Banco de Dados

Material de Apoio

Parte XIV– Banco de Dados Objeto Relacional

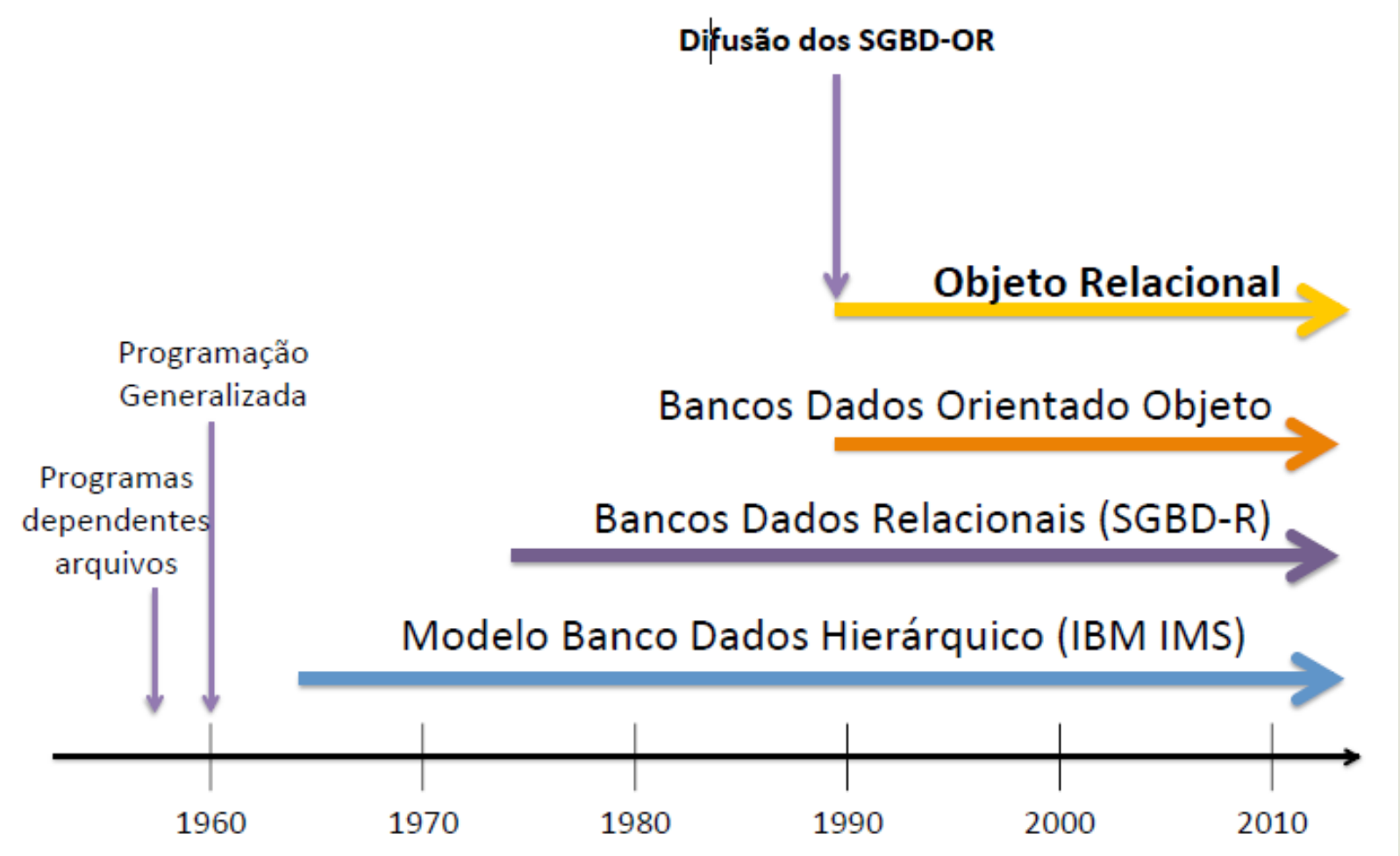
Prof. José Osvano da Silva, PMP, PSM I
joseosvano@unipac.br

1º sem / 2022

Sumário

- Introdução
- Visão geral dos conceitos de banco de dados de objeto
- Identidade de objeto e objetos *versus* literais
- Estruturas de tipo complexas para objetos e literais
- Encapsulamento de operações e persistência de objetos
- Encapsulamento de operações
- Persistência de objetos
- Hierarquias de tipo e herança
- Outros conceitos de orientação a objeto
- Recursos objeto-relacional: extensões do banco de dados de objeto para SQL
- Tipos definidos pelo usuário e estruturas complexas para objetos
- Identificadores de objeto usando tipos de referência

Introdução


Difusão dos SGBD-OR
Objeto Relacional
Programação Generalizada
Bancos Dados Orientado Objeto
Programas dependentes arquivos
Bancos Dados Relacionais (SGBD-R)
Modelo Banco Dados Hierárquico (IBM IMS)
1960
1970
1980
1990
2000
2010

Introdução

- **Banco de dados de objeto (BDO)**
  - **Sistemas de gerenciamento de dados de objeto (SGDO)**
  - Foram propostos para atender a algumas das necessidades de aplicações mais complexas
  - Especifica:
    - A estrutura dos objetos complexos
    - As operações que podem ser aplicadas a esses objetos

Visão geral dos conceitos de banco de dados de objeto

- Introdução aos conceitos e recursos orientados a objeto
  - Origens nas linguagens de programação OO
  - Um objeto possui dois componentes:
    - Estado (valor) e comportamento (operações)
  - Variáveis de instância
    - Mantém os valores que definem o estado interno do objeto
  - Operação definida em duas partes:
    - Assinatura ou interface e implementação

