SISTEMAS DE BANCO DE DADOS OBJETO-RELACIONAIS

Fernando da Fonseca de Souza
Ana Carolina Salgado

Agenda

- Introdução
- Características fundamentais de um SGBD O-R
  - **Extensão dos tipos de dados**
  - **Objetos Complexos**
  - **Herança**
  - **Regra**
- Principais características de projeto de BD objeto-relacional

Agenda

- Arquiteturas
- O SGBD Oracle 8i
  - **Tipos de Objetos**
  - **Tabelas de Objetos**
  - **Referências a Objetos**
  - **Coleções**
- Estudo de Caso

Introdução

- Em 1989, Manifesto de Orientação a Objetos
- Em 1990, Contra-Manifesto pró Relacional
- Na década de 90:
  - SGBD relacionais
  - SGBD relacionais estendidos
  - SGBD orientados a objetos

SGBD Objeto-Relacional

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

	Dados Simples	Dados Complexos
Com Consulta	2	4
Sem Consulta	1	3

Uma Classificação de Aplicações

1. Dados Simples sem Consultas

- Operadores: get file
  put file
- Boa performance
- Exemplo: um editor de texto

↓

Gerenciador de Arquivos (sistema operacional)

Uma Classificação de Aplicações

	Dados Simples	Dados Complexos
Com Consulta	2	4
Sem Consulta	Gerenciadores de Arquivos	3

Uma Classificação de Aplicações

2. Dados Simples com Consulta

- Linguagem de consulta
- Ferramentas de interfaces
- Performance (gerenciamento de transações consistente)
- Segurança

↓

SGBD relacionais

Uma Classificação de Aplicações

	Dados Simples	Dados Complexos
Com Consulta	SGBD Relacionais	4
Sem Consulta	Gerenciadores de Arquivos	3

Uma Classificação de Aplicações

3. Dados Complexos sem Consulta

- Necessidade de rotinas específicas para manipulação dos dados complexos
- Grande integração com uma linguagem de programação
- Performance na atualização de variáveis persistentes

↓

SGBD orientados a Objetos

Uma Classificação de Aplicações

	Dados Simples	Dados Complexos
Com Consulta	SGBD Relacionais	4
Sem Consulta	Gerenciadores de Arquivos	SGBD Orientados a Objetos

Uma Classificação de Aplicações

4. Dados Complexos com 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**

Uma Classificação de Aplicações

Com Consulta	SGBD Relacionais **100**	SGBD Objeto-relacionais **150**
Sem Consulta	Gerenciadores de Arquivos	SGBD Orientados a Objetos **1**
	Dados Simples	**Dados Complexos**

Por que Objeto-Relacional é a solução?

"***A Próxima Grande Onda***"

Duas forças do mercado:

1. Automação de novas aplicações multimídia, especialmente utilizando a Web (WWW):
   - **Consultas ad-hoc**
   - **Manipulação de objetos multimídia**
2. Aplicações comerciais vão migrar para o quadrante da direita:
   - **Sistemas de suporte à decisão com objetos complexos**
   - **Diminuição do custo do hardware**

Características Fundamentais de um SGBD O-R

- Extensão dos tipos de dados básicos
- Objetos Complexos
- Herança
- Regras

Extensão dos tipos de dados básicos

POR QUE?
- Tipos básicos limitados (inteiros, reais, etc.)
- Necessidade de operadores específicos sobre os tipos (básicos ou não)

Exemplo:

Posição geográfica da casa de um empregado, determinada por coordenadas (latitude, longitude)

Aplicação: Transporte solidário

Extensão dos tipos de dados básicos

SOLUÇÕES:

1. Modificar o sistema existente para incluir o que se necessita:
   Muito trabalho para o programador
   Performance inaceitável da aplicação

Exemplo:

```
alter table emp add column lat float;
alter table emp add column long float;
select r.nome from emp j, emp r
  where j.nome='José' and
  (j.long - r.long) ** 2 + (j.lat - r.lat) ** 2 < 1
```

Extensão dos tipos de dados básicos

2. Permitir a criação de novos tipos e respectivos operadores que serão incluídos no código do sistema

Exemplo:

```
alter table emp add column location point;
select r.nome from emp j, emp r
   where j.nome = 'José' and
    distance (j.location, r.location) < 1;
```


