Banco de Dados Objeto-Relacional

Motivação (1/2)

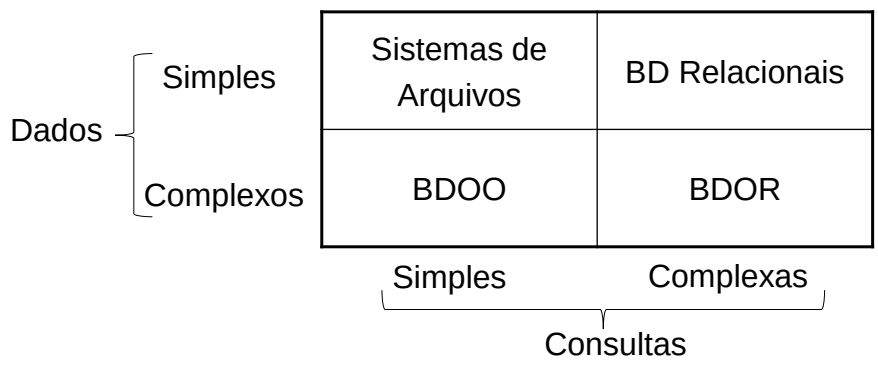
- SGBDs Relacionais (SGBDRs)
  - sistemas já consolidados no mercado
  - boa desempenho
    - muitos anos de pesquisa e aprimoramento
    - eficiência: otimização de consultas, gerenciamento de transações
  - não atendem adequadamente os requisitos de dados de novas categorias de aplicações

Motivação (2/2)

- SGBDs Orientado a Objetos (SGBDOO)
  - modelo de dados mais rico
    - adequado ao mercado de aplicações não-convencionais
  - pior desempenho se comparado ao SGBDR, principalmente na capacidades de consulta e atualização
- SGBDs Objeto-Relacional (SGBDOR)
  - combina as melhores características do modelo de objetos no modelo relacional
    - modelo rico (OO) + consultas eficientes (Relacional)
  - exemplos: Oracle, Informix, DB2, Postgres

Classificação de *Stonebreaker*

- Classifica os principais sistemas gerenciadores de dados em 4 quadrantes



BDR x BDOO x BDOR

Critério	BDR	BDOO	BDOR
padrão	SQL-2	ODMG 3.0	SQL-3
suporte a dados complexos	não	sim	sim
performance	alta	baixa	espera-se que seja alta
maturidade	maduro	razoavelmente maduro	razoavelmente novo
uso de SQL	SQL *full*	OQL (em geral, não é *full*)	SQL estendido para objetos
vantagem	eficiência de acesso	modelo de dados rico	modelo rico + eficiência de acesso
uso comercial	larga escala	pequena escala	Tende a alcançar larga escala

Características Objeto-Relacional do SQL

- A SQL foi especificada em 1970
- A SQL foi incrementada substancialmente em 1989 e 1992
- Um novo padrão aprovado em 1999, chamado SQL3, adicionou características de orientação a objetos

Suporte Objeto-Relacional na SQL-99

- Construtores de tipo para especificar objetos complexo
- Mecanismo para especificar identidade de objeto
- Mecanismo para encapsulamento de operações
- Mecanismo para suporte de herança
  - Especificação de especialização e generalização

Construtores de Tipo (1)

- Conhecido como **tipos definidos pelo usuário (TDUs)**
- Sintaxe para a definição de um tipo
  - **CREATE ROW TYPE nome_do_tipo_row AS (<declarações de componentes>)**
- Exemplo:

```
CREATE ROW TYPE tipo_Endereco AS (
        rua VARCHAR (45),
        cidade VARCHAR (25),
        cep CHAR (8));
```

Construtores de Tipo (2)

- Row types definem tipos para tuplas, e eles podem ser aninhados
- Exemplo:

```
CREATE ROW TYPE tipo_Companhia AS (
      comp_id REF(tipo_Companhia),
      comp_nome VARCHAR (2),
      endereco Tipo_Endereco
);
CREATE ROW TYPE Tipo_Endereco(
      ruaNro          VARCHAR(60),
      cidade          VARCHAR(40),
      CEP             INTEGER
);
```


