

MAC 5701 - Tópicos em Ciência da Computação

Profa. Yoshiko Wakabayashi

**Modelagem e Estruturação de Objetos Complexos
com Ênfase à Biologia Molecular Computacional**

Márcio Katsumi Oikawa e João Eduardo Ferreira
{koikawa, jef}@ime.usp.br

Novembro / 2001

Modelagem e Estruturação de Objetos Complexos com Ênfase à Biologia Molecular Computacional

Márcio Katsumi Oikawa
koikawa@ime.usp.br

João Eduardo Ferreira
jef@ime.usp.br

Sumário

1	Resumo	4
2	Introdução	4
3	Definição de Objetos Complexos	7
3.1	Abstração de Agregação	7
4	Formalização de Modelos	8
4.1	Funções	8
4.2	Objetos	9
4.3	Métodos	9
4.4	Tipos	10
4.5	Subtipos e Herança de Subtipos	12
4.6	Hierarquia de Agregação	13
5	Biologia Molecular	15
5.1	Usuários	16
5.2	Dados experimentais	16
5.3	Sequência	17
5.4	Características (Regiões Funcionais)	17
5.5	Referências Bibliográficas	18
5.6	Informações Adicionais	18
5.7	Estrutura dos Sub-domínios	19
6	Estágio de Pesquisa	21
7	Conclusão	22

Lista de Figuras

1	Distribuição lógica de uma cidade	5
2	Relação entre sub-domínios na representação de dados de Biologia Molecular	15

1 Resumo

O primeiro passo para representar dados em um modelo passa pelo processo de modelagem. Modelar dados não é tarefa simples, principalmente quando temos que trabalhar com estruturas não-convencionais, que mantêm, entre si, conexões com alto grau de complexidade. Neste trabalho, faremos um breve estudo sobre objetos complexos, seu tratamento, sua formalização e como eles podem ser usados para tratar conceitos de Biologia Molecular Computacional. Além disso, relacionaremos este estudo com uma possível proposta de trabalho de mestrado.

palavras-chave: objetos complexos, comunicação de objetos, formalização, Biologia Molecular Computacional

2 Introdução

Um dos motivos da grande disseminação da linguagem relacional para operações envolvendo bancos de dados é sua simplicidade sintática. Em linguagens relacionais, como SQL, dados são agrupados e tratados como conjuntos, não sendo necessário definir estruturas de dados complexas ou tipos compostos para seu tratamento. Além disso, operadores de controle de fluxo, operadores iterativos e definição de variáveis para armazenamento de resultados parciais não são necessários e sequer fazem parte da definição de tais linguagens [1].

Linguagens e operações relacionais funcionam bem para modelos de dados simples. Por este motivo, um banco relacional é capaz de trabalhar bem com grande parte das aplicações existentes no mundo real. Entretanto, com o passar do tempo, aplicações mais complexas foram surgindo, envolvendo tipos de dados mais complexos, de comportamento particular, dependendo da aplicação [2].

Infelizmente, o modelo relacional não é adequado para manipular bancos de dados cujos elementos possuem características complexas. Como a linguagem relacional trabalha com conjuntos, todos os dados são tratados uniformemente, como estruturas primitivas, não sendo possível tratar características individuais, como heranças, relações hierárquicas e sub-níveis de informação, dificultando, assim, a manipulação de dados com estruturas recursivas ou com alto grau de relacionamentos entre suas sub-estruturas.

Para ilustrar melhor, vamos apresentar um exemplo simples, apresentado em [3]. Considere um banco de dados que armazena o *layout* de uma cidade

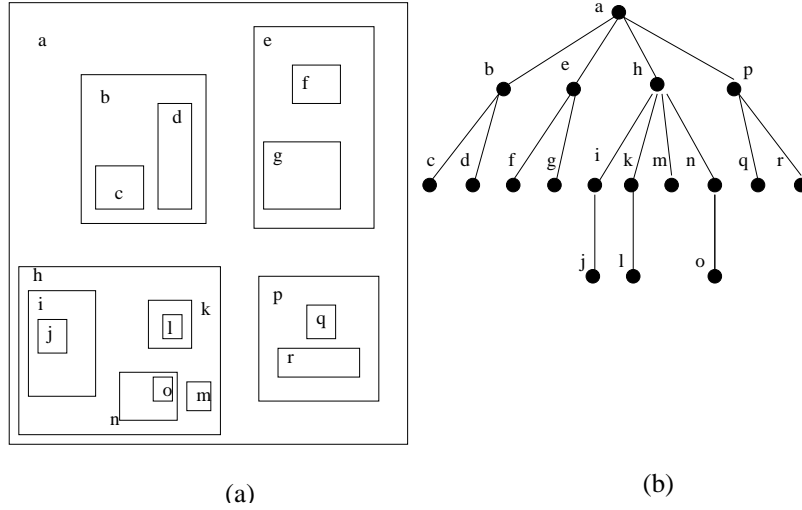


Figura 1: Distribuição lógica de uma cidade

qualquer, como a da figura 1. A cidade é dividida em subúrbios, os subúrbios em quadras, e as quadras em prédios. Uma maneira de representar cada uma destas áreas dentro do banco de dados poderia ser com as coordenadas superior esquerda e inferior direita de cada área, na forma:

$$area(id, x_1, x_2, y_1, y_2),$$

onde id é o identificador da área.