Extensão dos tipos de dados básicos

No SGBD Illustra:

- Cláusula de definição de tipo:

```
create type tipo1 (
    internallenght = 8,
    input = tipo1input,
    output = tipo1output);
create table tabela1 (
    name = varchar(30),
    atributo2 = tipo1);
```


Extensão dos tipos de dados básicos

3. Permitir a criação de funções tanto em SQL como em alguma linguagem de programação de uso geral, como C por exemplo, que serão armazenadas no sistema

No SGBD Illustra:

- Cláusula de definição de funções:

```
create function f-nome (t-nome1, ..., t-nomek)
    returns t-nome as (nome-arquivo ou
      expressão-SQL );
```


Extensão dos tipos de dados básicos

Exemplo:

```
create function salario (float,int)
    returns float as select ($1 * $2);
/*$1 (argumento): valor da hora do emp. */
/*$2 (argumento): no. de horas trabalhadas no
mês */
select nome from emp
    where salario (valor-hora,numero-horas)
      > 500,00;
```


Extensão dos tipos de dados básicos

Implementação de uma função na linguagem C:

Rendimento mensal de ações durante cinco anos

```
float rendimento (data-inicio) {
    compute resultado = current-date - data-
      inicio;
    convert resultado to months;
    if resultado < 12 return 0;
    else return resultado/60;  }
```


Extensão dos tipos de dados básicos

Se a função for compilada com sucesso e seu código executável armazenado em *invest,* ela deve ser registrada da seguinte forma:

```
create function rendimento (date)
    returns float as external name 'invest'
      language C;
```

Extensão dos tipos de dados básicos

SQL O-R:

- Cláusula de definição de operadores:

```
create operator
    binding opr-nome
    to function-nome;
```

- Cláusula de definição de função agregada:

```
create aggregate Mediana (integer)
    returns (integer) as Função-mediana;
```


Extensão dos tipos de dados básicos

Extensão de tipos completamente O-R:

1. **Ligação dinâmica**: as funções definidas pelo usuário são incluídas ao código do SGBD à medida que elas são necessárias (sem necessidade de "tirar o sistema do ar")
2. **Ativação no cliente ou no servidor**: o sistema deve suportar as duas opções, usadas dependendo da aplicação e da carga do sistema.

Extensão dos tipos de dados básicos

3. **Segurança**: com relação à execução das funções criadas pelo usuário, podendo causar problemas acidentalmente ou deliberadamente ao SGBD, pode-se usar uma das 3 técnicas abaixo:
   - **Funções ativadas em um processo separado do SGBD (seguro mas lento)**
   - **Uso de "firewalls", examinando o código objeto gerado pelo usuário e modificando se for encontrado algum problema (degradação mínima da performance)**
   - **Funções escritas em uma Linguagem Interpretada como Java (performance da própria linguagem)**

Extensão dos tipos de dados básicos

4. **"Callback"**: uma função deve se comportar da mesma maneira se estiver associada a um programa de aplicação ou chamada através de uma consulta em SQL.
5. **Métodos de acesso definidos pelo usuário**: apropriados aos tipos de dados definidos.
6. **Tipos de tamanho arbitrário**: os tipos definidos pelo usuário não têm restrição de tamanho.

Objetos Complexos

Objetos que são compostos por múltiplos tipos básicos ou definidos pelo usuário

Construtores de tipos:

- registros
- conjuntos (coleções)
- referências (apontadores)

Objetos Complexos

Outros construtores de tipos:

- Listas
- Pilhas
- Filas
- Array flexível (ex: texto)

  Aparentemente com utilidade limitada, estes construtores podem ser úteis em aplicações específicas.

