IS
  BEGIN
    RETURN x;
  END;
  MEMBER FUNCTION getY RETURN NUMBER IS
      --Semelhante getX
  MEMBER PROCEDURE setX(newx NUMBER) IS
  BEGIN
    x := newx;
  END;
  MEMBER PROCEDURE setY(newy NUMBER) IS
      --Semelhante setX
  MEMBER PROCEDURE desenhar IS
  BEGIN
    NULL;
  END;
 MEMBER PROCEDURE setXY(newx NUMBER, newy NUMBER) AS
  BEGIN
   setX(newx); setY(newy);
  END;
END;
```

Método abstrato

Oracle OR – Visão Geral

- Evolução de tipos
  - A partir do uso de ALTER TYPE é possível
    - Adicionar e excluir atributos
    - Adicionar e excluir métodos
    - Modificar as propriedades de um atributo
      - EX: Tamanho, precisão e tipo
    - Modificar o status FINAL e INSTANTIABLE de um tipo
    - . . .

Oracle OR – Visão Geral

- Evolução de tipos (Cont.)
  - EX:

```
ALTER TYPE tp_pessoa
   ADD ATTRIBUTE ( sexo VARCHAR2(1) ) CASCADE ;
```

Vai propagar a mudança para todos os tipos dependentes

```
ALTER TYPE tp_pessoa
   DROP MAP MEMBER FUNCTION get_id RETURN NUMBER  INVALIDATE;
```

Vai invalidar todos os tipos dependentes

```
ALTER TYPE tp_pessoa NOT FINAL CASCADE;
```

Vai permitir especializar o tipo tp_pessoa

```
ALTER TYPE tp_pessoa
   MODIFY ATTRIBUTE sexo VARCHAR2(10) CASCADE ;
```

Modificando o tamanho do atributo

Oracle OR – Visão Geral

- Herança de tipos
  - Permite criar uma hierarquia de sub-tipos especializados
  - Os tipos derivados (sub-tipos) herdam os atributos e métodos dos tipos ancestrais (super-tipos)
  - Os sub-tipos podem acrescentar novos atributos ou métodos e/ou redefinir os métodos dos super-tipos

Oracle OR – Visão Geral

- Herança de tipos (Cont.)
  - EX:

```
CREATE TYPE tp_funcionario UNDER tp_pessoa(
   depto_id        NUMBER,
   funcao          VARCHAR2(30),
   salario         NUMBER
) NOT FINAL;
```

Para permitir definição de sub-tipos

Por default um tipo de objeto é FINAL!

Oracle OR – Visão Geral

- Herança de tipos (Cont.)
  - Os métodos também podem ser declarados como FINAL
    - Os subtipos não podem redefinir sua implementação
  - Diferentemente dos tipos de objetos, os métodos são definidos por padrão como NOT FINAL

```
CREATE OR REPLACE TYPE tp_exemplo AS object (
x int,
MEMBER PROCEDURE redefinivel,
FINAL MEMBER PROCEDURE naoRedefinivel
) NOT FINAL;
```


Oracle OR – Visão Geral

- Herança de tipos (Cont.)
  - EX: Um retângulo onde o ponto é a origem das coordenadas do mesmo

```
CREATE OR REPLACE TYPE tp_retangulo UNDER tp_ponto(
  largura NUMBER,
  altura NUMBER,
  CONSTRUCTOR FUNCTION tp_retangulo (x1 tp_ponto, z NUMBER)
          RETURN SELF AS RESULT,
  MEMBER PROCEDURE setLargura(newwidth NUMBER),
  MEMBER PROCEDURE setAltura(newheight NUMBER),
  MEMBER FUNCTION getLargura RETURN NUMBER,
  MEMBER FUNCTION getAltura RETURN NUMBER,
  OVERRIDING MEMBER PROCEDURE desenhar
);
```

Herda o ponto (x,y)

Retângulos com h=2 *l

Método desenhar deve ser redefinido

Oracle OR – Visão Geral

- Herança de tipos (Cont.)
  - EX: Corpo do tipo

```
CREATE OR REPLACE TYPE BODY tp_retangulo AS
  CONSTRUCTOR FUNCTION tp_retangulo (x1 tp_ponto, z number)
          RETURN SELF AS RESULT IS
  BEGIN
      x := x1.x; y := x1.y; largura := z;  altura := 2*z; RETURN;
  END;
  MEMBER PROCEDURE setLargura (newwidth NUMBER) IS
  BEGIN
     largura := newwidth;
  END;
```


Oracle OR – Visão Geral

- Herança de tipos
  - EX. (Cont):