Criação de Tabelas

- Um tipo definido pelo usuário pode também ser usado para especificar os tipos row de uma tabela:

```
CREATE TABLE Companhia OF TYPE
tipo_Companhia
  (VALUES FOR comp_id ARE SYSTEM GENERATED);
```

- Sintaxe para especificar identificadores de objeto:

```
VALUES FOR <atributo_oid> ARE
<metodo_gerador_de_valores>
```

- Opções:
  - SYSTEM GENERATED
  - USER GENERATED

Exemplo de Especificação de OID

```
CREATE ROW TYPE TFornec(
   CodFornec     REF(TFornec),
   NomeFornec    VARCHAR(40),
   EndFornec     TEnd,
   StatusFornec  INTEGER
);

CREATE TABLE Fornec OF TYPE TFornec
  VALUES FOR CodFornec ARE SYSTEM GENERATED;
```

Especifica que a linha contém uma referência a uma linha do próprio tipo

Especifica que a referência dentro da linha deve apontar para a própria linha

Acesso a Atributos de Construtores de Tipo

- Notação de ponto (“ponto e ponto”) é usada para se referir a componentes de tipos
- Exemplo:

```
SELECT comp_id, endereco..ruaNro
FROM Companhia
WHERE endereco..cidade = 'Joinville'
```

Inserindo Valores em Atributos de Construtores

- Deve-se indicar os valores para todos os níveis de aninhamento
- Exemplo

```
INSERT INTO Companhia
  VALUES ('Companhia1', Tipo_Endereco('rua
  João Colin, 120', 'Joinville', 88000));
```


Atributos como Referências

- Um atributo componente de uma tupla pode ser uma referência a outro construtor de tipo:

```
CREATE ROW TYPE tipo_Empresa AS (
  empregado REF (tipo_Empregado) SCOPE (Empregado),
  companhia REF (tipo_Companhia) SCOPE (Companhia));
```

- A palavra-chave **SCOPE** especifica a tabela cujas tuplas podem ser referenciadas por um atributo referência

```
CREATE TABLE Empresa OF TYPE tipo_Empresa;
```

Acessando Atributos por Referência

- Exemplo

```
SELECT empregado->emp_nome
FROM Empresa
WHERE
  companhia->comp_nome = 'Companhia1';
```

indica uma referência a um OID e não a um atributo de um componente agregado

Inserindo Objetos de Referência

- Indicação dos valores de OIDs
- Exemplo

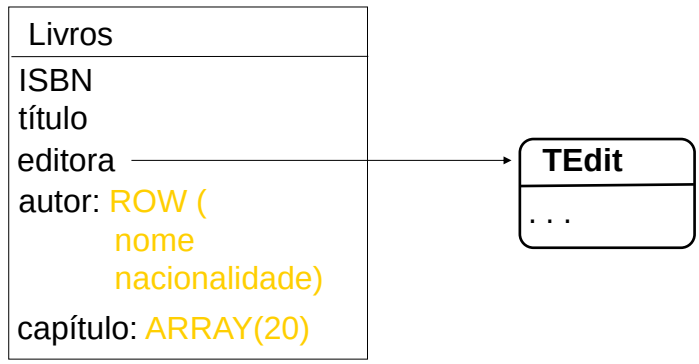
```
INSERT INTO Empresa
VALUES (REF(Emp01), REF(Comp02));
```

Definição de Objetos Complexos

- Novos tipos de dados
  - *Row* (tupla)
  - *Array* (coleção ordenada)
    - arrays n-dimensionais não são permitidos
- Exemplo

```
CREATE TABLE Livros (
  ISBN          CHAR(10),
  título   VARCHAR(60) NOT NULL,
  editora  REF(TEdit),
  autor    ROW (nome      VARCHAR(50),
                nacionalidade  VARCHAR(15)),
  capítulo VARCHAR(20) ARRAY[20]
);
```


