

# Faculdade de Engenharia da Universidade do Porto

Licenciatura em Engenharia Informática e Computação



Universidade do Porto

---

Faculdade de Engenharia

# FEUP

## Tecnologias de Bases de Dados

Base de Dados Orientada por Objectos

Outubro 2004

Nelson Rodrigues

nelson@fe.up.pt

## Introdução

Este relatório serve para documentar o desenvolvimento de um modelo de dados baseado na norma ODMG para um novo tipo de agenda electrónica.

A principal característica da agenda é a integração de todas as aplicações que a constituem.

A agenda irá conter quatro aplicações: Contactos, Calendário, Tarefas e Memorandos. Toda a informação contida na agenda pode estar relacionada com qualquer outro pedaço de informação, mesmo que seja pertencente a outra aplicação. Deverá ainda ser possível classificar a informação existente segundo um sistema de categorias hierárquico.

## Modelo ODMG

Abaixo apresenta-se o modelo de dados desenvolvido, primeiro com um modelo de classes UML e seguidamente de acordo com a norma ODMG.

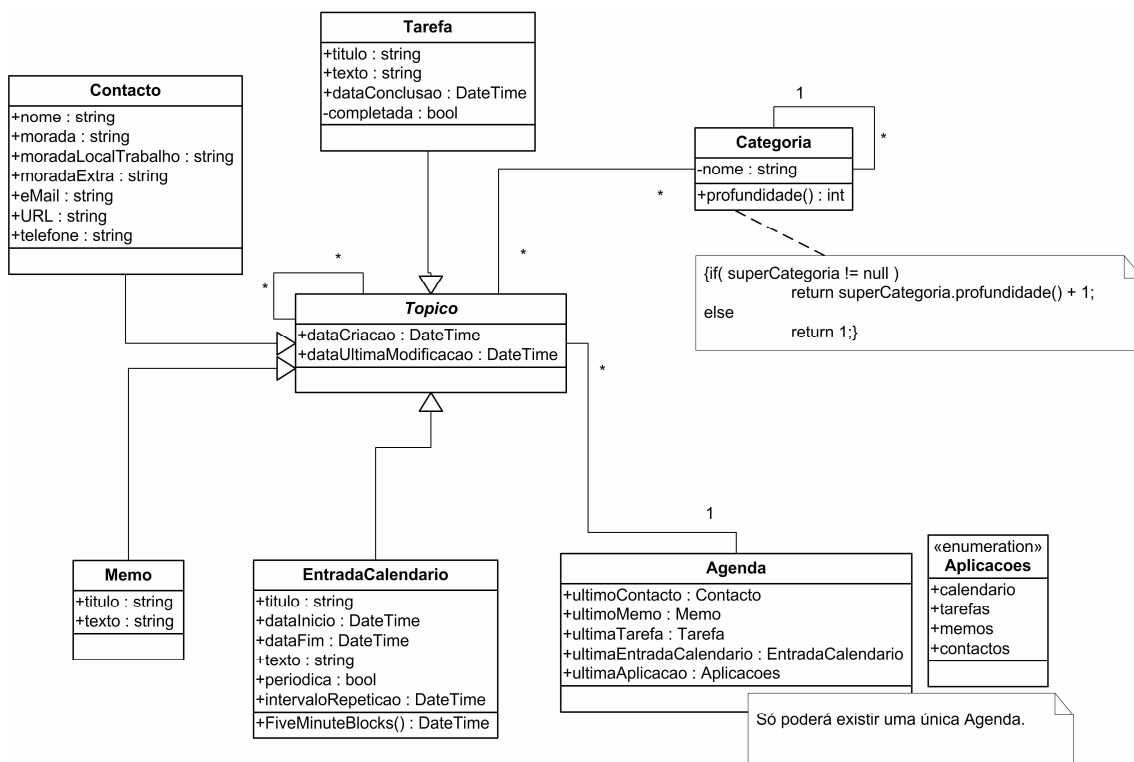


Figura 1 – Esquema UML do modelo de dados.

```

interface Categoria
(extent Categorias key (nome))
{
    attribute string nome;

    relationship Set<Topico> topicos inverse Topico::categorias;
    relationship Set<Categoria> subCategorias inverse
        Categoria::superCategoria;
    relationship superCategoria inverse Categoria::subCategorias;

    int profundidade();
}
  