Uma consulta comum poderia ser:

Encontrar todas as áreas dentro de uma dada região, por exemplo, entre as regiões¹ $A(3, 1)$ e $B(5, 3)$.

Em SQL [4], esta consulta poderia ser escrita da seguinte forma:

```
select * from REGIAO2 where  $x_1 \geq 3$ 
and  $y_1 \geq 1$  and  $x_2 \leq 5$  and  $y_2 \leq 3$ 
```

Técnicas comuns de indexação nos gerenciadores relacionais utilizam árvores-B e tabelas *hash*. Neste caso, o algoritmo de busca consome um

¹Considere área como sendo um subúrbio, uma quadra ou um prédio

²A partir de agora, utilizaremos letras capitais para representar possíveis tabelas dentro da estrutura física do banco de dados

tempo $O(n)$, onde n é o número de registros [5] do tipo AREA no banco de dados. O máximo que poderíamos fazer é indexar um dos registros por umas das coordenadas, mas mesmo assim a eficiência do algoritmo não mudaria. Note que, neste caso específico, não estamos tirando proveito da estrutura hierárquica que existe entre os elementos dos tipos SUBURBIO, QUADRA e PREDIO.

Pensando um pouco na disposição hierárquica dos dados, podemos inferir que uma área u está contida numa área v se

$$x_{1u} \geq x_{1v} \wedge y_{1u} \geq y_{1v} \wedge x_{2u} \leq x_{2v} \wedge y_{2u} \leq y_{2v},$$

em outras palavras, existe um caminho de u para v na árvore da figura 1(b). Utilizando a relação acima, podemos dividir o processo de busca da consulta dos elementos contidos na área u (ver figura 1(a)) em duas fases:

- a) Busca de todos os elementos que contém a AREA u , ou seja, os “ancestrais” de u ;
- b) Busca de todos os elementos contidos em u , ou seja, os “descendentes” de u .

Neste exemplo simples, podemos perceber o quanto pode ser valioso (e muitas vezes necessário) fazer uso de relações hierárquicas em muitas operações envolvendo um banco de dados. O algoritmo de busca, neste caso, utiliza as coordenadas, como no modelo relacional, mas é capaz de rastrear uma estrutura de árvore, como na figura 1(b), diminuindo o tempo de execução consumido para $O(\log n)$ [5]. O segredo para a sensível melhora do desempenho nas consultas é o tratamento de cada item do banco de dados com um objeto, que conserva suas relações hierárquicas com outros objetos.

Nem todas as aplicações são tão simples e, na grande maioria das vezes, é difícil enxergar relações entre as várias classes de objetos. Além disso, é difícil formalizar uma linguagem que se adeque bem à maioria dos casos que envolvam alguma relação entre objetos.

Neste trabalho, estamos interessados em estudar um pouco mais das características particulares de objetos complexos, bem como as tentativas de formalização de linguagem para representar fielmente não somente a natureza destes, mas também as relações envolvidas.

3 Definição de Objetos Complexos

Definiremos *objetos* como instâncias de tipos abstratos de dados. Tipos abstratos de dados (TAD) são classes de elementos que modelam uma estrutura real em um ambiente computacional, encapsulando dados e operações, apresentando algumas propriedades, tais como:

1. O TAD exporta um tipo;
2. O TAD exporta um conjunto de operações (interface de tipo);
3. A interface de tipo é o único mecanismo de acesso a estruturas internas do TAD;
4. Axiomas e precondições definem o domínio de aplicação do TAD.

A modelagem orientada a objetos trata objetos e seus relacionamentos. Os dois tipos mais importantes de relacionamentos entre objetos são as relações “é-parte-de” e “está-associado-com” [6]. Relações do tipo “é-parte-de” modelam composições entre objetos, ou seja, objetos que são formados por objetos. Chamaremos *objetos complexos* estes conjuntos de objetos que mantêm, entre si, relações do tipo “é-parte-de”. Consideraremos, neste caso, que os objetos componentes podem ser objetos simples, compostos por atributos de tipos primitivos, ou mesmo outros objetos complexos.

3.1 Abstração de Agregação

Um conceito fortemente ligado ao de objeto complexo é o de agregação. Só faz sentido falar em objetos complexos se estamos trabalhando com objetos que mantêm, entre si, alguma relação de agregação, seja ela lógica ou física.

Agregação física diz respeito ao grau de agregação em nível de implementação. Objetos agregados fisicamente não mantêm, necessariamente, relações de dependência conceitual. Há casos onde temos objetos conceitualmente independentes, mas que, por conveniência, mantêm, entre si, uma relação de interdependência. Neste caso, falamos em *associação*.

Agregação lógica diz respeito à própria dependência conceitual que existe entre os tipos agregados, que podem ser tipos simples ou outros tipos complexos. Normalmente objetos possuem agregação lógica também possuem agregação física. Em termos de agregação lógica, podemos ter relações de agregação relacional, agregação referencial, especializações, agregações cíclicas e acíclicas [1].