Modelagem de Objetos Complexos



Acesso a Objetos Complexos

```
insert into Livros
values ('65893/186-9', 'Banco de Dados
  Objeto-Relacional', REF('Campus'), ('João
  Souza', 'brasileira'), ARRAY['Introdução',
  'OO', 'BD Objeto-Relacional',
  'Conclusão']);

select capitulos[1]
from Livros
where autor..nome = 'João Souza'
```


Tipo Abstrato de Dados (TAD)

- Pode-se criar um tipo abstrato de dados com seus próprios métodos em adição aos atributos
- Permite herança de um tipo para um subtipo
- Definição

```
CREATE TYPE <nomeTAD> (
   <listaAtributos>
   [<declaraçãoDeMétodosDeComparação>]
   [<declaraçãoDeOutrosMétodos>] )
[INSTANTIABLE]   ⟵ pode gerar tabelas
[[NOT] FINAL]    ⟵ pode ou não ser especializado
```

TAD - Exemplo

```
CREATE TYPE TEmpregado (
          RG              INTEGER,
          nome            VARCHAR(40),
          endereço        Tend,
          gerente         REF(TEmpregado),
          salárioBase     DECIMAL (7,2),
          comissão        DECIMAL (7,2),

         METHOD salário() RETURNS DECIMAL (7,2);
       ... )
     INSTANTIABLE
     NOT FINAL;

   CREATE TABLE Empregados OF TYPE TEmpregado;
```


Sintaxe de declaração de métodos

- Sintaxe:

  METHOD <nome> (<lista_de_args>) RETURNS <tipo>;

- Exemplo

```
CREATE TYPE tipo_Endereco AS (
    rua VARCHAR (45),
    cidade VARCHAR (25),
    cep CHAR (5)
)
METHOD nro_apto ( ) RETURNS CHAR(8);
```

- A implementação do método pode ser especificada em um arquivo separado

```
METHOD
  CREATE FUNCTION nro_apto() RETURNS CHAR(8)
  FOR tipo_Endereco AS
  EXTERNAL NAME '/x/nro_apto.class' LANGUAGE 'java';
```

Implementação de método

- Exemplo

```
CREATE METHOD salário()
   FOR Tempregado
   RETURN REAL
   BEGIN
        RETURN salarioBase + comissão*0.8;
   END
```

- Consultas SQL podem invocar métodos ou funções

```
SELECT RG,nome
FROM empregados
WHERE salário() > 1000.00;
```


Herança em SQL

- **Herança** é especificada através da palavra-chave **UNDER**
- Exemplo

```
CREATE TYPE tipo_Gerente UNDER
    tipo_Empregado AS (
    depto_gerenciado CHAR (20)
);
```

- `tipo_Gerente` **herda** todas as características de `tipo_Empregado`
  - e ela tem um atributo adicional chamado `depto_gerenciado`

Projeto Lógico de BDOR (1/2)

Esquema EER	Esquema OR
entidade	tabela (pode-se definir adicionalmente um TAD ou **um construtor de tipo (ROW)** para cada entidade, para posterior reuso da definição)
entidade fraca	• **atributo com domínio ROW** OU • atributo de referência fraca -> forte
relacionamento 1:1	• fusão de entidades em uma tabela OU • **referências entre tabelas usando SCOPE**
relacionamento 1:N	•**atributo de referência (usando SCOPE) na tabela correspondente ao lado N** OU •atributo com domínio ARRAY do lado 1

Projeto Lógico de BDOR (2/2)

Esquema EER	Esquema OR
relacionamento M:N	• tabela de relacionamento OU • **atributo(s) com domínio(s) ARRAY**
atributo monovalorado	atributo atômico
atributo composto	atributo com domínio ROW
atributo multivalorado	atributo com domínio ARRAY
especialização	hierarquia de herança entre tipos ou tabelas
entidade associativa	mesmas recomendações para mapeamento de relacionamentos binários