



Oracle Objeto-Relacional

Pablo Vieira Florentino



Motivação -

Modelo Objeto-Relacional

- Resposta dos Bancos de Dados Relacionais à Orientação a Objetos
- Relacional – Suporte a SQL, transações, etc.
- Objeto – Suporte a tipos complexos de dados, elementos Multi-Valorados, Herança



Elementos do Modelo Objeto-Relacional

- Tipo Objeto (Object Type)
- Objetos
- Tabelas de Objetos (Object Tables)
- Visões de Objetos (Object Views)
- Métodos
- Herança de Tipos
- Tipo REF
- Coleções



Tipos de Objetos

- Três componentes:
 - Nome
 - Atributos
 - Métodos
- CREATE TYPE

```
CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONE VARCHAR2(20)
);
```



Objetos

- São instâncias de tipos de objetos
- Possui os atributos e métodos de seu tipo de objeto
- Pode-se atribuir valores a seus atributos e invocar seus métodos
- Tratamento de dados e comportamento dos objetos, através da utilização de métodos específicos a cada classe



Tabelas de Objetos p/ armazenamento

- Tabelas especiais onde cada linha representa um objeto

```
CREATE TYPE T_PESSOA AS OBJECT  
(  
    NOME VARCHAR2(30),  
    TELEFONE VARCHAR2(20)  
);
```

```
CREATE TABLE TAB_PESSOA OF T_PESSOA;
```



Tabelas de Objetos

- Pode-se ver uma tabela de objetos de duas formas:
 - Uma tabela de uma única coluna contendo objetos do tipo definido onde pode-se efetuar operações características do modelo OO
 - Uma tabela com cada coluna representando um atributo do tipo definido, onde pode-se efetuar operações relacionais



Tabelas de Objetos (ex.)

```
INSERT INTO TAB_PESSOA VALUES  
(  
    "John Smith",  
    "1-800-555-1212"  
);
```

```
SELECT VALUE(P) FROM TAB_PESSOA P  
WHERE P.NOME = "John Smith";
```




Métodos

- Funções ou Procedimentos que são chamadas pela aplicação para modelar o comportamento dos objetos
- São armazenados no banco de dados através de PL/SQL ou Java
- Possui as seguintes categorias
 - Membro
 - Estático
 - Construtor
 - Comparação
 - Mapeamento
 - Ordenação



Métodos Construtores

- Todo tipo de objeto tem um método construtor definido pelo sistema que cria o objeto e instancia os atributos



Métodos Construtores (ex.)

```
P = T_PESSOA('VICTOR', '9185-9389', '07/04/1973')
```



Métodos Membros

- É a forma como as aplicações podem ganhar acesso aos dados das instâncias dos objetos.
- Possui sempre um parâmetro implícito SELF, logo trabalha com os atributos de um objeto específico
- É chamado da seguinte forma:
 - OBJETO.METODO()



Métodos Membros (ex.)

```
CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONE VARCHAR2(20),
  MEMBER FUNCTION GET_NOME RETURN VARCHAR,
  ...
);
```

```
CREATE TYPE BODY T_PESSOA AS
MEMBER FUNCTION GET_NOME RETURN VARCHAR IS
BEGIN
  RETURN SELF.NOME;
END GET_NOME;

...
END;
```



Métodos Estáticos

- Trabalham com dados globais do tipo do objeto (uma coleção de instâncias da classe) e não com o objeto específico
- Não possuem o parâmetro SELF
- É chamado da seguinte forma:
 - TIPO.METODO()



Métodos Estáticos (ex.)

```
CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONE VARCHAR2(20),
  DATA_NASCIMENTO DATE,
  MEMBER FUNCTION GET_NOME RETURN VARCHAR,
  STATIC FUNCTION PESSOA_MAIS_VELHA RETURN T_PESSOA,
  ...
);
```



Métodos de Comparação

- Para comparar dois objetos de tipos criados pelo usuário, o mesmo deve criar uma ordenação para o tipo usando métodos de mapeamento (map methods) ou métodos de ordenação (order methods)



Métodos de Mapeamento

- Produzem um único valor de um tipo pre-definido (DATE, NUMBER, VARCHAR) para ser utilizado como comparação