Objetos Complexos

Usando construtores de tipos:

```
create type t-depto {
    dnome           varchar (30),
    pavimento       integer,
    fone            phone-t,
    veiculos        setof(t-veiculo),
    gerente         ref(t-emp),
    trabalhadores setof(ref(t-emp));
create table depto of type t-depto;
```


Objetos Complexos

Usando funções definidas pelo usuário:

```
create function bonus (t-emp)
    returns (float) as select (salario * 0.1);
select e.nome
    from emp e
    where bonus (e) > 1000;
```


Objetos Complexos

Usando *collections* para reduzir total de dados:

- Tradicional

```
select d.nome, e.nome, e.salario
    from emp e, depto d
    where e.dnome = d.nome and d.nome = 'di';
```

- Em um sistema O-R

```
select d.nome, set (e.nome, e.salario)
    from depto d
    where d.nome = 'di';
```


Novos Tipos Básicos e Objetos Complexos

- Razões para a existência destes mecanismos:
  - **Naturalidade: tipos como data, valor-monetário, telefone devem ser naturalmente definidos**
  - **Encapsulamento:**
    - ✓**os tipos definidos são acessados e manipulados através das funções definidas sobre o tipo**
    - ✓**os objetos complexos não são encapsulados e seus campos podem ser acessados individualmente**

Novos Tipos Básicos e Objetos Complexos

- Identificador de objetos (OID): identificam unicamente os objetos complexos. Os tipos não possuem OID
- Conversão de dados: conversão de formato na entrada e saída de tipos definidos são essenciais

Novos Tipos Básicos e Objetos Complexos

- Características de um sistema O-R:
  1. Dar suporte a uma rica coleção de tipos complexos
  2. Disponibilizar funções para todos os tipos complexos
  3. Não ter limite no tamanho dos tipos complexos
  4. Incluir em SQL suporte para tipos complexos

Herança

- Outro mecanismo para criação de novos tipos.
- A herança permite agrupar tipos compostos (registros) em uma hierarquia de herança.
- Incentiva a modularidade e a reutilização consistente dos componentes do esquema.
- É aplicada apenas sobre tipos de dados, as tabelas são associadas aos tipos.
- Um bom SGBD O-R permite herança múltipla.

Herança

create type t-pessoa (nome varchar(30));
create type t-emp (salario int) under t-pessoa;
create type t-estud (media float) under t-pessoa;

Herança múltipla:
create type t-estud-emp (percentual float)
under t-emp, t-estud;

Herança

create table pessoa of type t-pessoa;
create table emp of type t-emp under pessoa;
create table estudante of type t-estud under pessoa;
create table estud-emp of type t-estud-emp
under emp, estudante;

select nome from emp where salario = 1000;
/* inclui a tabela emp e as tabelas abaixo dela na hierarquia */
select nome from only (emp) where salario = 1000;
/* só a tabela emp é consultada */

Herança

- O comportamento da herança de funções é determinada por seus argumentos
- Quando uma função é aplicada sobre um tipo e não está definida naquele tipo, o sistema procura nos supertipos da hierarquia.
- É permitido overloading de funções

Herança

Resumo das características:

1. Tanto herança de dados quanto de funções devem ser oferecidas
2. Overloading: possibilidade de múltiplas implementações de funções com o mesmo nome
3. Tipos e tabelas devem ser conceitos separados
   - **Hierarquia de tipos com tabelas definidas separadamente**
4. Herança múltipla deve ser permitida

Regras

Regras são importantes para garantir a integridade dos dados. Elas também podem ser definidas de acordo com as restrições de uma aplicação.

- Nos SGBD relacionais: uso de "gatilhos"
- Nos SGBD O-R:

on evento
do ação

Regras podem ser executadas imediatamente antes ou depois de um evento.

Regras

Tipos de regras:

- Regras atualização-atualização

create rule altera-salario as
on update to emp.salario where current-nome = 'Marcos'
do update emp
set salario =new.salario
where nome = 'Jane';

Este tipo de regra é chamado de "gatilho" e serve para garantir a integridade dos dados.

Regras

- Regras consulta-atualização

create rule auditoria-salario as
on select to emp.salario where current-nome = 'Marcos'
do insert into valores -audit
(current.salario, user, current-timestamp)

Neste caso, quando o salário de Marcos for consultado, uma inserção é feita em uma tabela de auditoria.

Regras

- Regras atualização-consulta

create alerter alerta-atualiza-sal
(mechanism = 'callback');
create rule atualiza-sal-Marcos as
on update to emp.salario where current-name = 'Marcos'
do alert alerta-atualiza-sal;

Neste caso, quando o salário de Marcos for atualizado, o sistema notifica o usuário sobre a atualização

Regras

- Regras consulta-consulta

create rule salario-Marcos-Jane as
on select to emp.salario where current-nome = 'Jane'
do instead
select salario from emp
where nome = 'Marcos';

Esta regra indica que o salário de Marcos deve ser consultado quando se quiser acessar o de Jane.