Especializações representam um tipo especial de agregação. Na especialização, podemos verificar a existência de uma intersecção entre objetos e as operações destes. As especializações ocorrem quando o modelo exige separação de classes de objetos que mantêm, entre si, um mesmo grau de herança em relação a uma outra classe de objetos, ou possuem, entre si, uma forte relação conceitual.

4 Formalização de Modelos

O objetivo desta seção é fornecer conteúdo formal para definir objetos, relacionamentos e estruturas sobre as quais eles trabalham. A maioria das formalizações que serão apresentadas a seguir estão disponíveis em [1, 7, 8].

Todas as definições e provas que apresentaremos nesta seção são baseados nas seguintes premissas:

- (i) Existe uma coleção de domínios básicos finitos e não-vazios, representados por $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n, n \geq 1$;
- (ii) Existe um conjunto de identificadores contáveis e infinitos, chamado *Oid*;
- (iii) Estão disponíveis construtos, tais como listas, tuplas e conjuntos;
- (iv) Existe um conjunto de nomes contáveis e infinitos que podem ceder valores para nomear tipos e atributos (por exemplo), denominado \mathcal{N}

Todo $v \in \cup_{i=1,2,\dots,n} \mathcal{D}_i$ é um valor atômico, tal como todo elemento pertencente ao conjunto *Oid*. Todos os valores gerados a partir do construto *set* são chamados valores conjunto e todos os gerados pelo construto *tupla* são chamados valores estrutura.

4.1 Funções

1. Funções: f é uma função com domínio $S(f)$ e co-domínio $T(f)$ se, e somente se (sse) f é um conjunto de pares ordenados (x, y) e, para todo par $(x, y) \in f$ e $(x', y') \in f : x = x' \Rightarrow y = y'$;
2. Função sobre conjuntos: f é uma função sobre conjuntos sse $\forall x \in S(f) : f(x)$ é um conjunto;
3. Função sobre estruturas: f é uma função sobre estruturas sse $\forall x \in S(f) : f(x)$ é uma tupla;

4. Função sobre funções: f é uma função sobre funções sse $\forall x \in S(f) : f(x)$ é uma função;
5. Restrição de função: Seja f uma função e R um conjunto, tal que $R \subseteq S(f)$, então $f[R] =_{def} \{(x, y) | (x, y) \in f, x \in R\}$. Chamaremos $f[R]$ de *restrição de f sobre R*
6. Funções compatíveis: Sejam f e g funções. Falamos que f e g são funções compatíveis se $S(f) \cap S(g) \neq \emptyset$ e $f[S(f) \cap S(g)] = g[S(f) \cap S(g)]$
7. Composição de funções: Sejam f e g funções, tais que $T(f) \subseteq S(g)$ e $T(f) \neq \emptyset$, então $g \circ f =_{def} \{(x, g(f(x))) | x \in S(f) \text{ e } f(x) \in S(g)\}$. $g \circ f$ é chamada função composta.

4.2 Objetos

Dado um conjunto \mathcal{T} de tipos. Podemos definir um objeto como $o = (i, T, v)$, onde i é um identificador de objetos, T é um membro de \mathcal{T} e $v \in \mathcal{D}$. \mathcal{D} representa a união dos seguintes conjuntos de valores:

- (i) um valor *vazio* (null) é elemento de \mathcal{D} ;
- (ii) todo $v \in \cup_{i=1,2,\dots,n} \mathcal{D}_i$ é um elemento de \mathcal{D} ;
- (iii) todo elemento $i \in \mathcal{Oid}$, conjunto de identificadores, é elemento de \mathcal{D} ;
- (iv) $\{v | v \in \mathcal{D}\}$ é um elemento de \mathcal{D} ;
- (v) $(A_1 : v_1, A_2 : v_2, \dots, A_n : v_n)$, $v_i \in \mathcal{D}$ é um elemento de \mathcal{D} .

Objetos podem ser compostos por outros objetos através da relação “é-parte-de”, originando objetos complexos. As identidades de objetos complexos são tratados de forma independente de seus objetos constituintes. Na criação de nosso modelo, em particular, um objeto complexo é utilizado não somente como unidade de versão, controle de concorrência e autorização, mas também como uma unidade para realização de consultas, uma característica que necessita de extensões não-triviais no modelo convencional de buscas baseadas em álgebra relacional.

4.3 Métodos

Um método pode ser descrito por um cabeçalho e um corpo. O cabeçalho é composto por um nome, tipo dos objetos de entrada, número e tipos dos

parâmetros e o tipo do objeto de saída. O corpo do método descreve a semântica de seu algoritmo e sua implementação.

Podemos considerar um método como uma função que mapeia o produto dos domínios de sua entrada e seus parâmetros no domínio de sua saída.

Definição 4.3.1 (*método*):

Um método é descrito como uma tupla $\langle mn, f, mb \rangle$, onde $mn \in \mathcal{N}$ é o nome do método, f é uma função n -ária da forma

$$f : S \times P_1 \times \dots \times P_{n-1} \rightarrow T$$

tal que S é o argumento de entrada para o qual f é definido, P_i ($i = 0, \dots, n-1$ e $n > 0$) são os parâmetros do método e T é o parâmetro de saída.

A semântica da função f é definida pelo corpo do método, mb .