Métodos de Mapeamento

```
CREATE TYPE Rectangle_typ AS OBJECT (  
    length NUMBER,  
    width NUMBER,  
    MAP MEMBER FUNCTION area RETURN NUMBER,  
    ...  
);
```

```
CREATE TYPE BODY Rectangle_typ AS  
    MAP MEMBER FUNCTION area RETURN NUMBER IS  
    BEGIN  
        RETURN len * wid;  
    END area;
```

Comparação:

obj_1.area() > obj_2.area()



Métodos de Ordenação

- São mais gerais que os métodos de mapeamento
- É uma função com um parâmetro declarado para outro objeto do mesmo tipo e retorna:
 - <0 , caso o objeto SELF seja menor que o parâmetro
 - 0 , caso sejam iguais
 - >0 , caso o objeto SELF seja maior que o parâmetro



Métodos de Ordenação (ex.)

```
CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONE VARCHAR2(20),
  DATA_NASCIMENTO DATE,
  MEMBER FUNCTION GET_NOME RETURN VARCHAR,
  ORDER FUNCTION MATCH( P T_PESSOA ) RETURN INTEGER
  ...
);
```



Métodos de Ordenação (ex.)

```
CREATE TYPE BODY T_PESSOA AS
ORDER MEMBER FUNCTION MATCH (P T_PESSOA) RETURN INTEGER IS
BEGIN
    IF SELF.NOME < P.NOME THEN
        RETURN -1;
    ELSIF SELF.NOME > P.NOME THEN
        RETURN 1;
    ELSEIF SELF.DATA_NASCIMENTO < P.DATA_NASCIMENTO
        RETURN -1;
    ELSEIF SELF.DATA_NASCIMENTO > P.DATA_NASCIMENTO
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
...
END;
```



Tipo REF

- É um ponteiro lógico para um objeto
- Tipos REF e coleções de REFs modelam associações entre os objetos evitando o uso de chaves estrangeiras
- Provêm um fácil e intuitivo mecanismo de navegação entre os objetos via notação de `.`
- O Oracle executa as junções (internamente) quando necessárias e as evita caso contrário



REFs com Escopo (scoped REFs)

- Quando estiver declarando um atributo de um tipo de objeto ou uma coleção de objetos pode-se restringir para que contenham apenas referências de uma tabela de objetos específica. Este tipo REF é chamado Scoped REF
- Scoped REFs necessitam de menos espaço de armazenamento, e acesso mais eficiente que REFs sem escopo (Unscoped REFs)
- Um REF pode possuir escopo para uma tabela de objetos, para um tipo de objeto declarado, ou para qualquer subtipo do tipo declarado



REFs com Escopo (ex.) (scoped REFs)

```
CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONE VARCHAR2(20),
  DATA_NASCIMENTO DATE,
  PAI REF T_PESSOA SCOPE IS TAB_PESSOA,
  MEMBER FUNCTION GET_NOME RETURN VARCHAR,
  ORDER FUNCTION MATCH( P T_PESSOA ) RETURN INTEGER
  ...
);
```




Dangling REFs

- É possível que um REF para um objeto torne-se inválido por uma **remoção** do objeto ou mudança de **privilégios**. Este REF é chamado Dangling REF
- O Oracle possui um predicado IS DANGLING que testa se um REF encontra-se nestas condições



Desreferenciando REFs

- Acessar o objeto referenciado por um REF significa ***desreferenciar*** um REF
- O Oracle provê o operador Deref para desreferenciar um REF
 - (Desreferenciar um Dangling REF retorna um ponteiro NULL)



Visão de Objetos

- Forma de se acessar dados relacionais de uma forma orientada a objetos
- Pode-se possuir uma aplicação orientada a objetos sem se modificar o formato de armazenamento dos dados de forma relacional



Criação de Visão de Objetos

- Definir um tipo de objeto cujos atributos correspondam às colunas na tabela relacional
- Escrever uma consulta para a extração dos dados da tabela relacional. Especificar as colunas na mesma ordem dos atributos do tipo do objeto
- Especificar um valor único para servir como identificador do objeto