Visão geral dos conceitos de banco de dados de objeto

- Herança
  - Permite a especificação de novos tipos ou classes que herdam grande parte de sua estrutura e/ou operações de tipos ou classes previamente definidas
- Sobrecarga de operador
  - Capacidade de uma operação de ser aplicada a diferentes tipos de objetos
  - Um nome de operação pode se referir a várias implementações distintas

Identidade de objeto e objetos *versus* literais

- Identidade única
  - Implementada por meio de um identificador de objeto (OID) único
  - **Imutável**
  - A maioria dos sistemas de banco de dados OO permite a representação de objetos e literais (ou valores)

Estruturas de tipo complexas para objetos e literais

- Estrutura de complexidade arbitrária
  - Contém todas as informações necessárias que descrevem o objeto ou literal
- Aninhamento de **construtores de tipos**
  - Tipo complexo pode ser construído com base em outros tipos
- Construtores mais básicos:
  - **Átomo:** contém tipos básicos (inteiro, real, etc.);
  - **Struct (ou tupla):** gerador de tipos;
  - **Coleção:** para aninhar estruturas mais complexas;

Estruturas de tipo complexas para objetos e literais

- Tipos de coleção:
  - **Set**
  - **List**
  - **Bag**
  - **Array**
  - **Dictionary**
- **Linguagem de definição de objeto (ODL)**
  - Usada para definir os tipos de objeto para determinada aplicação de banco de dados.

ra objetos e literais

```
define type  FUNCIONARIO
tuple (
    Pnome:                  string;
    Minicial:               char;
    Unome:                  string;
    Cpf:                    string;
    Data_nascimento:        DATE;
    Endereco:               string;
    Sexo:                   char;
    Salario:                float;
    Supervisor:             FUNCIONARIO;
    Dep:                    DEPARTAMENTO;

define type  DATA
tuple (
    Ano:                    integer;
    Mes:                    integer;
    Dia:                    integer; );

define type  DEPARTAMENTO
tuple (
    Dnome:                  string;
    Dnumero:                integer;
    Ger:                    tuple (
                            Gerente: FUNCIONARIO;
                            Data_inicio: DATE; );
    Localizacoes:           set(string);
    Funcionarios:           set(FUNCIONARIO);
    Projetos:               set(PROJETO); );
```

Figura 11.1
Especificando os tipos de objeto FUNCIONARIO, DATA e DEPARTAMENTO usando construtores de tipo.

Encapsulamento de operações e persistência de objetos

- Encapsulamento
  - Relacionado aos conceitos de tipos de dados abstratos e ocultação de informações nas linguagens de programação
  - Define o comportamento de um tipo de objeto com base nas operações que podem ser aplicadas externamente
  - Os usuários externos só se tornam cientes da interface das operações
  - Divide a estrutura de um objeto em atributos visíveis e ocultos

Encapsulamento de operações

- **Construtor de objeto**
  - Usado para criar um objeto
- **Operação de destruição**
  - Usado para destruir (excluir) um objeto
- **Operações modificadoras**
  - Modificar os estados (valores) de vários atributos de um objeto
- **Recupera** informações sobre o objeto
- Uma operação costuma ser aplicada a um objeto usando a notação de ponto

Persistência de objetos

- **Objetos transientes**
  - Existem no programa em execução
  - Desaparecem quando o programa termina
- **Objetos persistentes**
  - Armazenados no banco de dados e persistem após o término do programa
  - **Mecanismo de nomeação**
  - **Acessibilidade**

```
define class SET_DEPARTAMENTO
    type set (DEPARTAMENTO);
    operations   adiciona_dep(d: DEPARTAMENTO):   boolean;
                (* acrescenta um departamento ao objeto SET_DEPARTAMENTO *)
                  remove_dep(d: DEPARTAMENTO): boolean;
                (* remove um departamento do objeto SET_DEPARTAMENTO *)
                  criar_set_dep: SET_DEPARTAMENTO;
                  destroi_set_dep: boolean;
end SET_DEPARTAMENTO;
...
persistent name TODOS_DEPARTAMENTOS: SET_DEPARTAMENTO;
(* TODOS_DEPARTAMENTOS é um objeto persistente nomeado do tipo SET_DEPARTA-
MENTO *)
...
d:= cria_dep;
(* cria um objeto DEPARTAMENTO na variável d *)
...
b:= TODOS_DEPARTAMENTOS.adiciona_dep(d);
(* torna d persistente incluindo-o no conjunto persistente TODOS_DEPARTAMENTOS *)
```

Figura 11.3
Criando objetos persistentes por nomeação e acessibilidade.

Hierarquias de tipo e herança

- Herança
  - Definição de novos tipos com base em outros predefinidos
  - **hierarquia de tipo** (ou **classe**)

- Tipo: **nome de tipo** e uma lista de **funções** visíveis (*públicas)*
  - Formato:
    - NOME_TIPO: função, função, ..., função

Hierarquias de tipo e herança

- **Subtipo**
  - Útil na criação de um novo tipo que é similar, mas não idêntico a um tipo já definido
  - Exemplo:
    - FUNCIONÁRIO subtype-of PESSOA: Salário, Data_contratação, Nível
    - ALUNO subtype-of PESSOA: Curso, Coeficiente

Hierarquias de tipo e herança

- **Extensões**
  - Coleção de objetos persistentes para cada tipo ou subtipo
  - Extensões são subconjuntos da extensão correspondente à classe OBJETO
- **Coleção persistente**
  - Armazenados permanentemente no banco de dados
- **Coleção transiente**
  - Existe por certo tempo durante a execução de um programa

Outros conceitos de orientação a objeto

- **Extensões**
  - Coleção de objetos persistentes para cada tipo ou subtipo
  - Extensões são subconjuntos da extensão correspondente à classe OBJETO
- **Coleção persistente**
  - Armazenados permanentemente no banco de dados
- **Coleção transiente**
  - Existe por certo tempo durante a execução de um programa