Dessa forma, dividimos um método em cabeçalho $\langle mn, f \rangle$ e corpo $\langle mb \rangle$.

Há casos em que a execução de um método produz, direta ou indiretamente, alterações de valor ou comportamento sobre outros objetos. Nestes casos, falamos em “efeito colateral”.

4.4 Tipos

Tipo é um conceito utilizado em um dado domínio para agrupar elementos de acordo com seu uso e comportamento. Um tipo denota uma estrutura e um domínio de todas as possíveis instâncias de elementos que obedecem a esta estrutura.

Definição 4.4.1 (*tipo*):

Um tipo T é descrito como uma tupla $(TN, Tfn, Struct, dom)$, onde $TN(T)$ representa o nome do tipo T , $Tfn(T)$ uma lista de funções em que T é o argumento de entrada, $Struct(T)$ uma estrutura para o tipo T , incluindo os supertipos imediatamente acima de T e $dom(T)$ o domínio das instâncias desta estrutura.

Recursivamente, podemos definir tipos como:

- (i) string, inteiro, real, booleano (verdadeiro ou falso) e data são tipos primitivos;

- (ii) se T_1, \dots, T_n são tipos e A_1, \dots, A_n são nomes distintos, então $(A_1 : T_1, \dots, A_n : T_n)$ também é um tipo, chamado *tipo tupla*;
- (iii) se T é um tipo, então $\{T\}$ também é um tipo, chamado *tipo conjunto*;
- (iv) se T é um tipo, então $\langle T \rangle$ também é um tipo, chamado *tipo lista*;
- (v) se T é um tipo, então $\mathbf{ref} T$ também é um tipo, chamado *tipo referência*;
- (vi) se S e T são tipos, então $f : S \rightarrow T$ também é um tipo, chamado *tipo função*.

Definição 4.4.2 (*Domínio de tipo*):

Seja \mathcal{U} o universo de todos os valores atômicos, isto é, $\mathcal{U} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \dots \cup \mathcal{D}_n \cup \text{Oid}$, e seja $\mathcal{P}(S)$ um conjunto potência sobre S . Para todo $R \in \mathcal{N}$, o domínio de R é denotado por $\text{dom}(R)$.

- (i) se R é um tipo básico, então $\text{dom}(R)$ é um conjunto de valores atômicos daquele tipo, ou seja, $\text{dom}(R) \subseteq \mathcal{U}$;
- (ii) se R é um tipo conjunto, dito $\{T\}$, ou tipo lista, dito $\langle T \rangle$, então $\text{dom}(R)$ é $\mathcal{P}(\text{dom}(T)) =_{\text{def}} \{v \mid v \subseteq \text{dom}(T)\}$;
- (iii) se R é um tipo tupla, dito $(A_1 : T_1, A_2 : T_2, \dots, A_n : T_n)$, então $\text{dom}(R) = \{(A_1 : o_1, \dots, A_n : o_n) \mid o_i \in \text{dom}(T_i), i = 1, \dots, n\}$;
- (iv) se R é um tipo referência, $\mathbf{ref} T$, então $\text{dom}(R) \subseteq \text{Oid}$ é um conjunto de objetos identidade, onde cada um destes referencia uma instância de um objeto do tipo T ;
- (v) se $R \in \mathcal{N}$ é um nome de função, e seu tipo de saída é T , então $\text{dom}(R) = \text{dom}(T)$.

Sejam \mathcal{O} um conjunto contável de objetos e \mathcal{T} um conjunto contável de tipos. Se, para cada $T \in \mathcal{T}$, existe um subconjunto X_t de \mathcal{O} associado, podemos definir (T, X_t) como classe de objetos implementando T . Dessa forma, X_t é um conjunto de objetos do tipo T , construído através do agrupamento de objetos, em \mathcal{O} , que pertencem a T .

Note que, se X_t for vazio, então (T, X_t) representa uma classe sem implementação, muito conhecida na literatura como *classe abstrata*.

4.5 Subtipos e Herança de Subtipos

Definição 4.5.1 (*Relação de Subtipos*):

Sejam T e S tipos e \preceq o símbolo que denota a relação de subtipos. Se $T \preceq S$, então dizemos que S é subtipo de T e, de forma equivalente, T é supertipo de S . Assuma $A_1, \dots, A_{n+m} \in \mathcal{N}$ são nomes distintos ($n > 1, m \geq 0$).

- (i) $T \preceq T$ para todos os tipos básicos;
- (ii) $(A_1 : T_1, \dots, A_n : T_n) \preceq (A_1 : S_1, \dots, A_n : S_n, A_{n+1} : S_{n+1}, \dots, A_{n+m} : S_{n+m})$, e $m \geq 0, n \geq 1$; se $T_i \preceq S_i, i = 1, \dots, n$.
- (iii) se $T \preceq S$, então $\{T\} \preceq \{S\}$, $\langle T \rangle \preceq \langle S \rangle$, e $\mathbf{ref} T \preceq \mathbf{ref} S$;
- (iv) $(S \rightarrow T) \preceq (S' \rightarrow T')$ se $S \preceq S'$ e $T' \preceq T$

Dessa forma, temos que, por (i), todos os tipos básicos são subtipos de si mesmos. Dado um tipo rotulado, $A : T$ por exemplo, se $T \preceq S$, então, também podemos dizer também que $A : S$. Além disso, seja uma função $f : S \rightarrow T$ e o um objeto, tal que $o : S'$ e $S \preceq S'$, então $f(o)$ é um valor válido e $f(o) : T$. A relação (iv) indica uma relação muito conhecida na literatura por *covariância* de tipos.

