cin.ufpe.br

Centro de Informática

U · F · P · E

Universidade Federal de Pernambuco

Banco de Dados Objeto Relacional

Aula1

Apresentado por:

Robson do Nascimento Fidalgo

rdnf@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

Introdução

- SGBDR – Visão Geral
  - É voltado para aplicações convencionais
  - Possui uma forte base matemática (Modelo Relacional)
    - Permite implementar SGBD que armazenam e recuperam, eficientemente, grandes volumes de dados
  - É fortemente dependente da 1a. Forma Normal
    - Não permite tabelas aninhadas
      - Isto é, atributo composto e/ou multivalorado
  - Não permite OO: criação de tipos e Herança
  - Deste 1986 (SQL1) tem uma linguagem padronizada pelo ANSI (American National Standards Institute) e atualmente está na SQL:1992 (ou SQL2)

Introdução

- SGBDR – Visão Geral

nome	*rua*	*cidade*	*nro-conta*
Mário	Av. S.Carlos	S.P.	1234
Rui	Rua XV	S.Carlos	1333
Rui	Rua XV	S.Carlos	7556
Silvia	Av.D.Pedro	Itu	5512
Silvia	Av.D.Pedro	Itu	7556

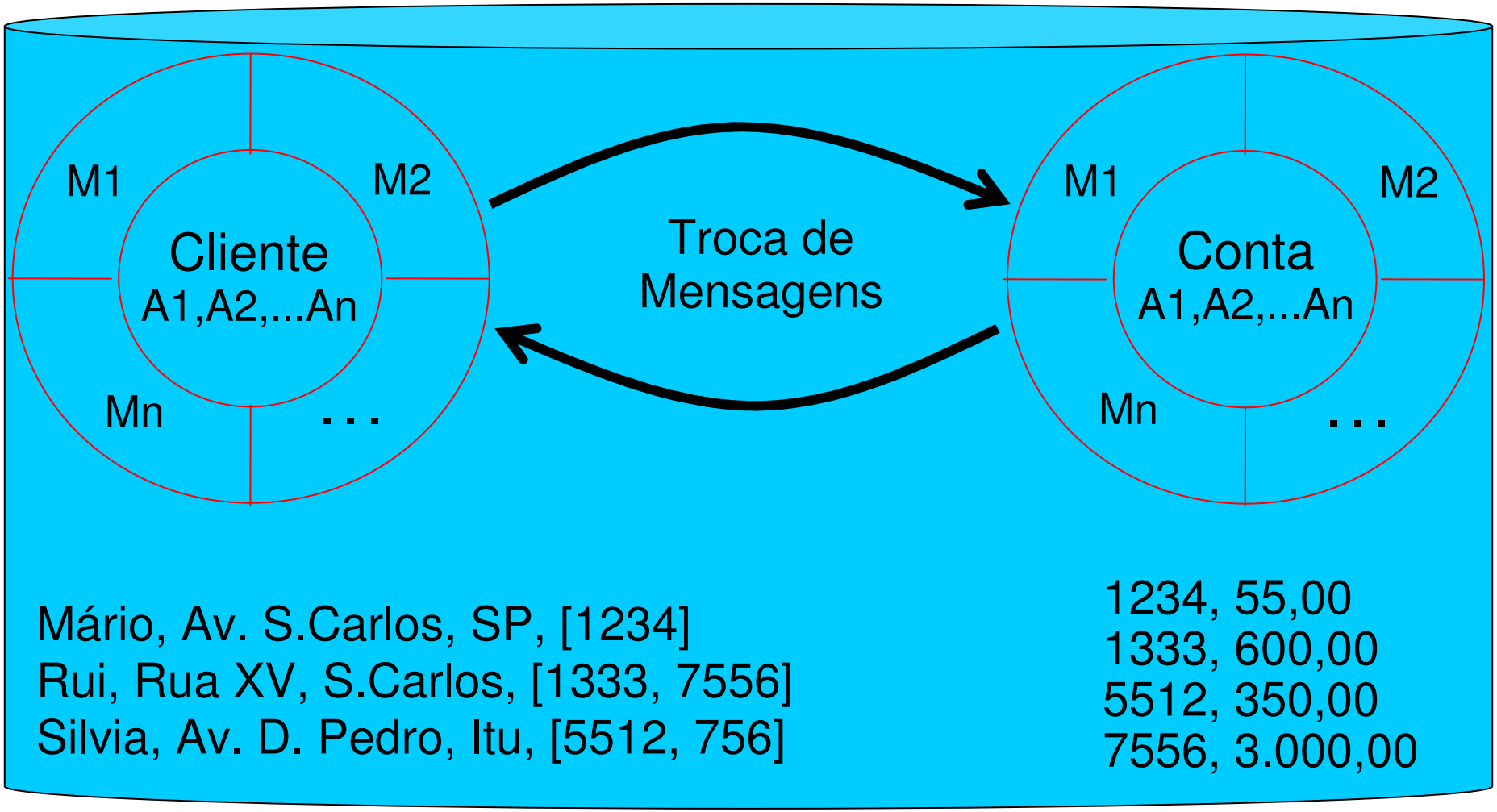
nro-conta	*saldo*
1234	55,00
1333	600,00
5512	350,00
7556	3.000,00