```

```
interface Topico
(extent Topicos)
{
    attribute DateTime dataCriacao;
    attribute DateTime dataUltimaModificacao;

    relationship Set<Categoria> categorias inverse
        Categoria::topicos;

    relationship Set<Topico> topicosRelacionados inverse
        Topico::topicosQueReferem;
    relationship Set<Topico> topicosQueReferem inverse
        Topico::topicosRelacionados;
}

interface Contacto : Topico
(extent Contactos key (nome, eMail))
{
    attribute string nome;
    attribute string eMail;
    attribute string morada;
    attribute string moradaLocalTrabalho;
    attribute string moradaExtra;
    attribute string URL;
    attribute List<string> telefones;
}

interface Tarefa : Topico
(extent Tarefas key (titulo))
{
    attribute string titulo;
    attribute string texto;
    attribute DateTime dataConclusao;
    attribute bool completada;
}

interface Memo : Topico
(extent Memos key (titulo))
{
    attribute string titulo;
    attribute string texto;
}

interface EntradaCalendario : Topico
(extent EntradasCalendario key (titulo, dataInicio, horaInicio))
{
    attribute string titulo;
    attribute DateTime dataInicio;
    attribute DateTime dataFim;
    attribute string texto;
    attribute bool periodica;
    attribute DateTime intervaloRepeticao;

    Set<DateTime> FiveMinuteBlocks();
}
}
```

```

interface Agenda
(extent Agendas)
{
    attribute enum Aplicacoes {calendario, tarefas, memos,
        contactos} ultimaAplicacao;

    relationship Tarefa ultimaTarefa;
    relationship Contacto ultimoContacto;
    relationship EntradaCalendario ultimaEntradaCalendario;
    relationship Memo ultimoMemo;

    relationship Set<Topico> topicosContidos;
}

```

## Perguntas OQL

- Quais os tópicos que têm referência a pelo menos três pessoas?

```

SELECT t.*
FROM Topicos as t
WHERE count(
    (SELECT * FROM Contactos)
    INTERSECTS
    (SELECT * FROM t.topicosReferenciados) ) >= 3;

```

- Qual o dia mais preenchido do mês, sendo que as sobreposições só contam uma vez?

```

SELECT max(x.ocupacao), x.dia
FROM (
    SELECT count(*) as ocupacao, d.dia
    FROM (
        SELECT DISTINCT *
        FROM flatten (
            SELECT ec.FiveMinutBlocks()
            FOM EntradasCalendario AS ec
        )
    ) AS d
    GROUP BY d.dia
) AS x;

```

- Quais os tópicos complexos que têm referências para tópicos de todas as aplicações?

```

SELECT t.*
FROM Topicos AS t
WHERE
    exists memo IN t.topicosReferenciados : memo IN Memos
AND
    exists tarefa IN t.topicosReferenciados : tarefa IN Tarefas
AND
    exists contacto IN t.topicosReferenciados : contacto IN Contactos
AND

```

```

exists entradaCalendario IN t.topicosReferenciados :
    entradaCalendario IN EntradasCalendario
AND
( t IN Contactos OR t IN EntradasCalendario );

```

## Problema da Profundidade

Como está evidenciado no modelo de dados desenvolvido, a classe Categoria possui um método que devolve a profundidade da categoria, este método seria posteriormente implementado utilizando uma linguagem de programação “normal”. Usando a interrogação que se segue é possível determinar a profundidade máxima da hierarquia de categorias.

```

SELECT max( c.profundidade() )
FROM Categorias AS c;