Definição 4.5.2 (*Hierarquia de subtipos*):

Uma hierarquia de subtipos (\mathcal{T}, \preceq) é um sistema de tipos resultante da aplicação da relação \preceq sobre o conjunto \mathcal{T} .

Exemplo 4.5.1 :

Considere o seguinte conjunto $\mathcal{T} = \{\text{veículo}, \text{avião}, \text{veículo terrestre}, \text{carro}, \text{trem}\}$. A hierarquia de subtipos (\mathcal{T}, \preceq) é formada por:

$$\begin{aligned} &\text{veículo} \preceq \text{avião}; \\ &\text{veículo} \preceq \text{veículo terrestre} \preceq \text{carro}; \\ &\text{veículo} \preceq \text{veículo terrestre} \preceq \text{trem}. \end{aligned}$$

Proposição 4.5.1 A estrutura (\mathcal{T}, \preceq) é um grafo acíclico dirigido sobre a relação \preceq .

Prova 4.5.1 Se provarmos que a estrutura (\mathcal{T}, \preceq) é uma estrutura fechada para reflexividade, anti-simetria e transitividade, provaremos que existe, entre seus elementos, uma ordem parcial, que pode ser representada através de um grafo acíclico (ver [1]).

De acordo com a definição 4.5.2, para dois dados tipos, S e T , em (\mathcal{T}, \preceq) , há uma relação de subtipos entre eles. Suponha $T \preceq S$, então, pela definição 4.5.1, $T \preceq T$, para qualquer $T \in (\mathcal{T}, \preceq)$, provando a reflexividade. Para dois dados tipos S e T , se $T \preceq S$, e $T \neq S$, pela definição 4.5.1, $S \preceq T$ é uma operação inválida (anti-simetria). Assumindo S , T e P tipos em (\mathcal{T}, \preceq) , e $T \preceq S$, $S \preceq P$, pelos itens (i), (ii) e (iii) da definição 4.5.1, podemos inferir que (\mathcal{T}, \preceq) é transitiva.

Do ponto de vista semântico, podemos encarar a estrutura hierárquica de subtipos como uma importante característica do modelo, uma vez que conduz à identificação de elementos reutilizáveis, que mantêm propriedades e características de seus subtipos. Este estudo gera, em nível de implementação, modelos mais poderosos no ponto de vista de mapeamento de informação, programação e simplicidade.

4.6 Hierarquia de Agregação

Objetos podem ser compostos por outros objetos através da relação “é-parte-de”. Um conjunto de objetos que mantêm relações do tipo “é-parte-de” é coletivamente chamado de objeto composto. Objetos compostos podem ser formados, na maioria das vezes, por construtores **set** e **tupla** aplicados sobre objetos primitivos, ou outros objetos compostos. Dessa forma, considere o_1, o_2, \dots, o_n objetos dados e A_1, A_2, \dots, A_n nomes distintos, então:

- (i) $o = [A_1 : o_1, A_2 : o_2, \dots, A_n : o_n]$ é um objeto composto (tupla) e o_i a projeção do objeto o sobre A_i , $i = 1, 2, \dots, n$;
- (ii) $o = \{o_1, o_2, \dots, o_n\}$ é um objeto (conjunto).

Postulado 4.6.1 *Seja O um conjunto não-vazio de objetos. Existe uma operação \diamond sobre O , chamada conexão, tal que:*

- (i) O é fechado sob \diamond , ou seja, $\forall x, y \in O$, se $x \diamond y$ é aplicável, então $x \diamond y \in O$;
- (ii) Objetos em O são idempotentes sob \diamond , ou seja, $\forall x \in O$, $x \diamond x = x$;
- (iii) \diamond é uma comutativa, ou seja, $\forall x, y \in O$, $x \diamond y = y \diamond x$;
- (iv) \diamond é associativa, ou seja, $\forall x, y, z \in O$, $x \diamond (y \diamond z) = (x \diamond y) \diamond z$.

Este postulado nos mostra que objetos podem se combinar e formar outros objetos, através da operação \diamond . A ordem de combinação entre os objetos não é importante, e novos objetos não são originados através da combinação de um objeto com ele mesmo.

Além disso, quando temos uma conexão \diamond entre dois objetos x e y , onde $x \neq y$, então (i) $x \diamond y = x$, ou (ii) $x \diamond y = y$, ou existe um outro objeto z , tal que (iii) $x \diamond y = z$, onde $z \neq x$ e $z \neq y$. A situação (i) ocorre em casos onde existe conexão entre objetos x e y , sendo que y é subtipo de x (relação “é-parte-de”). De forma equivalente, podemos analisar o caso (ii). Em (iii), temos que x e y são objetos diferentes e um não é subtipo do outro; nestes casos, nos referimos à conexão entre ambos como uma associação.