Introdução

- SGBDOO – Visão Geral
  - É voltado para aplicações ditas avançadas, nas quais o enfoque central está na especificação de objetos complexos
  - É uma abordagem fortemente influenciada por linguagens de programação OO (LPOO)
  - Pode ser entendido como uma forma de adicionar funcionalidades de SGBDs em um ambiente de LPOO
    - O SGBDOO fornece uma interface direta para que uma LP possa persistir e compartilhar objetos
  - O ODMG tem trabalhado para padronizar um(a):
    - Modelo de dados (Object Data Model - ODM)
    - Linguagem de consulta (Object Query Language - OQL) e
    - Linguagem de definição (Object Definition Languague - ODL)

Introdução

- SGBDOO – Visão Geral

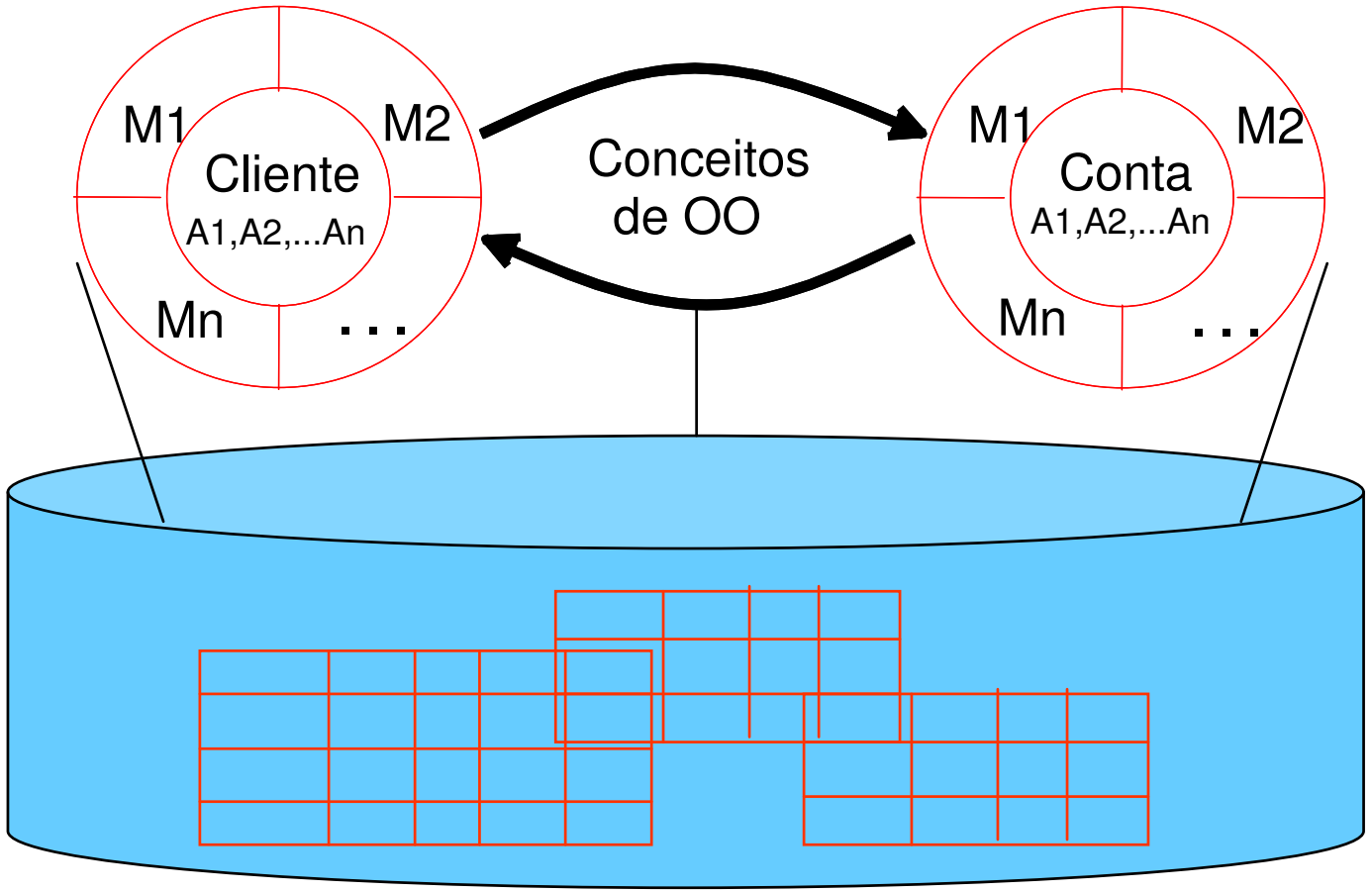


Introdução

- SGBDOR – Visão Geral
  - É uma proposta de extensão dos SGBDR, incorporando funcionalidades para suportar aplicações ditas avançadas
  - É uma camada OO sobre um engenho Relacional.
  - O padrão SQL:1999 (ou SQL3) incorpora o suporte para o modelo de dados OR.