Outros conceitos de orientação a objeto

- **Herança múltipla**
  - Subtipo herda funções (atributos e métodos) de mais de um supertipo
- **Herança seletiva**
  - Subtipo herda apenas algumas das funções de um supertipo

Recursos objeto-relacional: extensões do banco de dados de objeto para SQL

- **Construtores de tipo**
  - Especifica objetos complexos
- Mecanismo para especificar a **identidade de objeto**
- **Encapsulamento de operações**
  - Fornecido por meio do mecanismo de tipos definidos pelo usuário (UDTs)
- Mecanismos de **herança**
  - Fornecidos usando palavra-chave UNDER

Tipos definidos pelo usuário e estruturas complexas para objetos

- **UDT** sintaxe:
  - CREATE TYPE NOME_TIPO AS
  - (<component declarations>);

- ROW TYPE
  - Criar diretamente um atributo estruturado usando a palavra-chave **ROW**

```
(a) CREATE TYPE TIPO_END_RUA AS (
        NUMERO              VARCHAR (5),
        NOME_RUA            VARCHAR (25),
        NR_APTO             VARCHAR (5),
        NR_BLOCO            VARCHAR (5)
    );
    CREATE TYPE TIPO_END_BRASIL AS (
        END_RUA             TIPO_END_RUA,
        CIDADE              VARCHAR (25),
        CEP                 VARCHAR (10)
    );
    CREATE TYPE TIPO_TELEFONE_BRASIL AS (
        TIPO_TELEFONE       VARCHAR (5),
        CODIGO_AREA         CHAR (3),
        NUM_TELEFONE        CHAR (7)
    );

(b) CREATE TYPE TIPO_PESSOA AS (
        NOME                VARCHAR (35),
        SEXO                CHAR,
        DATA_NASCIMENTO     DATE,
        TELEFONES           TIPO_TELEFONE_BRASIL ARRAY [4],
        END                 TIPO_END_BRASIL
    INSTANTIABLE
    NOT FINAL
    REF IS SYSTEM GENERATED
    INSTANCE METHOD IDADE( ) RETURNS INTEGER;
    CREATE INSTANCE METHOD IDADE( ) RETURNS INTEGER
        FOR TIPO_PESSOA
        BEGIN
            RETURN /* CÓDIGO PARA CALCULAR A IDADE DE UMA PESSOA COM BASE NA
                      DATA DE HOJE E SUA DATA_NASCIMENTO */
        END;
    );

(c) CREATE TYPE TIPO_NOTA AS (
        DISCIPLINA          CHAR (8),
        SEMESTRE            VARCHAR (8),
        ANO                 CHAR (4),
        NOTA                CHAR
    );
    CREATE TYPE TIPO_ALUNO UNDER TIPO_PESSOA AS (
        CODIGO_CURSO        CHAR (4),
        COD_ALUNO           CHAR (12),
        GRAU                VARCHAR (5),
        HISTORICO_ESCOLAR   TIPO_NOTA ARRAY [100]
```

(continua)

Figura 11.4
Ilustrando alguns dos recursos de objeto da SQL. (a) Usando UDTs como tipos para atributos como Endereco e Telefone. (b) Especificando UDT para TIPO_PESSOA. (c) Especificando UDTs para TIPO_ALUNO e TIPO_FUNCIONARIO como dois subtipos de TIPO_PESSOA.

```
    INSTANTIABLE
    NOT FINAL
    INSTANCE METHOD COEFICIENTE( ) RETURNS FLOAT;
    CREATE INSTANCE METHOD COEFICIENTE( ) RETURNS FLOAT
        FOR TIPO_ALUNO
        BEGIN
            RETURN /* CÓDIGO PARA CALCULAR COEFICIENTE MEDIO DE UM ALUNO COM BASE EM
                      SEU HISTORICO ESCOLAR */
        END;
    );
    CREATE TYPE TIPO_FUNCIONARIO UNDER TIPO_PESSOA AS (
        CODIGO_EMPREGO      CHAR (4),
        SALARIO             FLOAT,
        CPF                 CHAR (11)
    INSTANTIABLE
    NOT FINAL
    );
    CREATE TYPE TIPO_GERENTE UNDER TIPO_FUNCIONARIO AS (
        DEP_GERENCIADO      CHAR (20)
    INSTANTIABLE
    );

(d) CREATE TABLE PESSOA OF TIPO_PESSOA
        REF IS ID_PESSOA SYSTEM GENERATED;
    CREATE TABLE FUNCIONARIO OF TIPO_FUNCIONARIO
        UNDER PESSOA;
    CREATE TABLE GERENTE OF TIPO_GERENTE
        UNDER FUNCIONARIO;
    CREATE TABLE ALUNO OF TIPO_ALUNO
        UNDER PESSOA;

(e) CREATE TYPE TIPO_EMPRESA AS (
        NOME_EMP            VARCHAR (20),
        LOCALIZACAO         VARCHAR (20));
    CREATE TYPE TIPO_EMPREGO AS (
        Funcionario REF (TIPO_FUNCIONARIO) SCOPE (FUNCIONARIO),
        Empresa REF (TIPO_EMPRESA) SCOPE (EMPRESA) );
    CREATE TABLE EMPRESA OF TIPO_EMPRESA (
        REF IS COD_EMP SYSTEM GENERATED,
        PRIMARY KEY (NOME_EMP) );
    CREATE TABLE EMPREGO OF TIPO_EMPREGO;
```

Figura 11.4 (continuação)
Ilustrando alguns dos recursos de objeto da SQL. (c) (continuação) Especificando UDTs para TIPO_ALUNO e TIPO_FUNCIONARIO como dois subtipos de TIPO_PESSOA. (d) Criando tabelas com base em alguns dos UDTs, e ilustrando a herança de tabela. (e) Especificando relacionamentos com REF e SCOPE.