Definição 4.6.1 *Seja O um conjunto não-vazio de objetos e $x \in O$. x é dito objeto composto se, e somente se, existem objetos $y, z \in O$, tais que $y \neq z$ e $x = y \diamond z$. Caso contrário, x é um objeto simples em O .*

Como a relação \diamond é idempotente, então (i) $x = x \diamond x$, ou seja, um objeto x é formado por, no mínimo, um objeto (ele mesmo), que pode estar associado a um outro objeto, que é básico ou resultado da conexão de outros objetos.

Definição 4.6.2 *Seja $x, y \in O$. Então x tem uma relação “é-parte-de” com y se, e somente se, $x \diamond y = y$.*

Dessa forma, abrangemos um caso particular da composição de objetos, onde um objeto é um subtipo de outro objeto. Mais que isso, a relação $x \diamond x = x$, ilustra que todo objeto é parte de si mesmo.

Proposição 4.6.1 *Uma relação “é-parte-de” apresenta ordem parcial.*

Prova 4.6.1 *Provaremos que existe a relação “é-parte-de” é parcialmente ordenada, provando as propriedades reflexiva, transitiva e anti-simétrica. Pela definição 4.6.2, temos que:*

- (i) a relação “é-parte-de” é reflexiva, ou seja, $x \diamond x = x$ (reflexividade);
- (ii) se $x \diamond y = y$ e $y \diamond x = x$, então $y = x \diamond y = y \diamond x = x$ (anti-simetria);
- (iii) se $x \diamond y = y$ e $y \diamond z = z$, então $x \diamond z = x \diamond (y \diamond z) = (x \diamond y) \diamond z = y \diamond z = z$ (transitividade).

5 Biologia Molecular

Como vimos na seção anterior, objetos complexos nos fornecem uma maneira de modelar elementos e relações em um nível de abstração acima da modelagem relacional de dados. Particularmente, este raciocínio será muito útil para a modelagem de estruturas em Biologia Molecular Computacional. Cadeias de elementos, estruturas proteicas, elementos de análise de microarray, clusters funcionais, conjuntos resultantes de operações de classificação, entre outros, são estruturas que, em função de sua complexidade semântica, não podem ser modelados, em sua plenitude, através da abordagem relacional.

Em Biologia Molecular Computacional, a modelagem de informação é uma difícil tarefa. Muitas vezes, organismos diferentes, mesmo advindos de ancestrais comuns, apresentam diferenças fisiológicas e comportamentais difíceis de identificar, até mesmo para especialistas. Dessa forma, é difícil encontrar pesquisadores que conheçam profundamente as características de comportamento de cada organismo, podendo associá-lo a seus ancestrais.

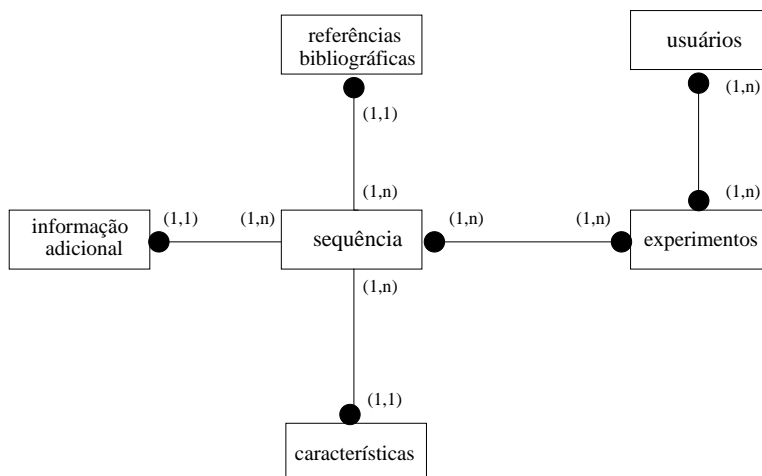


Figura 2: Relação entre sub-domínios na representação de dados de Biologia Molecular

Uma vez estudados os conceitos básicos para a definição de relações e objetos complexos, podemos tentar modelar o domínio de aplicação de Biologia Molecular sob o ponto de vista de modelagem orientada a objetos. Este domínio é constituído de seis subdomínios, sejam eles **usuários**, **sequência**,

dados experimentais, referências bibliográficas, dados experimentais, características e informações adicionais.

Os subdomínios se relacionam de forma associativa, tal como ilustramos na figura 2. Cada um dos sub-domínios é representado por um conjunto de objetos relacionados entre si, que descreveremos com mais detalhes nas sub-seções a seguir.

5.1 Usuários

Este sub-domínio é responsável pelo controle de acesso de usuários aos dados do sistema.

Definição 5.1.1 (*usuário*)

Podemos definir usuário como o ator devidamente cadastrado no sistema através de uma identificação única, que possui autorização para fazer uso dos recursos e serviços oferecidos pelo mesmo.

É fácil verificar a existência de dois níveis de usuários: pesquisador e administrador. Os usuários pesquisadores mantêm vínculo de propriedade com o domínio de dados experimentais, ou seja, somente os proprietários dos experimentos têm poder de acesso a seus dados. Existe a possibilidade de liberação de regras de restrição para outros usuários, entretanto, estas autorizações devem partir dos responsáveis pelo experimento. Os usuários administradores possuem permissão sobre todos os dados do sistema e são responsáveis pela sua manutenção, tanto do ponto de vista de serviços quanto de estrutura.