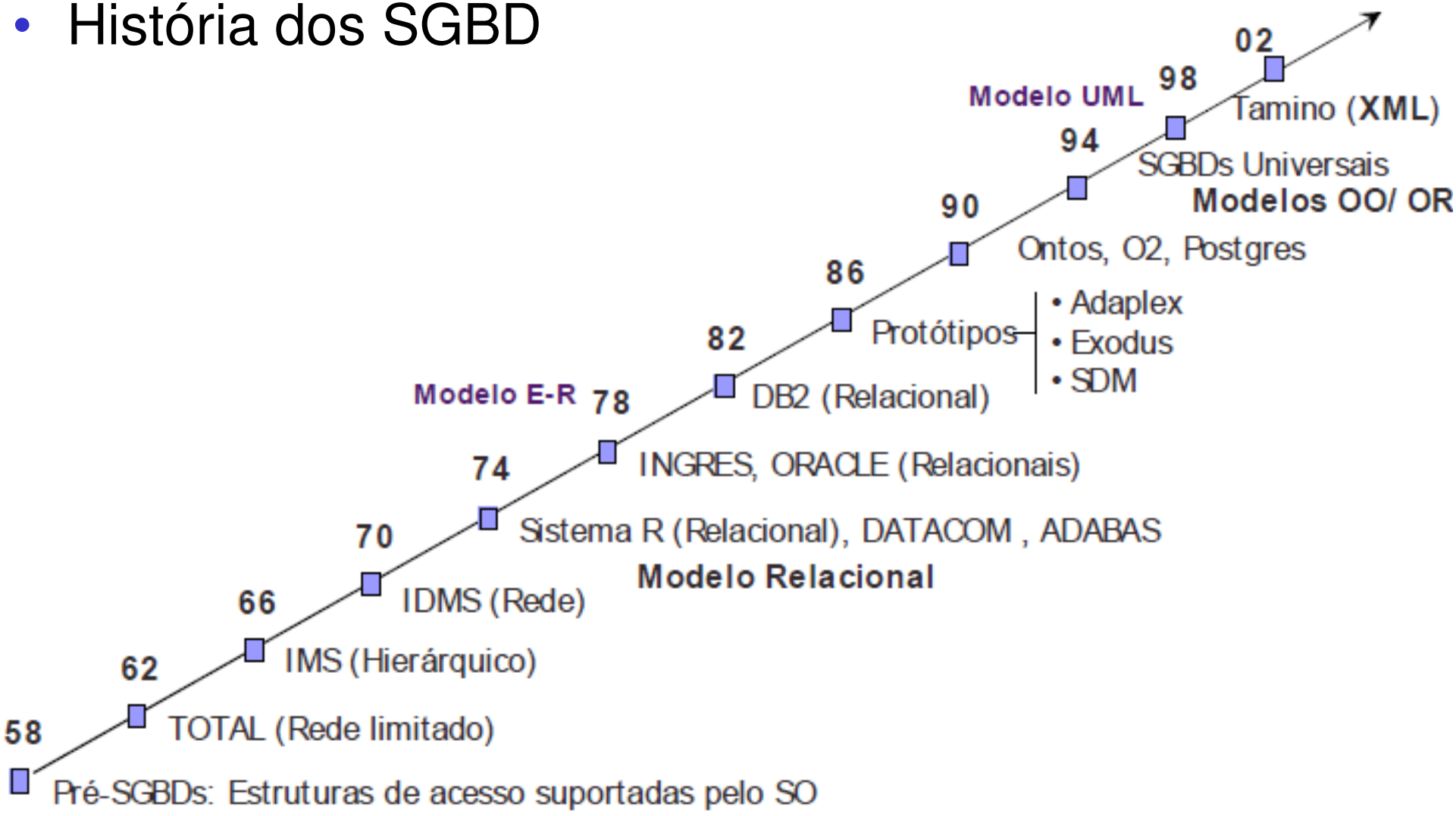
Introdução

- SGBDOR – Visão Geral



Introdução

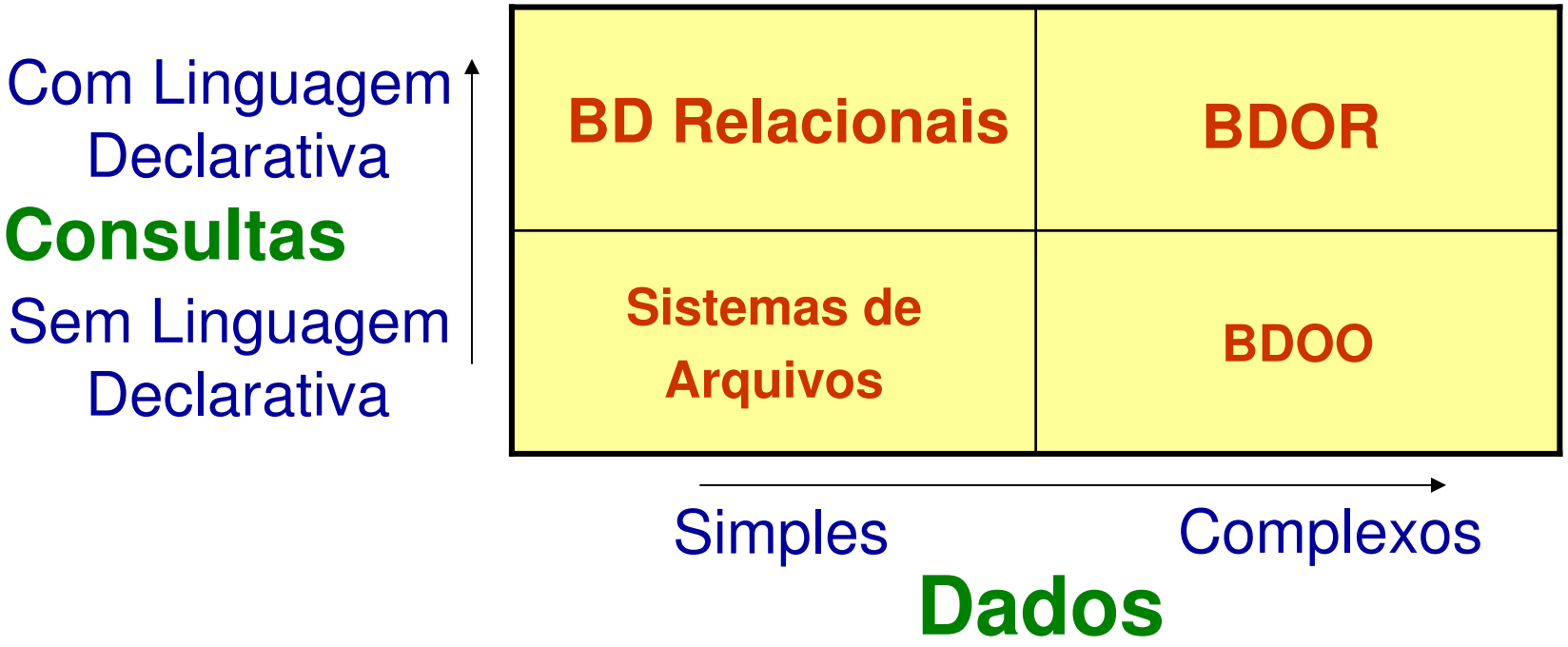
- História dos SGBD



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

Introdução

- Classificação de SGBDs



[M. Stonebreaker and D. Moore. (1997). Object Relational DBMSs: The next great wave. Morgan Kaufmann, 1997 apud. Nina Edelweiss e Renata de Matos Galante - Bancos de Dados Orientados a Objetos e Relacionais- Objeto, Mini-curso do SBBD2005]