Tipos definidos pelo usuário e estruturas complexas para objetos

- Tipo de array
  - Elementos de referência utilizando [ ]

- Função **CARDINALITY**
  - Retorna o número atual de elementos em um array

Identificadores de objeto usando tipos de referência

- **Tipo de referência**
  - Cria identificadores de sistema gerados pelo sistema
  - Exemplos:
    - REF IS SYSTEM GENERATED
    - REF IS <OID_ATRIBUTO>
    - <METODO_GERACAO_VALOR> ;

Criando tabelas baseadas nos UDTs

- **INSTANTIABLE**
  - Especifica que a UDT é instanciável
  - Faz com que uma ou mais tabelas sejam criadas

Encapsulamento de operações

- Tipo definido pelo usuário
  - Especifica métodos (ou operações) além dos atributos
  - Formato:

CREATE TYPE <NOME-TIPO> (

<LISTA DE ATRIBUTOS DE COMPONENTE E SEUS TIPOS>

<DECLARACAO DE FUNcoES (METODOS)>

);

Encapsulamento de operações

- Dada uma UDT **TYPE_T( ),** o SQL oferece:
  - Função construtora;
  - Função observadora;
  - Função alteradora;
  - Outras funções
    - DECLARE EXTERNAL <NOME_FUNCAO>
    - <ASSINATURA>
    - LANGUAGE <NOME_LINGUAGEM>;

Especificando herança e sobrecarga de funções

- Regras de herança:
  - Todos os atributos são herdados
  - A ordem dos supertipos na cláusula UNDER determina a hierarquia de herança
  - Uma instância de um subtipo pode ser usada em cada contexto em que uma instância do supertipo é utilizada
  - Um subtipo pode redefinir qualquer função que é definida em seu supertipo
  - Quando uma função é chamada, a melhor combinação é selecionada com base nos tipos de todos os argumentos
  - Para a ligação dinâmica, a execução dos tipos de parâmetros são considerados

Especificando relacionamentos por referência

- Um atributo componente de uma tupla pode ser uma
  - **referência** a uma tupla de outra tabela
    - Especificada usando a palavra-chave **REF**

- Palavra-chave **SCOPE**
  - Especifica o nome da tabela cujas tuplas podem ser referenciadas

Especificando relacionamentos por referência

- **Notação de ponto**
  - Montar expressões de caminho

- **–>**
  - Usado para desreferência

O modelo de objeto ODMG e a Object Definition Language (ODL)

- Modelo de objeto ODMG
  - Modelo de dados para **linguagem de definição de objeto (ODL)** e **linguagem de consulta de objeto (OQL)**

- Objetos e Literais
  - Blocos básicos de montagem do modelo de objeto

O modelo de objeto ODMG e a Object Definition Language (ODL)

- Um objeto tem cinco aspectos:
  - Identificador, nome, tempo criação de vida, estrutura e criação.
- **Literal**
  - Valor que não tem um identificador de objeto
- **Comportamento** refere-se às operações
- **Estado** refere-se às propriedades

O modelo de objeto ODMG e a Object Definition Language (ODL)

- **Interface**
  - Especifica apenas o comportamento de um tipo de objeto
  - Normalmente é **não-instanciável**
- **Class**
  - Especifica tanto o estado (atributos) quanto o comportamento (operações) de um tipo de objeto
  - **Instanciável**

Herança no modelo de objeto de ODMG

- **Herança de comportamento**
  - Também é conhecida como herança ISA ou de interface
  - Especificada pela notação de dois pontos (:)
- **Herança extendida**
  - Especificada pela palavra-chave **extends**
  - Herda estado e comportamento estritamente entre classes
  - A herança múltipla por extends não é permitida