Criação de Visão de Objetos (ex.)

```
CREATE TABLE EMP (  
    ID NUMBER (5),  
    NOME VARCHAR2 (20),  
    SALARIO NUMBER (9, 2),  
);
```

```
CREATE TYPE T_EMP (  
    ID NUMBER (5),  
    NOME VARCHAR2 (20),  
    SALARIO NUMBER (9, 2),  
);
```

```
CREATE VIEW V_EMP OF T_EMP WITH OBJECT IDENTIFIER (ID)  
AS  
SELECT E.ID, E.NOME, E.SALARIO  
FROM EMP E  
WHERE SALARIO > 2000;
```



Coleções

- Para a modelagem dos alguns relacionamentos um-para-muitos (1,N) o Oracle suporta dois tipos de dados de coleções:
 - VARRAYs
 - Nested Tables (Tabelas Aninhadas)
- Estes tipos de coleção podem ser utilizados em qualquer lugar onde os outros tipos podem ser utilizados
- Equivalentes aos sets, bags e lists



VARRAYS

- Possuem um número máximo de elementos, embora este limite possa ser modificado em tempo de execução
- Os elementos possuem uma ordenação
- VARRAYS são armazenados como BLOBs



VARRAYS (ex.)

```
CREATE TYPE T_TELEFONES AS VARRAY(3) OF VARCHAR2(20);

CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONES T_TELEFONES,
  DATA_NASCIMENTO DATE,
  PAI REF T_PESSOA SCOPE IS TAB_PESSOA,
  MEMBER FUNCTION GET_NOME RETURN VARCHAR,
  ORDER FUNCTION MATCH( P T_PESSOA ) RETURN INTEGER
  ...
);
```



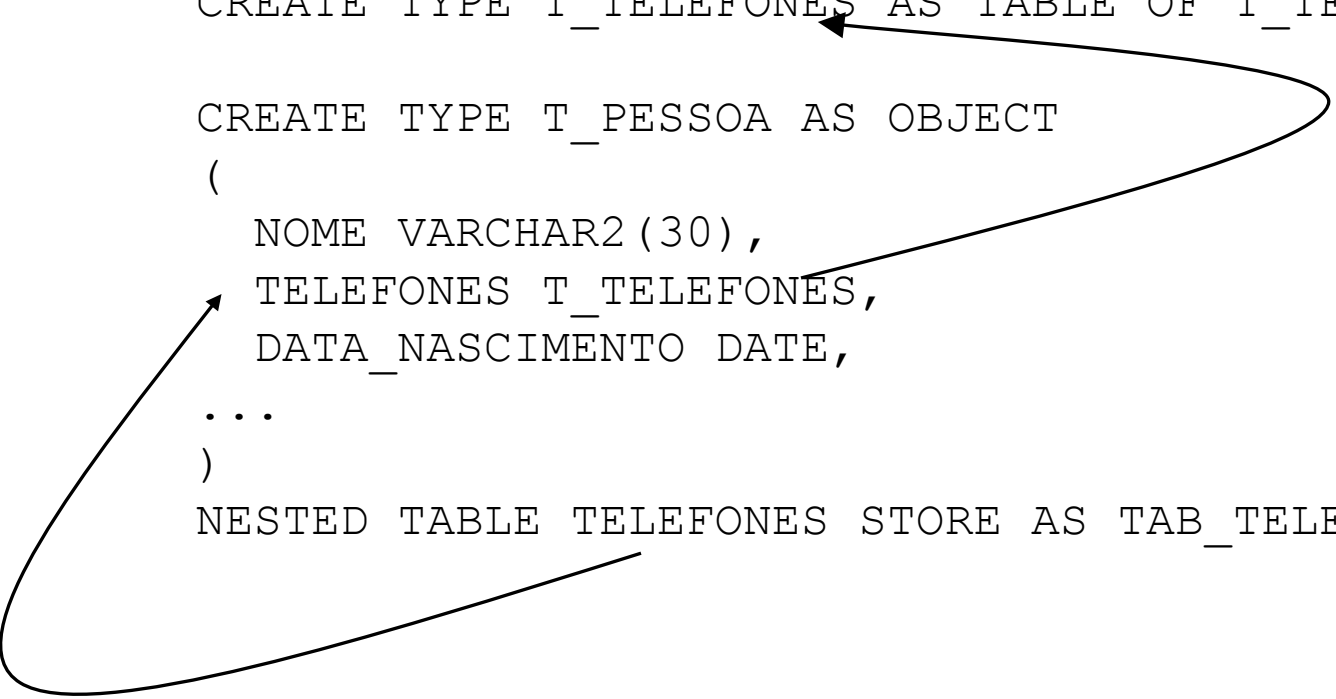

Nested Tables

- Podem conter qualquer número de elementos e pode-se efetuar select, insert e delete assim como em tabelas regulares
- Não existe uma ordenação para os elementos
- São armazenados em uma tabela de armazenamento onde cada elemento é mapeado para uma linha nesta tabela