Introdução

- Classificação de SGBDs

Com Linguagem Declarativa

Consultas

Sem Linguagem Declarativa

BD Relacionais	**BDOR**
Sistemas de Arquivos	**BDOO**

Simples Complexos

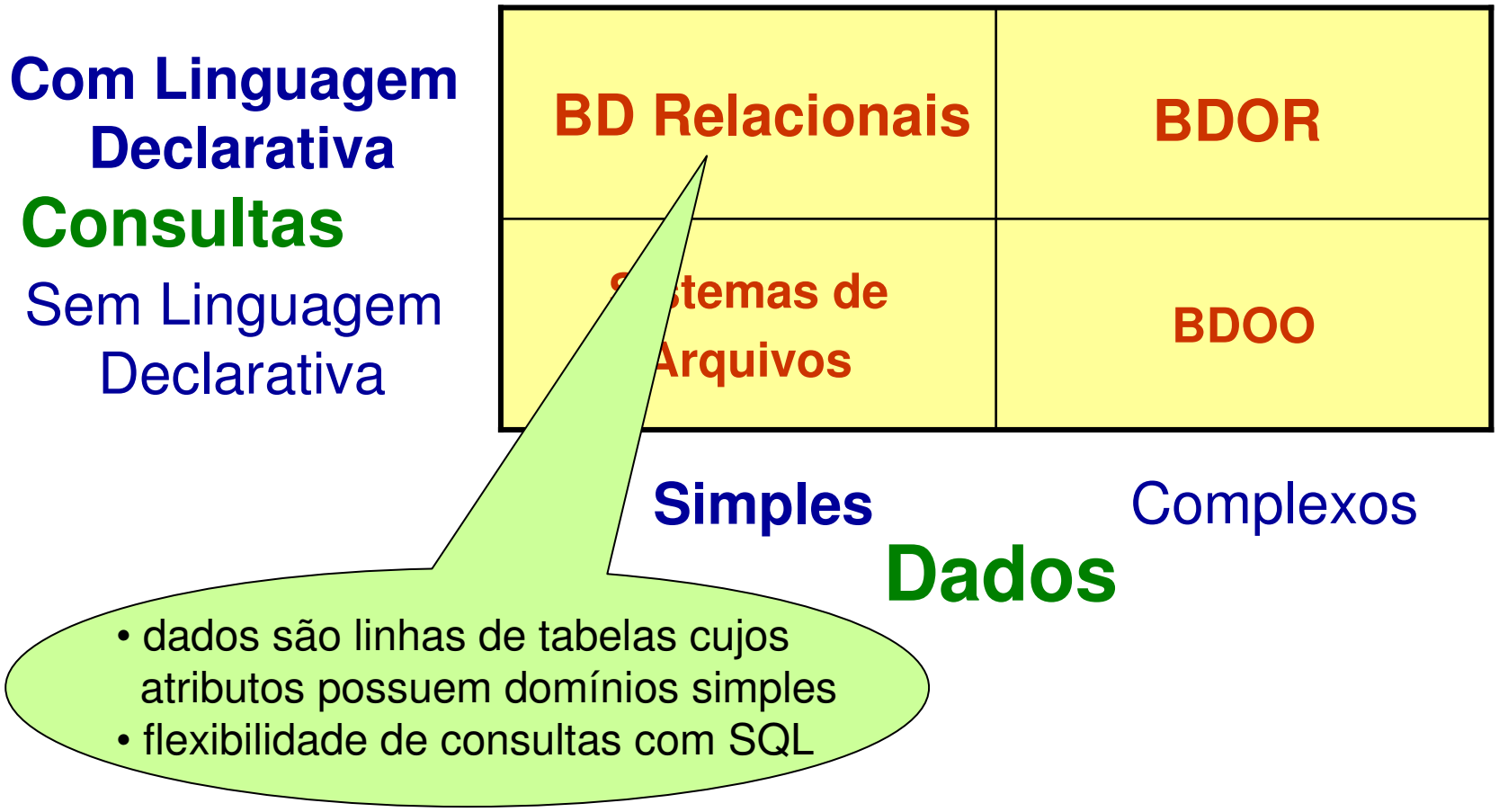
Dados

- dados são registros de tamanho fixo
- poucas consultas pré-definidas, em geral buscas por igualdade de campos dos registros

[M. Stonebreaker and D. Moore. (1997). Object Relational DBMSs: The next great wave. Morgan Kaufmann, 1997 apud. Nina Edelweiss e Renata de Matos Galante - Bancos de Dados Orientados a Objetos e Relacionais- Objeto, Mini-curso do SBBD2005]

Introdução

- Classificação de SGBDs



[M. Stonebreaker and D. Moore. (1997). Object Relational DBMSs: The next great wave. Morgan Kaufmann, 1997 apud. Nina Edelweiss e Renata de Matos Galante - Bancos de Dados Orientados a Objetos e Relacionais- Objeto, Mini-curso do SBBD2005]