Interfaces e classes embutidas no modelo de objeto

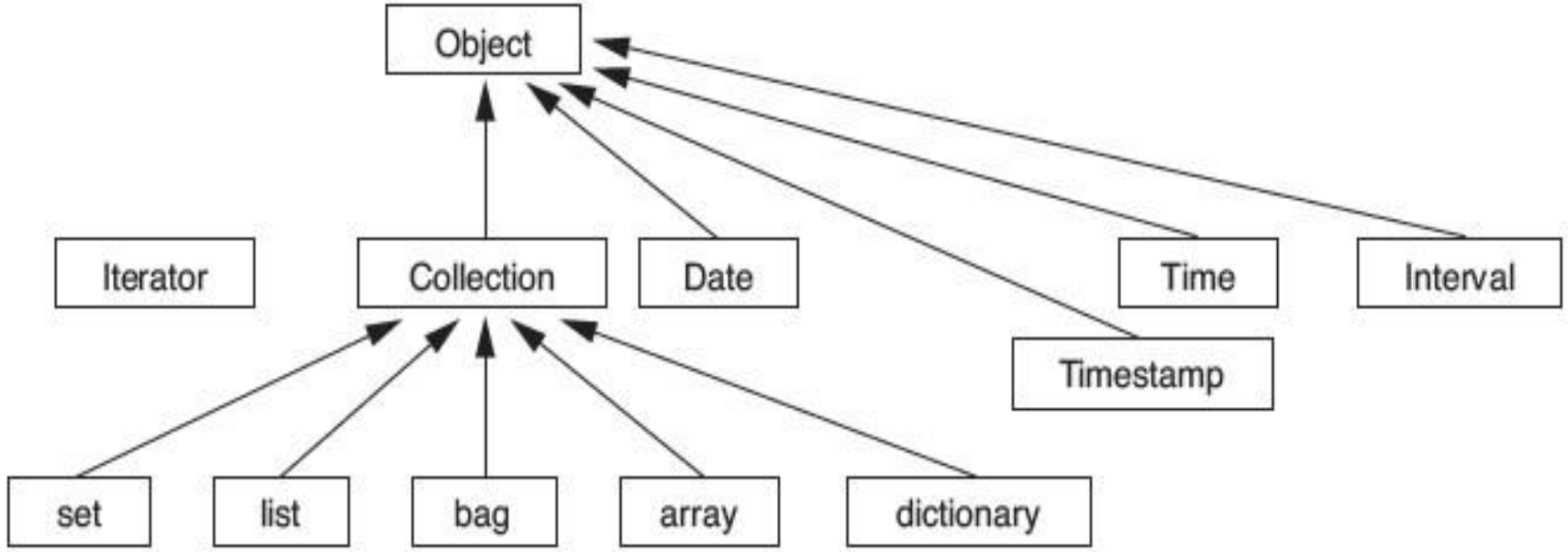


Figura 11.6

Hierarquia de herança para as interfaces embutidas do modelo de objeto.

Interfaces e classes embutidas no modelo de objeto

- **Objetos de coleção**
  - Herdam a interface Collection básica

- I = O.create_iterator()
  - Cria um objeto de iteração para o objeto de coleção

- Objetos Collection são especializados ainda mais em :
  - set, list, bag, arraye dictionary

Extensões, chaves e fábrica de objetos

- **Extensões**
  - Tem todos os objetos persistentes dessa classe
- **Chave**
  - Uma ou mais propriedades cujos valores são restritos a serem únicos para cada objeto na extensão
- **Fábrica de objeto**
  - Usado para gerar ou criar objetos individuais por meio de suas operações

A li

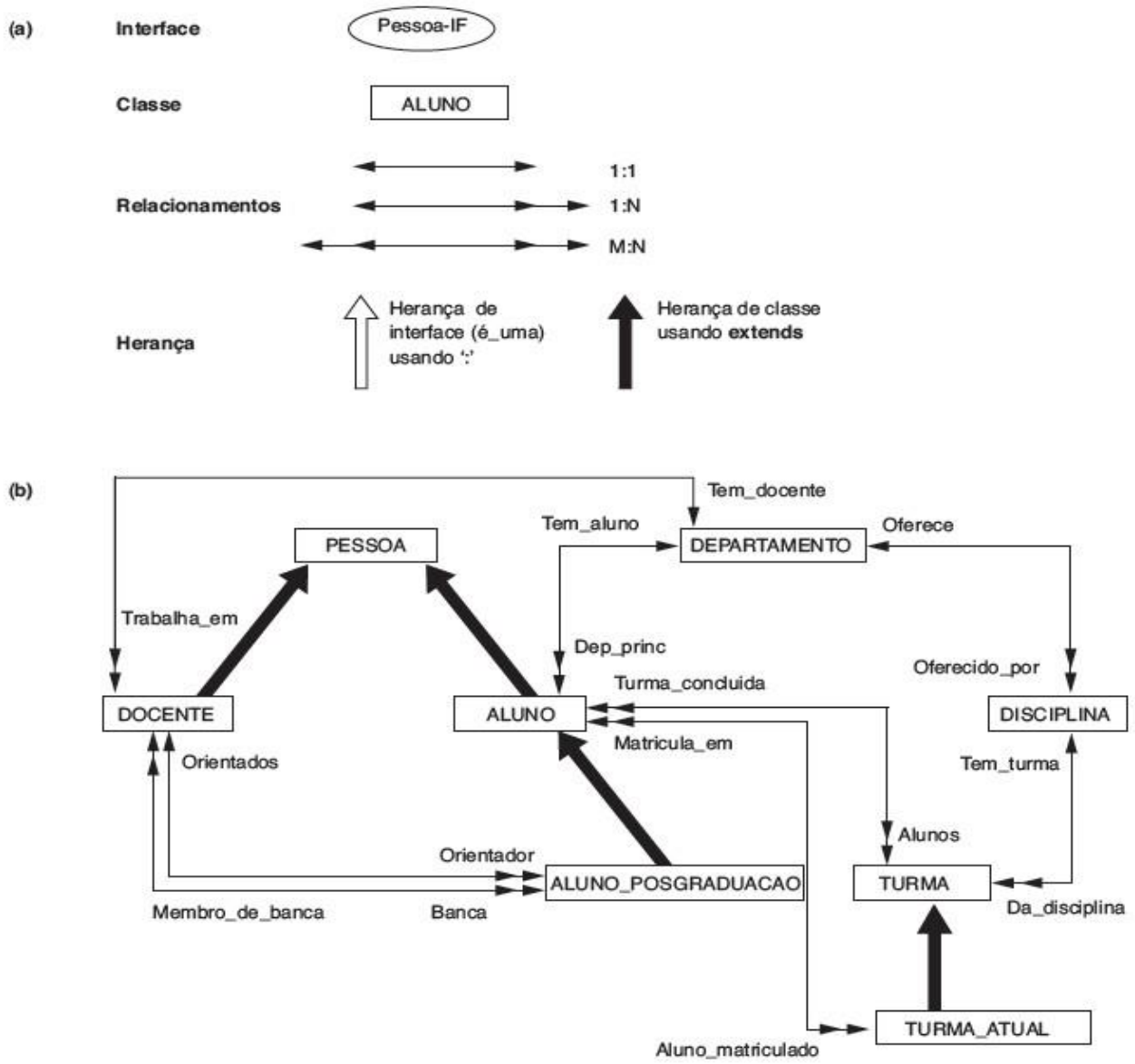


Figura 11.9

Exemplo de um esquema de banco de dados. (a) Notação gráfica para representar esquemas ODL. (b) Um esquema gráfico de banco de dados de objeto para parte do banco de dados UNIVERSIDADE (as classes NOTA e TITULO_ACADEMICO não aparecem).