```
MEMBER PROCEDURE setAltura(newheight NUMBER) IS
          BEGIN
    altura := newheight;
  END;
  MEMBER FUNCTION getLargura RETURN NUMBER IS
  BEGIN
     RETURN largura;
  END;
```


Oracle OR – Visão Geral

- Herança de tipos
  - EX (cont):

```
MEMBER FUNCTION getAltura RETURN NUMBER IS
  BEGIN
     RETURN altura;
  END;
 OVERRIDING MEMBER PROCEDURE desenhar IS
  BEGIN
     DBMS_OUTPUT.PUT_LINE('Desenhar retângulo:(' || SELF.getX() ||
     ',' || SELF.getY() || '), largura ' || getLargura() || ', altura ' ||
getAltura());
  END;
END;
```

Oracle OR – Visão Geral

- Herança (Cont.)
  - Exemplo de uso – Objeto retângulo
    - Criação de tabela

```
CREATE TABLE tb_retangulo of tp_retangulo;
```

    - Criação de objeto

```
INSERT INTO tb_retangulo VALUES
        (tp_retangulo(tp_ponto(3,2),5));
```


Oracle OR – Visão Geral

- Herança
  - Exemplo de uso – Objeto retângulo (Cont.)
    - Consulta objetos

```
select * from tb_retangulo;
```

X	Y	LARGURA	ALTURA
3	2	5	10

Ponto (x,y)

Oracle OR – Visão Geral

- Tabelas de objetos
  - São tabelas especiais onde cada linha armazena um objeto
    - Provê uma **visão relacional** desses objetos
  - As linhas de uma tabela de objetos possuem um OID (object identifier) implícito (definido pelo ORACLE)
  - Os objetos de uma tabela de objetos podem ser referenciados (REF) por outros objetos
  - Nos comandos de manipulação de objetos utilizar aliases para as tabelas
- Pode ser definido pelo programador (só recomendado para objetos interoperáveis entre diferentes BD)

Oracle OR – Visão Geral

- Tabelas de objetos (Cont.)
  - A tabela tb_pessoa pode ser definida com
    - Uma única coluna (tabela de objetos)
      - Cada linha sendo um objeto do tipo tp_pessoa

```
CREATE TABLE tb_pessoa of tp_pessoa
(id     PRIMARY KEY );
```

Fazer o mesmo para outras restrições
EX: UNIQUE, NOT NULL, FOREIGN KEY, CHECK

    - Múltiplas colunas
      - Uma coluna para cada atributo do tipo tp_pessoa
        - EX: Todos já vistos até agora

Oracle OR – Visão Geral

- Tabelas de objetos para herança
  - Não há estrutura de armazenamento associada com os tipos que pertencem a uma hierarquia
    - Deve-se criar tabelas de objetos para manipular as hierarquias dos tipos, formando uma hierarquia de tabelas

```
CREATE TABLE tb_retangulo OF tp_retangulo;
```


Oracle OR – Visão Geral

- Tabelas de objetos para herança (Cont.)
  - Descrição das tabelas especializadas

```
DESC tb_retangulo;

Nome            Nulo?    Tipo
--------------- -------- ------------
X                        NUMBER
Y                        NUMBER
LARGURA                  NUMBER
ALTURA                   NUMBER
```

Oracle OR – Visão Geral

- Objeto de linha e objeto de coluna
  - Além dos objetos armazenados em tabelas (Row Objects), pode haver objetos armazenados em colunas (Column Objects)
    - Column Objects: são objetos armazenados em **colunas** de tabelas relacionais ou como **atributos** de tipos objetos

```
CREATE TABLE tb_contatos (
 contato       tp_pessoa,
 dt_contato    DATE );
```