Regras

Aspectos negativos do uso de regras:

- Várias regras ativando o mesmo evento, algumas vezes pode causar resultados imprevisíveis
- Regras encadeadas podem gerar laços infinitos
- Cancelar uma ação que é parte de uma regra também cancela a transação
- O tempo de ativação de uma regra pode mudar o estado final do banco de dados

Regras

Resumindo, um sistema de regras deve oferecer:

- Eventos e ações que podem ser consultas ou atualizações
- Integração com outras facilidades O-R
  - Funções podem aparecer em uma regra,
  - Regras devem ser herdadas, etc.
- Execução semântica imediata ou postergada, na mesma transação ou em transações diferentes
- Garantia de não entrar em 'loop'

Projeto de Banco de Dados para SGBD O-R

- Um projeto mal feito é responsável pelo fracasso de muitas aplicações
- Um bom projeto é um desafio mesmo para BD relacionais
- A maioria dos produtos são baseados no modelo E-R
  - **Identificar entidades, relacionamentos, atributos, chaves primárias e estrangeiras**
  - **Mapear o esquema E-R gerando tabelas normalizadas**

Projeto de Banco de Dados para SGBD O-R

Por que tantos problemas com projetos de BD?

1. Dificuldade em ajustar o esquema
   - Normalizar ou desnormalizar tabelas em função de melhoria da performance analisando as conseqüências para o resto do esquema

⇩

Intuição e Prática

Projeto de Banco de Dados para SGBD O-R

2. Desinteresse em seguir todos os passos do projeto

   O projeto é um processo evolutivo e tem que ser revisto em diversas circunstâncias. É uma tarefa que demanda um grande esforço e é necessário uma pessoa responsável unicamente por esta atividade

Projeto de Banco de Dados para SGBD O-R

- Projeto irreal da aplicação
  - Um implementador pode usar uma dada funcionalidade do SGBD para resolver vários problemas sem analisar se é um bom mecanismo ou não. Ex: triggers (gatilhos)

⇩

Sérios problemas de performance

Projeto de Banco de Dados para SGBD O-R

3. Fracasso em atingir os objetivos de performance em grandes BD

   Carregar e testar grandes BD previamente para que qualquer problema de performance seja identificado a tempo de uma ação corretiva

⇩

Este passo não pode ser menosprezado

Projeto de Banco de Dados para SGBD O-R

4. Dificuldade em capturar todo o esquema
   - Capturar todas as restrições do esquema, regras do tipo: “todo empregado deve trabalhar em um departamento” e todas as ações a serem tomadas em caso de falha.

⇩

Difícil de conseguir este tipo de informação do usuário

Projeto de Banco de Dados para SGBD O-R

5. Excesso de foco na modelagem formal da aplicação
   - Busca por uma ferramenta apropriada e decisão de construir uma própria depois de uma longa procura.

⇩

Atividades que não ajudam na construção da aplicação

Projeto de Banco de Dados para SGBD O-R

Desafios em projeto de BD O-R:

O-R é um super-conjunto do relacional, os problemas são os mesmos porém mais complexos

1. Escolher entre as várias opções
   - **Como implementar as tabelas emp e depto ?**
     - ✓**O atributo dnome na tabela empregado como ref(depto)**
     - ✓**O relacionamento trab-em como setof(ref(emp)) na tabela depto**

Projeto de Banco de Dados para SGBD O-R

2. Escolher entre as várias opções (Cont.)
   - ✓A função local(emp) que vai localizar em que local o empregado trabalha
   - ✓ou qualquer um dos casos anteriores de outra forma
   - **Em um bom esquema O-R, pode-se ter também:**
     - ✓**herança**
     - ✓ **extensão de tipos básicos**
     - ✓ **objetos complexos**

Projeto de Banco de Dados para SGBD O-R

Realmente se requer arte e uma boa intuição para escolher a melhor maneira de implementar um esquema O-R.

Pela riqueza do modelo, o projeto de um banco de dados O-R é proporcionalmente mais difícil.

Projeto de Banco de Dados para SGBD O-R

3. Representação de dados versus procedimentos

Consideremos o exemplo de data e idade de um empregado.