A lingua

```
class PESSOA
(       extent          PESSOAS
        key             Cpf )
{       attribute       struct Projnome {       string          Pnome,
                                                string          Mnome,
                                                string          Unome }         Nome;
        attribute       string                                                  Cpf;
        attribute       date                                                    Data_nascimento;
        attribute       enum Genero{M, F}                                       Sexo;
        attribute       struct Endereco {       short           Nr,
                                                string          Rua,
                                                short           Nr_apto,
                                                string          Cidade,
                                                string          Estado,
                                                short           Cep }           Endereco;
        short           Idade( ); };
class DOCENTE extends PESSOA
(       extent          DOCENTE )
{       attribute       string                  Nivel;
        attribute       float                   Salario;
        attribute       string                  Escritorio;
        attribute       string                  Telefone;
        relationship    DEPARTAMENTO            Trabalha_em inverse DEPARTAMENTO::Tem_docente;
        relationship    set<ALUNO_POSGRADUACAO> Orientados inverse ALUNO_
                        POSGRADUACAO::Orientador;
        relationship    set<ALUNO_POSGRADUACAO> Membro_de_banca inverse ALUNO_
                        POSGRADUACAO::Banca;
        void            dar_aumento(in float aumento);
        void            promocao(in string novo_nivel);    };
class NOTA
(       extent          NOTAS )
{
        attribute       enum Valores Nota{A,B,C,D,F,I,P} Nota;
        relationship    TURMA Turma inverse TURMA::Alunos;
        relationship    ALUNO Aluno inverse ALUNO::Turma_concluida;};
class ALUNO extends PESSOA
(       extent                          ALUNOS )
{       attribute string                Tipo_aluno;
        attribute       DEPARTAMENTO    Dep_secund;
        relationship    DEPARTAMENTO Dep_prino inverse DEPARTAMENTO::Tem_aluno;
        relationship    set<NOTA> Turma_concluida inverse NOTA::Aluno;
        relationship    set<TURMA_ATUAL> Matricula_em inverse TURMA_ATUAL::Aluno_matriculado;
        void            troca_dep_prino(in string dnome) raises(departamento_invalido);
        float           coeficiente( );
        void            matricula(in short nr_turma) raises(turma_invalida);
        void            aloca_nota(in short nr_turma; IN ValorNota nota)
                        raises(turma_invalida,nota_invalida); };
class TITULO_ACADEMICO
```

Figura 11.10
Esquema ODL possível para o banco de dados UNIVERSIDADE da Figura 11.8(b).

Projeto conceitual de banco de dados de objeto

- Diferenças entre o projeto conceitual do BDO e do BDR, é a manipulação de:
  - Relacionamentos
  - Herança
- Diferença filosófica entre o modelo relacional e o modelo de objeto dos dados
  - Em relação à especificação comportamental