Nested Tables (ex.)

```
CREATE TYPE T_TELEFONES AS TABLE OF T_TELEFONE;  
  
CREATE TYPE T_PESSOA AS OBJECT  
(  
    NOME VARCHAR2(30),  
    TELEFONES T_TELEFONES,  
    DATA_NASCIMENTO DATE,  
    ...  
)  
NESTED TABLE TELEFONES STORE AS TAB_TELEFONES;
```





Consultas em Coleções

```
SELECT P.NOME, P.TELEFONES  
FROM TAB_PESSOA P;
```

```
NOME          TELEFONES  
-----  
'VICTOR'     T_TELEFONES('9185-9389', '2569-5107')
```



Consultas em Coleções

```
SELECT P.NOME, TEL.*  
FROM TAB_PESSOA P, TABLE(P.TELEFONES) TEL;
```

NOME	TELEFONE
-----	-----
'VICTOR'	'9185-9389'
'VICTOR'	'2569-5107'



Herança

- O Oracle implementa herança simples, ou seja, um subtipo pode ter apenas um supertipo
- Pode se especializar os atributos e métodos de um supertipo da seguinte maneira:
 - Adicionar novos atributos
 - Adicionar novos métodos
 - Modificar a implementação de alguns métodos



Tipos FINAL e NOT FINAL

- Para permitir que um tipo possa possuir subtipos este deve ser definido como NOT FINAL. Por default um tipo de objeto é FINAL.

```
CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONES T_TELEFONES,
  DATA_NASCIMENTO DATE,
  ...
) NOT FINAL;
```



Métodos FINAL e NOT FINAL

- Para garantir que um método não seja sobrescrito nos subtipos este deve ser declarado como FINAL. Ao contrário de tipos de objetos, por default, um método é NOT FINAL.

```
CREATE TYPE T_PESSOA AS OBJECT
(
  NOME VARCHAR2(30),
  TELEFONES T_TELEFONES,
  FINAL MEMBER FUNCTION GET_NOME RETURN VARCHAR,
  ...
) NOT FINAL;
```



Criando Subtipos

```
CREATE TYPE T_ALUNO UNDER T_PESSOA
(
    DRE VARCHAR2(15),
    ...
);
```




Tipos de Objetos Abstratos

- Não há construtor
- Não se pode instanciar estes objetos

```
CREATE TYPE T_PESSOA AS OBJECT (...)  
NOT INSTANTIABLE NOT FINAL;
```

```
CREATE TYPE T_ALUNO UNDER T_PESSOA (...);
```



Tipos de Objetos Abstratos

- Um método também pode ser declarado **NON INSTANTIABLE** para criar um método em um tipo de objeto sem implementação (esta irá se encontrar nos subtipos – Uma INTERFACE)
- Somente em tipos de objetos **NON INSTANTIABLE**



Overload de métodos

```
CREATE TYPE MyType_typ AS OBJECT (  
    ...,  
    MEMBER PROCEDURE foo(x NUMBER), ...  
) NOT FINAL;
```

```
CREATE TYPE MySubType_typ UNDER MyType_typ (...,  
    MEMBER PROCEDURE foo(x DATE),  
    STATIC FUNCTION bar(...)) ...  
    ...  
);
```



Mapeamento

- Modelo OO x Modelo OR
- Modelo OO x Modelo Relacional



Referências

- Trabalho Oracle
- Mais consultas
- REFERÊNCIAS
 - **Oracle 9i**, Application Developer's Guide - Object-Relational Features, Release 2 (9.2), March 2002
 - Stonebraker & Moore, Object-Relational DBMSs: The Next Great Wave, 1995
 - Ambler, S., Mapping objects to relational databases