Cada usuário pesquisador pertence a um grupo de pesquisa. Normalmente, experimentos são acessíveis a todos os integrantes do grupo, que possuem interesse direto no seu desenvolvimento. O sistema deve prever regras de restrição de acesso baseadas na identificação de cada usuário e cada grupo.

Usuários e grupos mantêm, entre si, uma relação de conexão associativa, ou associação. Esta associação possui cardinalidade $n - m$ [9, 10] e, do ponto de vista de orientação a objetos, é uma relação associativa simples.

5.2 Dados experimentais

Dados experimentais são informações resultantes de processos de análise sobre dados iniciais. Durante a fase de sequenciamento, várias operações de agrupamento, filtragem, purificação e comparação são realizadas sobre

as sequências. Cada um destes resultados possui um determinado significado semântico dentro do experimento, que pode variar de acordo com a interpretação do pesquisador.

Dessa forma, os dados experimentais são resultado da aplicação de algoritmos e da interpretação dos resultados por parte do pesquisador. A própria escolha dos algoritmos influi diretamente não só no resultado, como na qualidade do trabalho final.

Dados experimentais são altamente dinâmicos, e semanticamente variáveis. Dessa forma, é difícil definir estruturas genéricas para cobrir este sub-domínio. Mesmo assim, podemos utilizar alguns conceitos que tornam possível generalizar porções deste ambiente. Por exemplo, a idéia de agrupamento é um conceito muito presente e que pode ser usado como objeto generalizador, tal como o estudo feito em [11, 12].

5.3 Sequência

Sequência é o sub-domínio mais estruturalmente mais simples, e semanticamente mais importante do modelo. Basicamente, todas as operações realizadas em algum experimento, envolvem a manipulação de informação completa ou parcial da sequência de nucleotídeos.

Definição 5.3.1 (*sequência*):

Uma sequência s é uma cadeia de caracteres que representa o encadeamento de bases de nucleotídeos formadoras de DNA (ácido desoxirribonucleico), ou RNA (ácido ribonucleico). Estruturalmente, a sequência pode ser representada por uma cadeia de nucleotídeos, representados pelos caracteres a (adenina), c (citosina), g (guanina) e t (timina).

5.4 Características (Regiões Funcionais)

Definição 5.4.1 (*Características ou Regiões Funcionais*) *Características são regiões da cadeia de DNA que possuem algum papel biológico considerado importante para o funcionamento do organismo, ou que é capaz de provocar alterações (sejam elas grandes ou não) no metabolismo por sua ativação, desativação, presença, ou ausência.*

Sequências de DNA condicam várias informações internas de cada organismo vivo, entretanto, nem todas possuem algum papel direto no funcionamento efetivo de células ou tecidos [13, 14]. Existem regiões funcionalmente ativas e outras não. As regiões ativas são aquelas que codificam sequências de aminoácidos, gerando proteínas, que são consideradas as

unidades funcionais básicas de todas as operações metabólicas, servindo como catalisadores (ou inibidores) de reações químicas, por exemplo. Regiões não-ativas podem representar locais sem função, ou que perderam sua capacidade de codificar aminoácidos durante o processo evolutivo da espécie.

Não é difícil perceber a particularidade deste sistema, assim como a complexidade associada aos atores do sistema e suas relações.

5.5 Referências Bibliográficas

Definição 5.5.1 (*Referências bibliográficas*):

As referências bibliográficas são trabalhos científicos que divulgam resultados de um processo de pesquisa desenvolvido por um pesquisador, ou grupo, bem como seus fundamentos teóricos e sua evolução de raciocínio.

Referências bibliográficas representam um importante item no ambiente de Biologia Molecular, uma vez que armazenam os resultados de pesquisa, as técnicas usadas e para obtê-los.

Na seção 5.7, veremos, com mais detalhes, como modelar a parte estrutural de objetos complexos, com um exemplo direcionado ao sub-domínio de referências bibliográficas.

5.6 Informações Adicionais

Definição 5.6.1 *O sub-domínio de informações adicionais armazena dados de nível cadastral da sequência, normalmente são informações que não sofrerão alterações ou remoções, mas podem ser utilizados como critério para algumas operações importantes (por exemplo, agrupamento).*

Do ponto de vista semântico, os objetos que fazem parte deste domínio apresentam grandes variações, entretanto, sua função dentro do ambiente não justifica a criação de sub-domínios específicos para seu tratamento. Além disso, sua característica estática pode ser utilizada para a geração de objetos com algum grau de flexibilidade, sem sacrificar, no entanto, seu conteúdo semântico.

Um dos principais objetos deste modelo é, sem dúvida, a taxonomia da sequência. Dados taxonômicos são de grande importância para verificar algumas propriedades entre sequências de famílias próximas e ajuda na elaboração de hipóteses e direcionamento de pesquisa. A taxonomia pode levar à inferência de comportamentos ou reações baseados em observações evolutivas.