```
CREATE TYPE tp_contatos AS OBJECT (
 contato       tp_ pessoa,
 dt_contato    DATE );
```


ORACLE OR – Visão Geral

- OID (OBJECT IDENTIFIER)
  - Cada objeto possui um identificador único ou manipulador
  - É automaticamente atribuído quando um objeto é armazenado em uma object table
  - É armazenado em uma coluna oculta de 16 bytes do tipo RAW
  - Pode ser referenciado por colunas em outras tabelas, analogamente à chave estrangeira referenciando uma chave primária

Oracle OR – Visão Geral

- Referência de objetos
  - É um ponteiro lógico para um Row Object
    - Usado para fazer referência
    - É definido a partir do OID do objeto
    - Oferece acesso rápido/direto
    - Não garante integridade referencial

ORACLE OR - Visão Geral

- Referência de objetos (Cont.)
  - REF

    Referências para objetos são do tipo REF
    - Um atributo pode ser declarado como um REF (uma referência) para um tipo de objeto
    - Referências para objetos são úteis para identificar unicamente e localizar um objeto
    - Somente é possível obter referências para objetos que possuam OID, ou seja, só é possível referenciar objetos armazenados em object tables

ORACLE OR - Visão Geral

- Referência de objetos (Cont.)
  - REF em Colunas
    - Uma coluna de uma tabela (ou um atributo de um object type) pode ser declarado como sendo do tipo REF

```
...
<atributo> REF <tipo de objeto>;
...
```


ORACLE OR - Visão Geral

- Referência de objetos (Cont.)
  - REF como operador
    - Quando é necessário obter o identificador de um objeto de uma tabela, utiliza-se o operador REF(), tendo como argumento o aliás de uma object table

```
SELECT REF(P) FROM <tabela> P WHERE ...;
```

ORACLE OR - Visão Geral

- Referência de objetos (Cont.)
  - Uma coluna do tipo **REF** pode referenciar objetos do tipo indicado que estejam em qualquer tabela
  - Para restringir o escopo de referências para uma única tabela usar

SCOPE IS

61

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - EX:

```
CREATE OR REPLACE TYPE tp_cliente as OBJECT(
cod_cli          VARCHAR (3),
nm_cli           VARCHAR (60));

CREATE TABLE tb_cliente OF tp_cliente(
cod_cli          PRIMARY KEY,
nm_cli           NOT NULL);

CREATE OR REPLACE TYPE tp_dependente as OBJECT(
cod_dep          VARCHAR(3),
nm_dep           VARCHAR (60),
ref_titular      REF  tp_cliente);

CREATE TABLE tb_dependente OF tp_dependente(
cod_dep          PRIMARY KEY,
nm_dep           NOT NULL,
ref_titular      SCOPE IS tb_cliente);
```

62

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - EX:

```
INSERT INTO tb_cliente VALUES ('C1' , 'Rita');

INSERT INTO tb_cliente VALUES ('C2' , 'Ana');

INSERT INTO tb_dependente
SELECT 'D1', 'Paulo', REF (C)
FROM tb_cliente C WHERE cod_cli = 'C1';

INSERT INTO tb_dependente
SELECT 'D2', 'Pedro', REF (C)
FROM tb_cliente C WHERE cod_cli = 'C2';
```

63

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - EX:

Referência do objeto Cliente

```
SELECT *
FROM tb_dependente D;
```

```
COD_DEP NM_DEP    REF_TITULAR
-------------------------------------------
D1      Paulo     Q2459QW8RNDGS0D98G765SF
D2      Pedro     5XBGVX3B75XCN490VM0VBX4
```

Referência do próprio objeto

```
SELECT REF(D)
FROM tb_dependente D
WHERE D. nm_dep = 'Paulo';
```

```
REF(D)
------------------------
HRD23K56RNDGS0DUY6TGDE4
```

64

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - EX:

OID gerado pelo Oracle

```
SELECT nm_dep, ref_titular
FROM tb_dependente;
```

```
NM_DEP      REF_TITULAR
-------------------------------------
Paulo       Q2459QW8RNDGS0D98G765SF
Pedro       5XBGVX3B75XCN490VM0VBX4
```

```
SELECT D.ref_titular.cod_cli  as cod_cliente,
       D.ref_titular.nm_cli  as nm_cliente,
       D.nm_dep  as nm_dependente
FROM tb_dependente D;
```

Referência aos objetos tp_cliente

```
COD_CLIENTE  NM_CLIENTE  NM_DEPENDENTE
-------------------------------------
C1           Rita        Paulo
C2           Ana         Pedro
```

65

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Verificando a validade das referências (Dangling)
    - EX:

```
DELETE FROM tb_cliente WHERE cod_cli = 'C1';
```

Remove o objeto Rita

```
SELECT D.ref_titular.cod_cli  as cod_cliente,
       D.ref_titular.nm_cli  as nm_cliente,
       D.nm_dep  as nm_dependente
FROM tb_dependente D;
```

O objeto Rita não é listado, mas Paulo continua aparecendo

```
COD_CLIENTE  NM_CLIENTE  NM_DEPENDENTE
-------------------------------------
                         Paulo
C2           Ana         Pedro
```

66

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Verificando a validade das referências (Dangling)
    - EX:

```
SELECT D.ref_titular.cod_cli  as cod_cliente,
       D.ref_titular.nm_cli  as nm_cliente,
       D.nm_dep  as nm_dependente
FROM tb_dependente D
WHERE D.ref_titular IS DANGLING;
```

```
COD_CLIENTE  NM_CLIENTE  NM_DEPENDENTE
-----------------------------------
                         Paulo
```

Só aparecem os objetos sem referências válidas

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Verificando a validade das referências (Dangling)
    - EX:

```
SELECT D.ref_titular.cod_cli  as cod_cliente,
       D.ref_titular.nm_cli  as nm_cliente,
       D.nm_dep  as nm_dependente
FROM tb_dependente D
WHERE D.ref_titular IS NOT DANGLING;
```

```
COD_CLIENTE  NM_CLIENTE  NM_DEPENDENTE
-----------------------------------
C2           Ana         Pedro
```

Só aparecem os objetos com referências válidas

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Verificando a validade das referências (Dangling)
    - EX:

```
SELECT D.ref_titular.cod_cli as cod_cliente,
       D.ref_titular.nm_cli as nm_cliente,
       D.nm_dep as nm_dependente
FROM tb_dependente D
WHERE D.ref_titular IS NOT NULL;
```

DANGLING != NULL

```
COD_CLIENTE  NM_CLIENTE  NM_DEPENDENTE
-----------------------------------
                         Paulo
C2           Ana         Pedro
```


Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Garantindo a integridade referencial
    - Cláusula WITH ROWID
      - Importa o OID e a identificação física da linha onde o objeto está armazenado
      - Mantém o acesso direto ao objeto (Bom desempenho)
    - EX:

```
DROP TABLE tb_dependente;

CREATE TABLE tb_dependente OF tp_dependente(
cod_dep       PRIMARY KEY,
nm_dep        NOT NULL,
ref_titular   WITH ROWID REFERENCES tb_cliente);
```

Faz a REF para o objeto

Garante a integridade referencial

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Garantindo a integridade referencial
    - Ex. de erros:

```
INSERT INTO tb_dependente
SELECT 'D2', 'Pedro', REF (C)
FROM tb_cliente C WHERE cod_cli = 'C1';
```

Não deve permitir inserir, pois o cliente C1 foi excluído

```
DELETE FROM tb_cliente
WHERE cod_cli = 'C2'
```

Deve lançar um erro, pois o cliente C2 tem dependente

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Desreferenciando o REF (DEREF)
    - O operador DEREF "desfaz" o REF
      - Retorna um objeto referenciado por uma coluna do tipo REF
      - Desreferenciar um objeto dangling retorna um objeto null

Oracle OR – Visão Geral

- Referência de objetos (Cont.)
  - Desreferenciando o REF (DEREF)
    - EX:

```
SELECT DEREF(D.ref_titular)  as deref_titular,
       D.nm_dep  as nm_dependente
FROM tb_dependente D;
```

```
DEREF_TITULAR(COD_CLI, NM_CLI)        NM_DEPENDENTE
------------------------------------------------------
TP_CLIENTE('C1', 'Rita')              Paulo
TP_CLIENTE('C2', 'Ana')               Pedro
```


Oracle OR – Visão Geral

- Referência de Objetos (Cont.)
  - Desreferenciando o REF (DEREF)
    - EX. (sem usar DEREF):

```
SELECT D.ref_titular  as sem_deref,
       D.nm_dep as nm_dependente
FROM tb_dependente D;
```

```
SEM_DEREF                                NM_DEPENDENTE
------------------------------------------------------
P07XZC8V6Z0F97X6VZ965X6VZ4X8VXCVB6Z      Paulo
XCVU6CHBD967B436CB74B5X9B2BX2VQ4WFF      Pedro
```


Oracle OR – Visão Geral

- Referência de Objetos
  - Desreferenciando o REF (DEREF) (Cont.)
    - Acessando diretamente um atributo de um objeto referenciado

Ambos válidos

```
SELECT DEREF(D.ref_titular).nm_cli  as nm_cliente,
D.nm_dep  as nm_dependente
FROM tb_dependente D;
```

- Neste caso o DEREF não é necessário

```
SELECT D.ref_titular.nm_cli as nm_cliente,
D.nm_dep as nm_dependente
FROM tb_dependente D;
```