- **No modelo relacional seriam implementados dois atributos com controle de consistência entre eles.**
- **No O-R, existe a possibilidade de implementação de procedimentos.**

Projeto de Banco de Dados para SGBD O-R

Logo, um pode ser implementado como um atributo e o outro calculado por um procedimento e identificado quando o procedimento será executado (se na entrada do dado ou em uma consulta).

⇩

Espaço versus performance

Projeto de Banco de Dados para SGBD O-R

4. Regra versus representação de dados

Como usar regras para representar dados?

```
create rule salario-Marcos-Jane as
    on select to emp.salario where current-
    nome = 'Jane'
    do instead
            select salario from emp
            where nome = 'Marcos';
```


Projeto de Banco de Dados para SGBD O-R

Neste caso o salário de Jane nunca é armazenado, mas computado quando necessário. Dois aspectos devem ser considerados:

- **Performance: frequência de acesso ao salário de Jane**
- Semântica da regra: o que acontece se Marcos for demitido ?

Projeto de Banco de Dados para SGBD O-R

Em resumo:

- **Projeto O-R é mais difícil que o relacional**
- **São necessárias pessoas qualificadas para definir qual a melhor opção e não existem muitas no mercado.**
- **SGBDs devem ser mais fáceis de usar no futuro**
  - ✓**O esquema deve ser ajustado automaticamente pelo sistema para melhorar performance**
  - ✓**O sistema deveria prevenir o usuário das consequências de suas escolhas em termo de performance.**

Opções de Arquitetura para o SGBD

Os principais sistemas O-R são:

- Illustra
- Omniscience
- UniSQL

Estes são sistemas que escreveram a 'máquina' O-R completamente ('do zero').

Os sistemas relacionais têm 'máquinas' que são cuidadosamente otimizadas.

Opções de Arquitetura para o SGBD

O que deve ser feito para transformar uma 'máquina' relacional em uma O-R?

- Re-escrever o tradutor para ser dirigido a tabelas
- Re-escrever o otimizador e incluir funções de otimização e suporte à herança e objetos complexos
- Re-escrever o executor para dar suporte às funções definidas pelo usuário e seus 'links' dinâmicos

Opções de Arquitetura para o SGBD

- Re-escrever os métodos de acesso para serem abertos a novas implementações
- Re-escrever o código da B-tree para ser genérico
- Re-escrever o gerenciamento de armazenamento para suportar tipos de dados longos
- Re-escrever as aplicações para consultas genéricas

Opções de Arquitetura para o SGBD

As quatro estratégias individuais dos sistemas relacionais são:

1. **Não fazer nada**
2. **Re-escrever a 'máquina' completamente**
3. **Evoluir iterativamente adicionando novas funcionalidades**
4. **Escrever uma 'capa'**

Opções de Arquitetura para o SGBD

As estratégias com um parceiro O-R são:

5. **Implementar um 'gateway' de um sistema O-R para um relacional**
6. **Acoplar a 'máquina' de execução de um sistema O-R ao gerenciador de armazenamento de um sistema relacional**

As estratégias para um sistema O-O são:

7. **Estender um SGBD O-O com funcionalidades de um sistema O-R**
8. **Acoplar uma 'máquina' O-R sobre um SGBD O-O**

Opções de Arquitetura para o SGBD

Estratégia	Aspectos	Fabricantes
Evoluir o sistema	Mais seguro mas mais demorado Boa estratégia para um só fabricante	IBM, Oracle Microsoft
Criar uma 'capa'	Possível perda de performance	Hewlett-Packard
Fazer um 'gateway'	Possível perda de performance mas produz rapidamente um produto	Computer Associates
Acoplar O-R a armaz. relacional	Boa estratégia se a performance não for comprometida	Tandem
Estender um sistema O-O	Consome tempo. Problemas de segurança inerentes	Maioria dos fabricantes O-O
Acoplar O-R à ling. persistente	Boa estratégia se a performance não for comprometida	Versant

PRINCIPAIS SERVIDORES UNIVERSAIS

- Informix Dynamic Server with Universal Data Option
- IBM DB2 Universal Server
- Oracle8i

Object Type
Object Tables
OID
REFs
Coleções

ORACLE 8i - Object Type

- Tipo definido pelo usuário que captura tanto a estrutura como o comportamento de um objeto
- Uma instância, chamada de objeto, pode ser armazenada em tabelas (objetos persistentes) ou podem existir apenas temporariamente (objetos transientes)

Sintaxe resumida:

```
CREATE TYPE <nome do tipo> AS OBJECT (
    <lista de atributos e métodos>
);
```


ORACLE 8i - Object Type

- A declaração de um tipo também pode incluir métodos
  - **Método Construtor**
  - **Métodos Membro**
  - **Método MAP**
  - **Método ORDER**
- Um object type 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.