Mapeando um esquema EER para um esquema BDO

1 - Crie uma classe ODL para cada tipo de entidade EER

2 - Inclua propriedades de relacionamento para cada relacionamento binário

3 - Inclua operações apropriadas para cada classe

4 - Uma classe ODL que corresponde a uma subclasse no esquema EER

- – Herda o tipo e os métodos de sua superclasse no esquema ODL

Mapeando um esquema EER para um esquema BDO

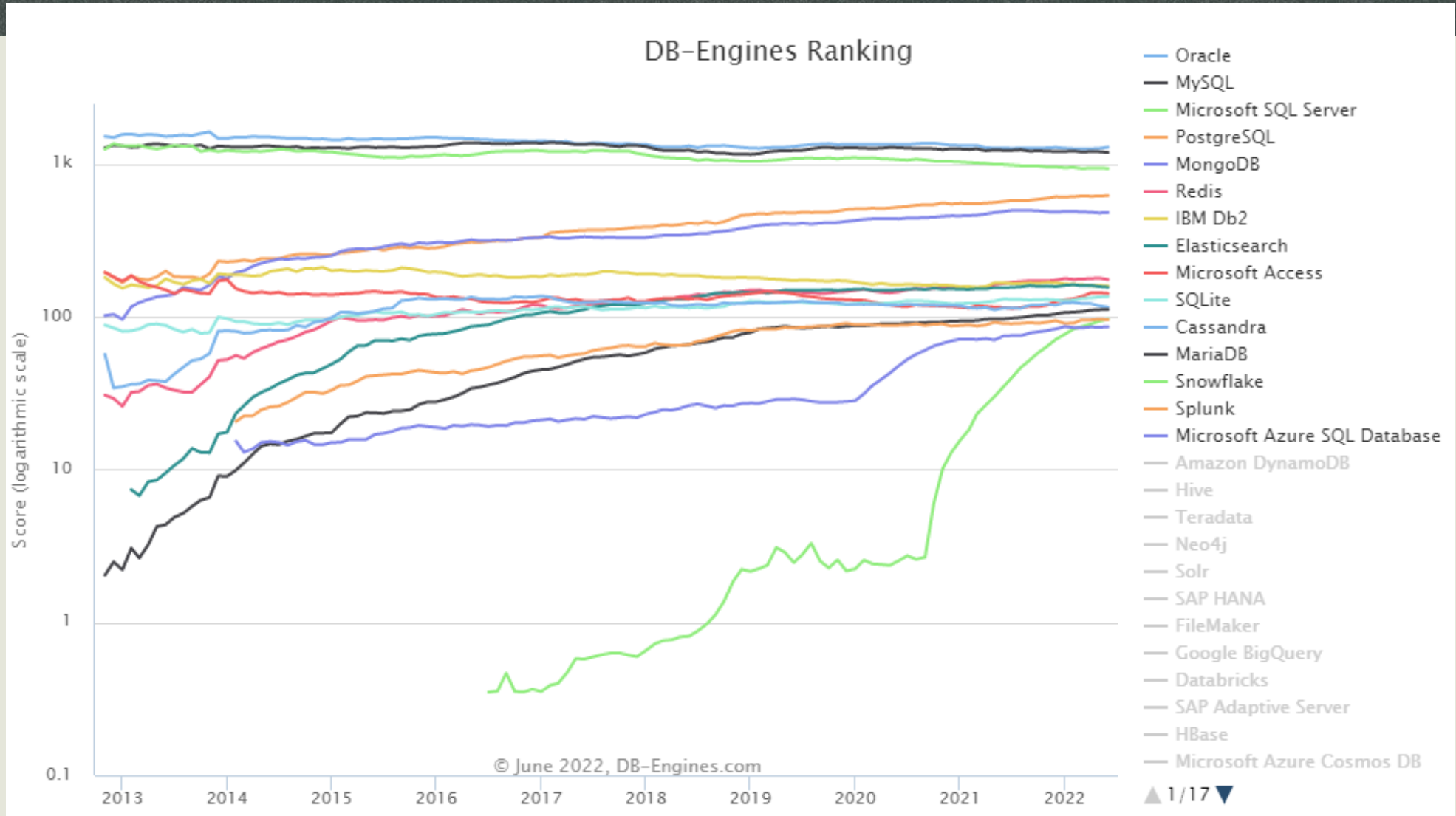
5 - Tipos de entidade fraca

- Mapeados da mesma maneira que os tipos de entidade regulares
- Categorias (tipos de união)
  - Difíceis de mapear para ODL
- Um n-ário com grau n > 2
  - Mapeado para uma classe separada, com referências apropriadas a cada classe participante