```
NM_CLIENTE                               NM_DEPENDENTE
------------------------------------------------------
Ana                                      Pedro
```


Oracle OR – Visão Geral

- Referência de Objetos (Cont.)
  - Operador VALUE
    - Exibe os dados das instâncias dos objetos
    - Usa o mesmo formato que DEREF
    - EX:

```
SELECT VALUE(D) value_titular
FROM tb_dependente D;
```

```
VALUE_TITULAR (COD_DEP, NM_DEP, REF_TITULAR)
------------------------------------------------------
TP_DEPENDENTE('D1', 'Paulo', Q2459QW8RNDGS0D98G765SF)
TP_DEPENDENTE('D2', 'Pedro', 5XBGVX3B75XCN490VM0VBX4)
```


Oracle OR – Visão Geral

- Coleções de objetos (Cont.)
  - Podem ser usadas para representar
    - Atributos multivalorados
    - Relacionamentos 1:n, n:1 ou n:m
    - São de dois tipos
      - VARRAY
      - NESTED TABLE

Oracle OR – Visão Geral

- Coleções de objetos (Cont.)
  - VARRAY X NESTED TABLE
    - Varray: coleção **ordenada** de uma quantidade **fixa** de elementos (índice inicia a partir de 1)
      - São armazenados como objetos contínuos
      - Um varray é armazenado "in line", ou seja, na mesma estrutura da tabela
    - Nested table: coleção **desordenada** de uma quantidade **arbitrária** de elementos
      - É uma tabela aninhada (tabela de uma tabela)
      - É armazenada "out line", ou seja, em uma outra estrutura (tabela)

Oracle OR – Visão Geral

- Coleções de objetos (Cont.)
  - VARRAY X NESTED TABLE
    - Varray: Indicada quando é necessário acessar elementos pelo índice ou manipular a coleção inteira como um valor
    - Nested table: indicada quando é necessário eficiência na execução de consultas sobre coleções

Oracle OR – Visão Geral

- Coleções de objetos (Cont.)
  - EX. (VARRAY):

```
CREATE OR REPLACE TYPE tp_fone AS OBJECT (
   cod_pais        VARCHAR2(2),
   cod_area        VARCHAR2(2),
   numero          VARCHAR2(8));

CREATE OR REPLACE TYPE tp_va_fone AS VARRAY(5) OF tp_fone;

CREATE TABLE tb_lista_fone_departamento(
  cod_depto NUMBER(5),
  lista_fone tp_va_fone);

INSERT INTO tb_lista_fone_departamento VALUES (100,
 tp_va_fone (tp_fone('55', '81', '22222222'),
             tp_fone('55', '81', '33333333'),
             tp_fone('55', '81', '44444444')));
```


Oracle OR – Visão Geral

- Coleções de objetos (Cont.)
  - EX. (VARRAY):

```
SELECT * FROM
tb_lista_fone_departamento;
```

```
COD_DEPTO   LISTA_FONE(COD_PAIS, COD_AREA, NUMERO)
---------------------------------------------------
100         TP_FONE_VARRAY(
            TP_FONE('55','81','22222222'),
            TP_FONE('55','81','33333333'),
            TP_FONE('55','81','44444444'))
```


Oracle OR – Visão Geral

- Coleções de objetos (Cont.)
  - EX. (NESTED TABLE):

```
CREATE TYPE tp_nt_fone AS TABLE OF tp_fone;

CREATE TABLE tb_lista_fone_contato(
  cod_contato NUMBER(5),
  lista_fone tp_nt_fone)
NESTED TABLE lista_fone STORE AS tb_lista_fone;

INSERT INTO tb_lista_fone_contato VALUES (001,
 tp_nt_fone (tp_fone('55', '81', '55555555'),
             tp_fone('55', '81', '66666666')));
```


Oracle OR – Visão Geral

- Coleções de objetos (Cont.)
  - EX. (NESTED TABLE) – ligeiramente modificado:

```
CREATE OR REPLACE TYPE tp_nt_fone AS TABLE OF tp_fone;

CREATE OR REPLACE TYPE tp_lista_fone_contato AS OBJECT(
  cod_contato NUMBER(5),
  lista_fone tp_nt_fone);

CREATE TABLE tb_lista_fone_contato OF tp_lista_fone_contato
NESTED TABLE lista_fone STORE AS tb_lista_fone;

INSERT INTO tb_lista_fone_contato VALUES (001,
 tp_nt_fone (tp_fone('55', '81', '55555555'),
             tp_fone('55', '81', '66666666')));
```


Oracle OR – Visão Geral

- Coleções de Objetos
  - EX. (NESTED TABLE):