ORACLE 8i - Object Tables

- Armazenam objetos
- São tabelas do banco de dados definidas usando somente um object type, ao invés de colunas relacionais
  - **Suas colunas correspondem aos atributos do object type utilizado**
  - **Suas linhas são objetos do tipo da tabela**
  - **Cada objeto na tabela possui um object identifier (OID).**

Sintaxe resumida:
```
CREATE TABLE <tabela> OF <object type> (
  <declaração de propriedades de atributos/chaves>);
```


ORACLE 8i - DML

- INSERT

Sintaxe resumida:
```
INSERT INTO <tabela> VALUES
            (<object type> (v1, ..., Vn));
```

Construtor

- SELECT

Sintaxe resumida:
```
SELECT X.<atributos> FROM <tabela> X
       WHERE ...;
ou
SELECT VALUE(P) FROM <tabela> P
   WHERE ...;
```


ORACLE 8i - DML

- UPDATE

Sintaxe resumida:
```
UPDATE <tabela> P SET P.atributo = v1
 WHERE ...;
```

- DELETE

Sintaxe resumida:
```
DELETE FROM <tabela> T WHERE ....;
```


ORACLE 8i - OIDs e REFs

- 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 8i - OIDs e REFs

- REFs

  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 OIDs, ou seja, só é possível referenciar objetos armazenados em object tables**

ORACLE 8i - OIDs e REFs

- REFs em Colunas
  - **Uma coluna de uma tabela (ou um atributo de um object type) pode ser declarado como sendo do tipo REF**
    - ✓**REF <tipo de objeto>**
- REFs como Operadores
  - **Quando é necessário obter o identificador de um objeto em uma tabela, utiliza-se o operador REF(), tendo como argumento o aliás de uma object table**
    - ✓**SELECT REF(P) FROM <tabela> P WHERE ...;**

ORACLE 8i - OIDs e REFs

- O OPERADOR DEREF()

  Retorna um objeto referenciado por uma coluna do tipo REF

Sintaxe resumida:
```
SELECT DEREF(<atributo do tipo REF>)
   FROM <tabela>;
```


ORACLE 8i - Coleções

- Dois tipos
  - **Varying arrays**
  - **Nested tables**
- Atributos de um object type e colunas de uma tabela podem ser do tipo coleção

ORACLE 8i - VARRAY

- É um conjunto ordenado de elementos do mesmo tipo
  - **Cada elemento possui uma posição que o identifica unicamente no varray**
  - **Uma posição é um inteiro no intervalo entre 1 e um valor máximo declarado**
  - **Um varray é armazenado "in line", ou seja, na mesma estrutura da tabela.**

Sintaxe resumida:
```
CREATE TYPE t_array AS VARRAY(n) OF t_type;
```


ORACLE 8i - NESTED TABLES

- É uma coleção não ordenada de elementos homogêneos
- Não impõe limites para o número de elementos e é armazenada "out line", ou seja, em uma outra estrutura (tabela)
- Não é permitido o uso de NESTED TABLE recursiva

Sintaxe resumida:
```
CREATE TYPE t_nested AS TABLE OF <object type>;
CREATE TYPE t_type1 (..., atn t_nested, ... );
CREATE TABLE type1 OF t_type1;
NESTED TABLE atn STORE AS atn_t;
```


ORACLE 8i - NESTED TABLES

- Manipulando Elementos
  - **É necessário utilizar uma extensão de SQL chamada "flattened subquery" cujo propósito é selecionar uma única Nested Table a ser manipulada**
  - **A expressão THE( ) é utilizada para tal finalidade**

Sintaxe resumida:
```
SELECT C.<atributo> FROM
THE(SELECT D.atn FROM type1 D WHERE ...) C;
```