Introdução

- Classificação de SGBDs

Com Linguagem Declarativa

Consultas

Sem Linguagem Declarativa

BD Relacionais	**BDOR**
Sistemas de Arquivos	**BDOO**

Simpl... **Complexos**

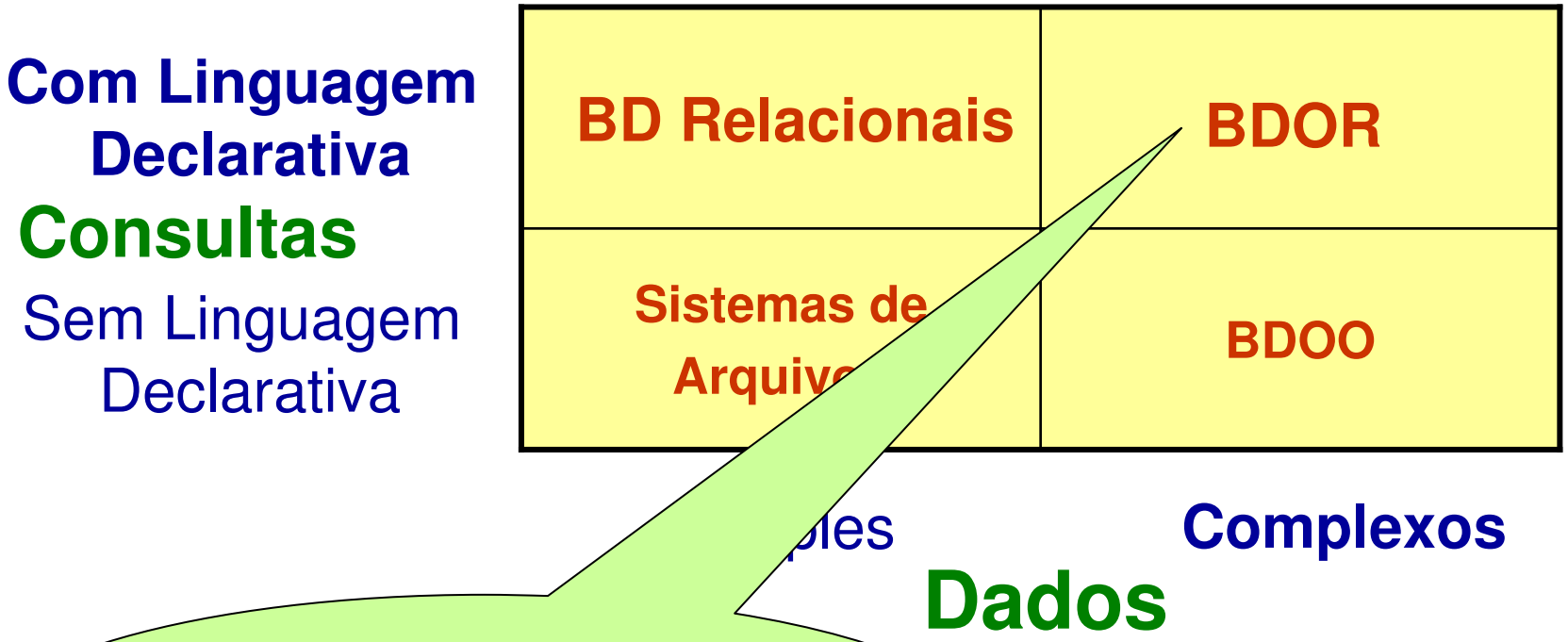
Dados

- dados são objetos com estrutura complexa
- capacidade de consulta limitada, baseada em navegação por objetos (poucos usam todos os recursos da OQL)

[M. Stonebreaker and D. Moore. (1997). Object Relational DBMSs: The next great wave. Morgan Kaufmann, 1997 apud. Nina Edelweiss e Renata de Matos Galante - Bancos de Dados Orientados a Objetos e Relacionais- Objeto, Mini-curso do SBBD2005]

Introdução

- Classificação de SGBDs



[M. Stonebreaker and D. Moore. (1997). Object Relational DBMSs: The next great wave. Morgan Kaufmann, 1997 apud. Nina Edelweiss e Renata de Matos Galante - Bancos de Dados Orientados a Objetos e Relacionais- Objeto, Mini-curso do SBBD2005]

Oracle OR – Visão Geral

- Conceitos Básicos
  - Tipo de Objetos
  - Métodos
  - Herança de Tipos
  - Evolução de Tipos
  - Tabelas de Objetos
  - Tabelas de Objetos com Herança
  - Objetos de Linha e Objetos de Coluna
  - Referência de Objetos
  - Coleção de Objetos

ORACLE 10g

Oracle OR – Visão Geral

- Tipos de Objetos
  - Tipo de objeto é um tipo abstrato de dados (TAD)
    - 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

- Tipos de Objetos
  - Permite capturar inter-relacionamento estrutural de objetos, estendendo a estrutura bidimensional relacional