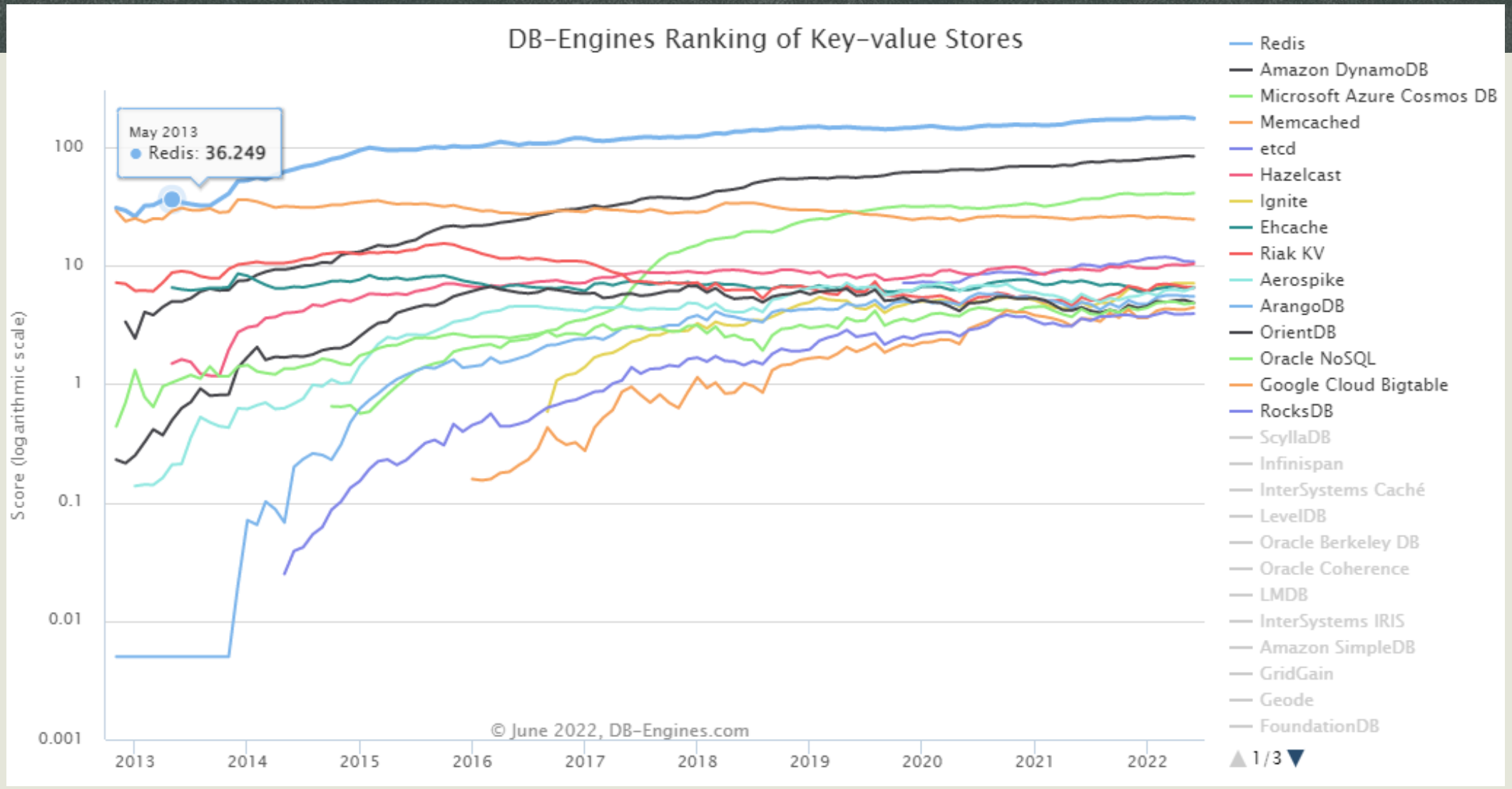
Exemplo de BDO

Tipo	Exemplos notáveis deste tipo
Cache de Chave-Valor	Apache Ignite, Coherence, eXtreme Scale, Hazelcast, Infinispan, Memcached, Velocity
Armazenamento Chave-Valor	ArangoDB, Aerospike
Armazenamento Chave-Valor (Eventualmente-Consistente)	Oracle NoSQL Database, Dynamo, Riak, Voldemort
Armazenamento Chave-Valor (Ordenado)	FoundationDB, InfinityDB, LMDB, MemcacheDB
Servidor de Estruturas de Dados	Redis
Armazenamento de Tuplas	Apache River, GigaSpaces
Banco de dados de Objeto	Objectivity/DB, Perst, ZopeDB
Armazenamento de Documentos	ArangoDB, BaseX, Clusterpoint, Couchbase, CouchDB, DocumentDB, IBM Domino, MarkLogic, MongoDB, Qizx, RethinkDB
Armazenamento de coluna ampla	Amazon DynamoDB, Bigtable, Cassandra, Druid, HBase, Hypertable, Firebase, CosmosDB

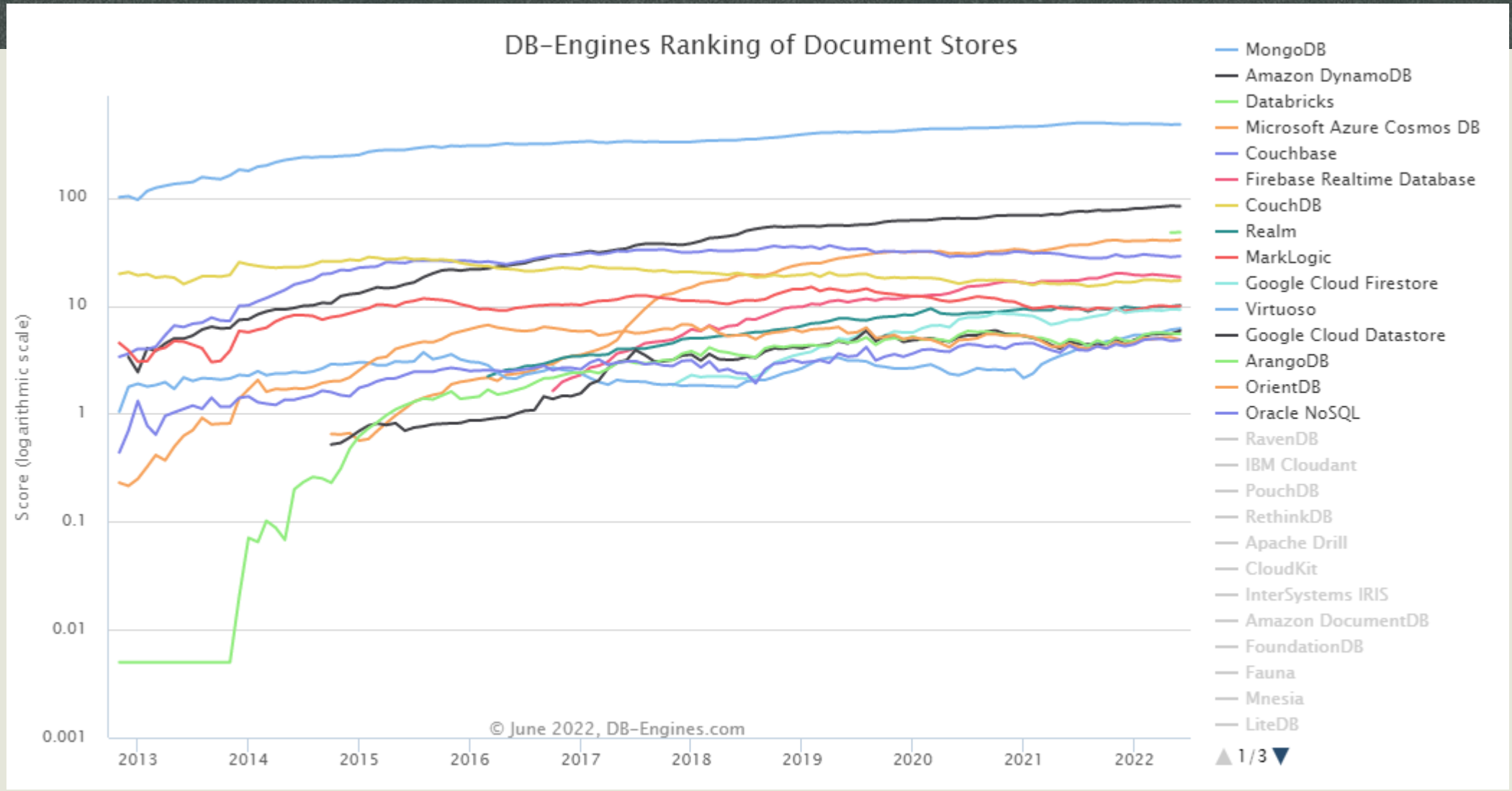
Uso de BD


DB-Engines Ranking
Score (logarithmic scale)
1k
100
10
1
0.1
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
© June 2022, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
Redis
IBM Db2
Elasticsearch
Microsoft Access
SQLite
Cassandra
MariaDB
Snowflake
Splunk
Microsoft Azure SQL Database
Amazon DynamoDB
Hive
Teradata
Neo4j
Solr
SAP HANA
FileMaker
Google BigQuery
Databricks
SAP Adaptive Server
HBase
Microsoft Azure Cosmos DB
1/17

Uso de BD Chave e Valor



Uso de BD Orientado a Documento



Referências

- historical trend of document stores popularity (db-engines.com)

Dúvidas

José Osvano da Silva
joseosvano@unipac.br