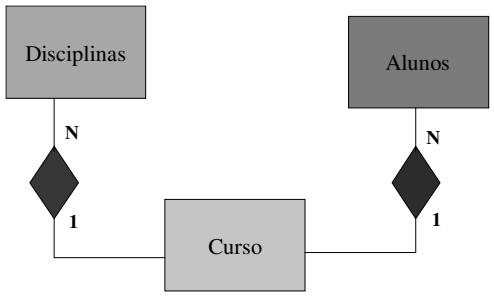

ORACLE 8i - NESTED TABLES

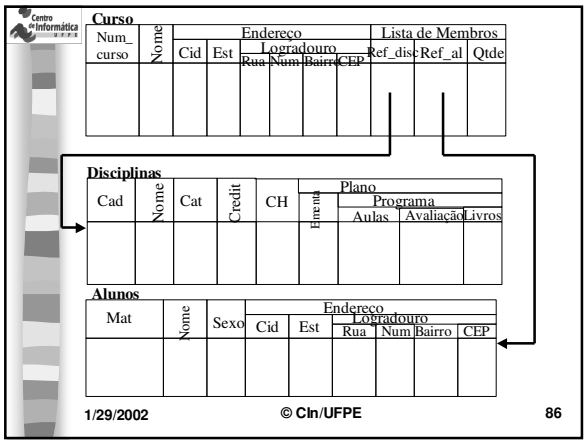
- Um grande potencial das Nested Tables torna-se visível quando é combinado o seu uso com o uso de referências (REF)

Sintaxe resumida:
```
CREATE TYPE t_tipo as object ( ..., atn ref t_tipo1, ...);
CREATE TYPE t_lista_tipo1 as table of t_tipo;
CREATE TYPE t_tipo2 as object(.., lista1 t_lista_tipo1,...);
```

Uma Aplicação

Exemplo - Cursos, as Disciplinas do perfil e seus Alunos





Criação de Object-Types

```
create type t_logradouro as object(
   Rua Varchar2(20),
   Num Varchar2(10),
   Bairro Varchar2(20),
   Cep Char(8)          Tipo Logradouro
);
```

```
create type t_endereco as object(
   Cid Varchar2(20),
   Est Char(2),
   Logradouro t_logradouro
);              Tipo Endereço
```


Criação de Object-Types

```
create type t_programa as object(
   Aulas Varchar2(100),
   Avaliacao Varchar2(50),
   Livros Varchar2(60)
);   Tipo Programa de Disciplina
```

```
create type t_aluno as object(
   Mat integer,
   Nome Varchar2(20),
   Sexo char(1),
   Endereco t_endereco
);                    Tipo Aluno
```

```
create type t_plano as object(
   Ementa Varchar2(40),
   Programa t_programa
);   Tipo Plano de Disciplina
```


Criação de Object-Types

```
create type t_disciplina as object(
   Cad integer,
   Nome Varchar2(20),
   Cat Char(2),
   Credit integer,
   CH integer,
   Plano t_plano
);          Tipo Disciplina
```

```
create type t_membro as object(
   Ref_disc ref t_disciplina,
   Ref_al ref t_aluno,
   Qtde integer
);          Tipo Membro de Curso
```

```
create type t_lista_membro as table of t_membro;
                                  Tipo da Nested Table
```


Criação de Object-Types

```
create type t_curso as object(
   Num_curso integer,
   Nome Varchar2(20),
   Endereco t_endereco,
   Lista_membro t_lista_membro,
   member function Total_disciplinas return number,
   member function Total_UnidMatric return number,
   Pragma Restrict_references (Total_disciplinas,WNDS,WNPS),
   Pragma Restrict_references (Total_UnidMatric,WNDS,WNPS)
);                                          Tipo Curso
```

Criação de Object-Tables

```
create table Alunos of t_aluno(
  Mat primary key);          Tabela Alunos

create table Disciplinas of t_disciplina(
  Cad primary key);     Tabela Disciplinas

create table Curso of t_curso(                    Tabela Cursos
  Num_curso primary key)
  Nested table lista_membro store as tabela_lista_membro;

alter table tabela_lista_membro add(scope for(Ref_disc)
                is disciplinas);

alter table tabela_lista_fornecimento add(scope for(Ref_al)
                is aluno);
```