```
SELECT * FROM tb_lista_fone_contato;
```

```
COD_CONTATO    LISTA_FONE(COD_PAIS, COD_AREA, NUMERO)
-----------------------------------------------------
001            TP_FONE_NESTED(TP_FONE('55', '81', '55555555'),
                              TP_FONE('55', '81', '66666666'))
```

Oracle OR – Visão Geral

- Coleções de Objetos
  - EX. (NESTED TABLE):

```
SELECT *
FROM TABLE(SELECT C.lista_fone
           FROM tb_lista_fone_contato C
           WHERE C.cod_contato = 001);
```

```
CO CO NUMERO
-- -- --------
55 81 55555555
55 81 66666666
```

A função TABLE também pode ser usada para consultar um NESTED TABLE/VARRAY

```
SELECT C.cod_depto, T.*
FROM tb_lista_fone_departamento C, TABLE(C.lista_fone) T;
```

```
COD_DEPTO CO CO NUMERO
----------------------
      100 55 81 22222222
      100 55 81 33333333
      100 55 81 44444444
```


Oracle OR – Visão Geral

- Coleções de Objetos
  - EX. (NESTED TABLE):

Para excluir uma NESTED TABLE atribui-se NULL

```
UPDATE tb_lista_fone_contato C
SET C.lista_fone  = NULL
WHERE cod_contato = 001 ;
```

Para inserir novamente valores na NESTED TABLE, esta tem que ser recriada.

```
UPDATE tb_lista_fone_contato C
SET C.lista_fone  = tp_nt_fone (
                    tp_fone('55', '81', '55555555'),
                    tp_fone('55', '81', '66666666'))
WHERE cod_contato = 001 ;
```


Oracle OR – Visão Geral

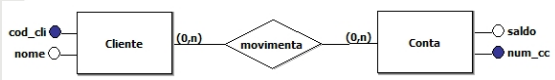
- Coleções de Objetos
  - EX. (NESTED TABLE):

Para atualizar apenas alguns valores da NESTED TABLE

```
UPDATE TABLE (
SELECT lista_fone
FROM tb_lista_fone_contato
WHERE cod_contato = 001) F
SET F.cod_pais = '66'
WHERE F.numero = '66666666';
```


Oracle OR – Visão Geral

- Coleções de Objetos
  - EX. (NESTED TABLE DE REFERÊNCIAS):



Oracle OR – Visão Geral

- Coleções de Objetos
  - EX. (NESTED TABLE DE REFERÊNCIAS):

```
CREATE OR REPLACE TYPE tp_conta;  -- Tipo incompleto

CREATE OR REPLACE TYPE tp_nt_ref_conta AS TABLE OF REF tp_conta;

CREATE OR REPLACE TYPE tp_cliente AS OBJECT (
   cod_cli          NUMBER(4),
   nm_cli           VARCHAR2(60),
   lista_conta      tp_nt_ref_conta);

CREATE OR REPLACE TYPE tp_nt_ref_cliente AS TABLE OF REF tp_cliente;

CREATE OR REPLACE TYPE tp_conta AS OBJECT(
   num_cc           NUMBER(4),
   saldo            NUMBER(8,2),
   lista_cliente    tp_nt_ref_cliente);
```


Oracle OR – Visão Geral

- Coleções de Objetos
  - EX. (NESTED TABLE DE REFERÊNCIAS):

```
CREATE TABLE tb_cliente OF tp_cliente
 (cod_cli PRIMARY KEY)
 NESTED TABLE lista_conta STORE AS tb_lista_conta;

CREATE TABLE tb_conta OF tp_conta
(num_cc PRIMARY KEY)
NESTED TABLE lista_cliente STORE AS tb_lista_cliente;
```

Oracle OR – Visão Geral

- Coleções de Objetos
  - EX. (NESTED TABLE DE REFERÊNCIAS):

```
INSERT INTO tb_cliente VALUES
( 0001, 'Jorge', tp_nt_ref_conta() );

INSERT INTO tb_cliente VALUES
( 0002, 'Rosa', tp_nt_ref_conta() );

INSERT INTO tb_conta VALUES
( 1111, 1000.00, tp_nt_ref_cliente(
    (SELECT REF(C) FROM tb_cliente C WHERE C.cod_cli = 0001),
    (SELECT REF(C) FROM tb_cliente C WHERE C.cod_cli = 0002)));
```

Centro de Informática
91