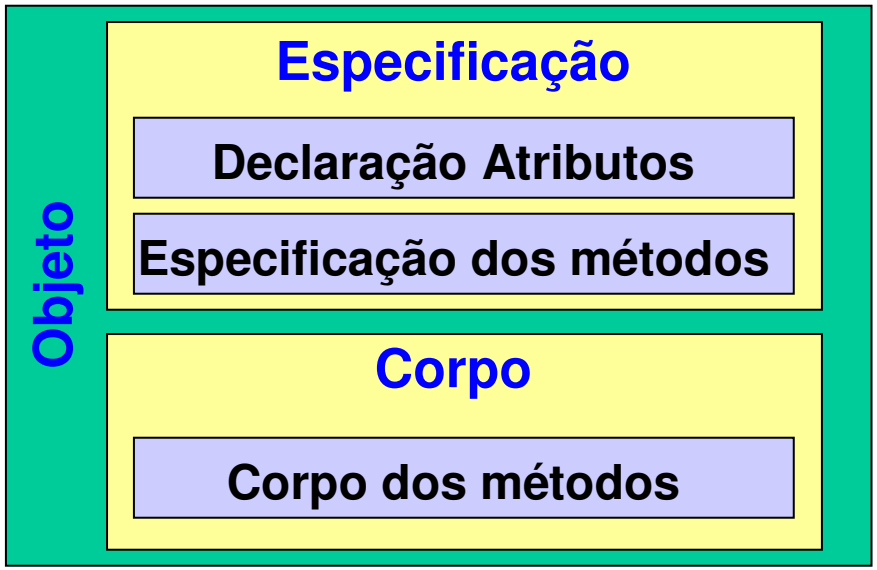
Clientes

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
. . .	. . .	. . .	. . .	. . .	. . .

A estrutura acima pode ser feita em estrutura OR, mas não pode ser feita em estrutura Relacional

Oracle OR – Visão Geral

- Tipos de Objetos
  - São especificados a partir de
    - Atributos → propriedades do objeto (opcional)
    - Métodos → procedimentos ou funções (opcional)



Interface Pública

Implementação Privada

Oracle OR – Visão Geral

- Tipos de Objetos
  - 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

- Tipos de Objetos
  - EX:

Especificação

```
CREATE OR REPLACE TYPE tp_pessoa AS OBJECT (
  id          INT,
  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 personToInt RETURN INT
) NOT FINAL;
```

para permitir criar um subtipo

Atributos

Métodos

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

Oracle OR – Visão Geral

- Tipos de Objetos
  - EX :

```
CREATE OR REPLACE TYPE tp_apartamento AS OBJECT (
  numero   INT,
  andar    INT,
  aluguel  NUMBER (9,2),
  ORDER MEMBER FUNCTION comparaOrdemApto(X IN tp_apartamento)
  RETURN INTEGER
);
```