Criação de Funções Membro (Métodos)

```
create or replace type body t_curso as
  member function Total_disciplinas return number is
  begin
    return self.lista_membro.count
  end;
end;                              Método Total de Disciplinas
```


Criação de Funções Membro (Métodos)

```
create or replace type body t_curso as
   member function Total_UnidMatric return number is
   contador integer;
   nUnidades number := 0;
   begin
     for contador in 1..self.lista_membro.count loop
          nUnidades := nUnidades +
            self.lista_membro(contador).Qtde;
     end loop;
     return nUnidades;
   end;
end;          Método Total de Unidades de Matrícula
```


Inserção de Dados

```
insert into Alunos values
  (t_aluno(1,'Vera Fischer','F',t_endereco('Rio de Janeiro',
       'RJ',t_logradouro('Rua da Urca',100,'Urca',
         '207390000'))));

insert into Alunos values
  (t_aluno( 2,'Jo Soares','M',t_endereco('Sao Paulo','SP',
      t_logradouro('Av. Paulista',5000,'Paraiso','11500105'))));
                              Inserir Dados na Tabela Alunos
```


Inserção de Dados

```
insert into Disciplinas values
  (t_disciplina(10,'Banco de Dados','Ob',4,60,
    t_plano('Introdução, Modelos, Modelo Relacional',
    t_programa('Introdução - 10h, Modelos - 20h, Modelo
      Relacional - 30h','Prova - 50%, Trabalhos - 50%',
        'Elsmary & Navathe'))));

insert into Disciplinas values
  (t_disciplina( 11,'Programação 1','Ob',4,60,
    t_plano('Introdução, Estruturas de dados, Laços',
    t_programa('Introdução - 15h, Estruturas de dados - 25h,
      Laços - 20h','Prova - 40%, Trabalhos - 60%',
      'Viviane Mizrahi'))));  Inserir Dados na Tabela Disciplinas
```


Inserção de Dados

```
insert into Curso values
  (t_curso(1,'Ciência da Computação',
    t_endereco('Recife','PE',
    t_logradouro('Campus da UFPE',100,
      'Cidade Universitária','50730040')),
       t_lista_membro()));

insert into Curso values
  (t_curso(3,'Sistemas de Informação',
     t_endereco('Recife','PE',
     t_logradouro('Av. Abdias de Carvalho',1000,
      'Bongi','50100031')),
       t_lista_membro())); Inserir dados em Curso
```

Inserção de Dados

```
insert into
  the(select c.lista_membro
      from curso c
      where p.num_curso = 1)
  select ref(d),ref(a),1
  from disciplinas d,alunos a
  where d.Cad = 10 and a.Mat = 1;
```

Inserções na Nested Table

```
insert into
  the(select c.lista_membro
      from curso c
      where c.num_curso = 1)
  select ref(d),ref(a),10
  from disciplinas d,alunos a
  where d.Cad = 11
   and a.Mat = 1;
```


Consultas

- Listar nomes dos cursos e disciplinas, bem como quantidade de unidades de matrícula do aluno com matrícula = 2

```
select l.ref_disc.nome,c.nome,l.qtde
from curso c, the(select lista_membro
                  from curso c1
                  where c1.num_curso = c.num_curso) l
where l.ref_al.Mat = 2;
```


Consultas

- Listar nomes dos alunos e disciplinas, quantidade de unidades de matrícula do curso com número = 2

```
select l.ref_al.nome,l.ref_disc.nome,l.qtde
from curso c,the(select lista_membro
                 from curso c1
                 where c1.num_curso = 2) l
where c.num_curso = 2;
```


Consultas

- Listar nome do curso, total de disciplinas e total de unidades de matrícula, do curso com número = 1

```
select c.nome,c.Total_disciplinas(),c.Total_UnidMatric()
from curso c
where c.num_curso = 1;
```


Um Pouco de História...

Tecnologia	Anos 70	Anos 80	Anos 90
Laboratórios de Pesquisa	Relacional	Orientado a Objetos	Ainda não identificado
Meio Comercial	Hierárquico e Redes	Relacional	Objeto-Relacional
Sistemas Legados (legacy)		Hierárquico e Redes	Relacional