```

## Modelo Relacional

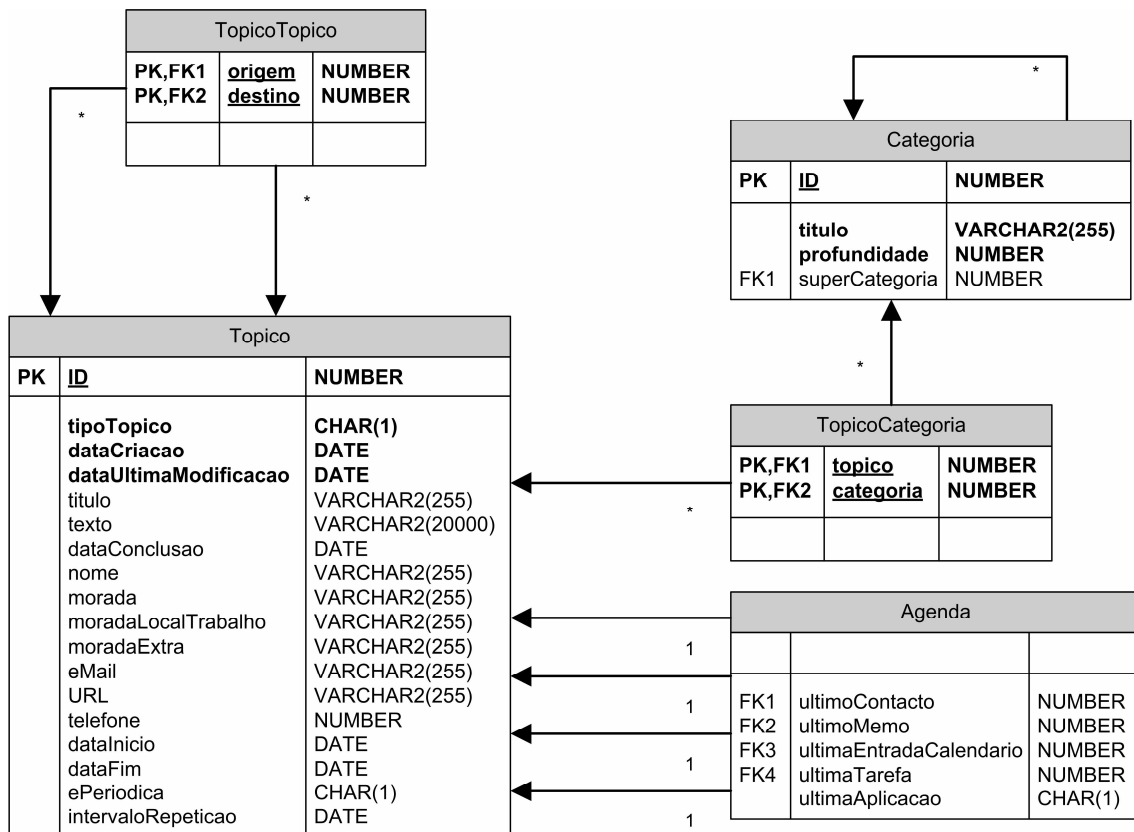


Figura 2 – Diagrama relacional do modelo de dados.

A principal diferença entre as duas formas de modelar o problema é a falta de um sistema de herança no modelo relacional. Sem a possibilidade de utilizar esta facilidade optou-se por utilizar uma única entidade para representar os quatro tipos de informação principais. Desta forma sempre que for inserido uma nova linha de informação uma grande parte dos campos ficará vazio. Outra desvantagem é o facto de não poderem existir restrições do tipo “not null” na maioria dos campos da entidade Tópico.

Outra grande diferença é o facto de não poderem existir métodos nas entidades, o que isto obriga é na entidade Categoria ter de existir um atributo que codifica a profundidade a que se encontra a categoria na hierarquia de categorias, este atributo terá de ser inserido pela interface da aplicação quando a categoria for criada. A solução encontrada no modelo orientado por objectos é muito mais elegante uma vez que encapsula melhor os dados e elimina da interface a responsabilidade da actualização do campo referido anteriormente.

## **Conclusões**

Ao efectuar este trabalho ficaram bem explicitadas as principais diferenças entre os dois tipos de modelação que foram utilizadas. Por um lado temos as bases de dados orientadas por objectos que se revelaram muito úteis para modelar problemas em que claramente temos um tipo de dados central, neste caso o Tópico, e vários outros tipos que descendem deste. Outra maneira de modelar este problema é usando um modelo relacional que para este problema específico não se revelou particularmente bom.

A linguagem de interrogações OQL foi também muito útil para fazer as perguntas pedidas, tendo ficado claro que se trata de uma linguagem bem mais poderosa que o SQL.

Foi um trabalho muito interessante e que serviu de sobremaneira para melhorar o meu conhecimento sobre bases de dados orientadas por objectos.