Oracle OR – Visão Geral

- Tipos de Objetos
  - 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 setXY(newx NUMBER, newy number),
   NOT INSTANTIABLE MEMBER PROCEDURE desenhar
) NOT FINAL NOT INSTANTIABLE ;
```

Para criar um tipo abstrato.

Note que também tem que ser NOT FINAL!

Oracle OR – Visão Geral

- Tipos de Objetos
  - 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
    - CONSTRUTOR
    - MAP ou ORDER
    - MEMBER

Um tipo objeto sempre possui um **Construtor**, pode possuir zero ou mais métodos **Member** e pode possuir um método **Map** ou um método **Order**, porém não os dois

Oracle OR – Visão Geral

- Métodos - Construtor
  - Um tipo especial de método que permite criar novas instâncias de um determinado tipo de objeto, atribuindo valores aos seus atributos
  - O método construtor é uma função, devolvendo como resultado uma nova instância do objeto
  - É definido implicitamente pelo Oracle ou explicitamente pelo programador
    - Pode-se ter mais de 1 construtor para cada Tipo de Objeto

Oracle OR – Visão Geral

- Métodos - Construtor
  - Por padrão, seu nome é o mesmo do Tipo de Objeto
  - Seus parâmetros são exatamente os dados dos atributos do Tipo de Objeto que deseja-se instanciar, e eles devem ocorrer na mesma ordem da definição dos atributos

- Método - MEMBER
  - São os métodos mais comuns
    - Implementam as operações das instâncias do tipo
  - São invocados através da qualificação de objeto
    - Objeto.método()

Oracle OR – Visão Geral

- Métodos - MAP ou ORDER
  - Valores de tipos de dados primitivos como CHAR ou REAL possuem uma ordem predefinida, permitindo compará-los
  - Tipos de objeto possui múltiplos atributos de diferentes tipos de dados, não possuindo um eixo escalar de comparação

Oracle OR – Visão Geral

- Métodos - MAP ou ORDER
  - **MAP** faz uma comparação de tipos padrão, usando os atributos do objeto como fatores da comparação
  - **ORDER** 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
  - Também São invocados através da qualificação de objeto
    - Objeto.método()

Oracle OR – Visão Geral

- Métodos - MAP ou ORDER

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 – MAP
  - Permite comparar objetos, mapeando as instâncias dos objetos em um dos tipos escalares (ex: DATE, NUMBER, VARCHAR2) ou tipo SQL (ex: CHARACTER ou REAL)
  - Mapeia os objetos em um eixo escalar
  - Não tem parâmetro e retorna um dos atributos do objeto

Oracle OR – Visão Geral

- Métodos – MAP
  - É chamado implicitamente quando há comparação de objetos
    - DISTINCT, GROUP BY, UNION e ORDER BY (ordenação de linhas)
  - Só podem ser declarados em um subtipo se houver um método MAP declarado no supertipo
  - Exemplo de uso: obj1.map() > obj2.map()

Oracle OR – Visão Geral

- Métodos – ORDER
  - Faz comparação direta de objeto-a-objeto (não mapeia os objetos em um eixo escalar)
  - Retorna, segundo a lógica do método, se o objeto corrente (SELF) é menor, maior ou igual ao outro objeto que está sendo comparado
  - Indicado para comparações complexas (objetos binários)
    - EX: vídeo, áudio e som

Oracle OR – Visão Geral

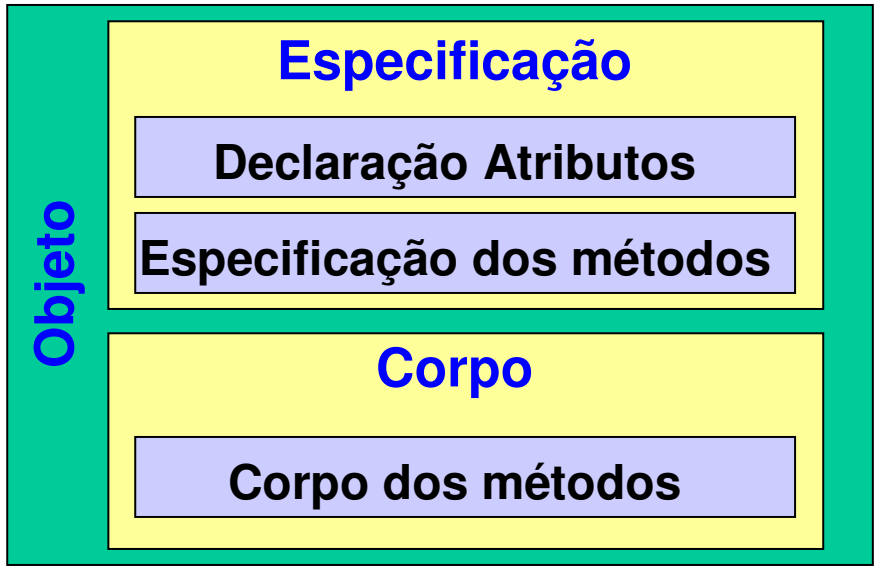
- Métodos – ORDER
  - Deve retornar sempre um inteiro com sinal (INTEGER)
  - Deve ter 1 parâmetro de entrada
    - Um objeto X do mesmo tipo de SELF
    - Exemplo de uso: obj.order(obj)
  - A interpretação do Oracle é sempre
    - Positivo, SELF > X
    - Negativo, SELF < X
    - Zero, SELF = X
  - Não podem ser definidos em subtipos e é menos eficiente do que um método MAP

Oracle OR – Visão Geral

- Métodos - MAP ou ORDER (resumindo)
  - São mutuamente exclusivos!
  - ORDER exige como parâmetro um obj. do mesmo tipo
  - ORDER compara o obj. corrente com o obj. do parâmetro
  - ORDER retorna inteiro: negativo <, positivo >, sem sinal =
  - MAP não exige parâmetro
  - MAP compara vários objetos
  - MAP retorna a posição do objeto no grupo

Oracle OR – Visão Geral

- Métodos



Interface Pública

Implementação Privada

Oracle OR – Visão Geral

- Métodos
  - 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
  - Exemplo de construção de corpo com método MEMBER :

```
CREATE OR REPLACE TYPE BODY tp_pessoa IS
  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;
-- continua
```

Oracle OR – Visão Geral

- Métodos
  - Exemplo de construção de corpo com método MAP:

```
MAP MEMBER FUNCTION personToInt RETURN INT IS
p INT := id;
BEGIN
    RETURN p;
END;
END;
```

Oracle OR – Visão Geral

- Métodos
  - Exemplo de construção de corpo com método 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;
```

  - Exemplo de uso do método construtor:

```
tp_funcionario(1,“José Silva Cardoso”,1000.00);
```

Oracle OR – Visão Geral

- Métodos
  - Exemplo de construção de corpo com método abstrato:

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

Note que o Método abstrato desenhar não foi especificado

cin.ufpe.br

Centro de Informática

U · F · P · E

Universidade Federal de Pernambuco