Além da taxonomia, outros dados importantes dizem respeito à origem da informação. Métodos, técnicas e métricas utilizados durante o processo de obtenção da sequência podem ser de grande valia para a reprodução do experimento ou para a identificação de falhas operacionais.

Este sub-domínio é altamente amplo. Uma das maiores dificuldades é modelar seus objetos de forma que sua estrutura seja suficientemente genérica e flexível, possibilitando a integração de novos objetos, no decorrer do tempo.

5.7 Estrutura dos Sub-domínios

Para ilustrar uma possível estrutura de um objeto de um dos sub-domínios, vamos analisar a estrutura de um objeto publicação, pertencente ao domínio de referências bibliográficas. Este exemplo é tratado com mais detalhes em [15].

A estrutura de um objeto **Publicação** é representada por:

```
Publicação = {
    (título: string,
    autores: < ( nome: string,
                  inicial: character ) >,
    revista: [ comentarios: string,
              < medline-rev: string,
                titulo-rev: string,
                revista: string,
                issn: string > ]
    volume: string,
    edição: string,
    ano: inteiro,
    paginas: string,
    abstract: string,
    palavras-chave: {string} ) }
```

Este exemplo ilustra um caso onde é necessário realizar o tratamento de objetos considerando suas relações de agregação. O *título* representa o nome do trabalho científico armazenado no sistema. O atributo *autores* contem a lista de autores envolvidos no trabalho. A *revista* armazena as informações do veículo de publicação utilizado, normalmente revistas, anais de conferência, etc. Os atributos *volume*, *edição*, *ano* e *páginas* armazenam dados sobre o exato local de publicação da referência. Já *abstract* e *palavras-chave*

armazenam o resumo do artigo e as palavras-chave relevantes, respectivamente.

Note, neste exemplo, que somente os atributos *título*, *volume*, *edição*, *ano*, *páginas* e *abstract* são formados por tipos primitivos. O atributo *autor* é formado por um tipo composto, obtido através da construção de uma lista de tuplas. Cada uma das tuplas é formada por dois objetos do tipo string,

(nome: string, inicial: string),

onde nome armazena o último sobrenome do autor e inicial armazena as iniciais dos pré-nomes deste mesmo autor. Esta tupla, do ponto de vista semântico, representa um objeto “autor”. O atributo é formado por uma lista de objetos do tipo autor.

O atributo *revista* é formado por um tipo variante (ou união disjunta) formado por comentários ou uma lista de objetos primitivos. Esta lista de objetos é formada por medline-rev (abreviação do título da revista no padrão Medline), titulo-rev (abreviação ISO do título da revista), revista (título completo da revista) e issn (International Standard Serial Number).

O atributo *palavras-chave* é um conjunto de palavras que indica a área de pesquisa e os tópicos principais tratados na publicação, sendo representado por um tipo agregado conjunto de objetos do tipo básico string.

Abaixo, apresentamos uma das instâncias desta classe de objetos, onde podemos visualizar a relação entre os atributos e conseguimos ter idéia do comportamento das relações complexas.

```
Publicação = {
    (título: "Structure of the human perforin gene",
     autores: < ( nome: "Lichtenheld", inicial: "MG" )
                (nome: "Podack", inicial: "ER">,
     revista: [ comentarios: "",
                < medline-rev:"",
                  titulo-rev: "",
                  titulo-rev: "",
                  issn: ""],
     volume: "143",
     edição: "12",
     ano: 1989,
     paginas: 4267-4274,
     abstract: "We have cloned the human perforin
                (P1) gene ...",
```

```
palavras-chave: {Sequência de amino ácidos,  
                sequência de bases,  
                exons  
                genes, estrutural} ) }
```

6 Estágio de Pesquisa

Os conceitos apresentados neste trabalho são de grande importância na definição de critérios de modelagem de dados genéticos. Na área de bancos de dados, atualmente, não há grandes contribuições científicas na área de modelagem de dados para este domínio de aplicação específico. Mais que isto, a visão de entidades na área de Biologia Molecular Computacional como instâncias de classes de objetos é uma abordagem que pode trazer grandes benefícios do ponto de vista conceitual e estrutural, uma vez que considera aspectos semanticamente complexos, como agregações, hierarquias, subtipos, etc.

Este estudo é de fundamental importância para o prosseguimento de meus trabalhos de mestrado, devido ao seu conteúdo teórico. A definição de estruturas e o entendimento de seu comportamento, bem como os meios de comunicação, incluindo regras de restrição, servem de base para a elaboração de técnicas de modelagem para sistemas integradores de dados.

O principal papel do integrador é possibilitar a comunicação entre diferentes atores do cenário de heterogeneidade. Uma vez conhecidos a natureza dos objetos, suas propriedades e mecanismos de relacionamento, é possível dimensionar melhor as possibilidades de melhoria na definição de estruturas de integração genéricas e flexíveis [16].

Este trabalho faz parte do projeto de mestrado “Ambiente de Integração de Dados e Processos de Análise de Genoma”, financiado pela FAPESP, sob o número 00/10062-3.

7 Conclusão

O surgimento de aplicações com nível conceitual mais complexo trouxe a necessidade de trabalhar com estruturas de objetos. A grande vantagem da modelagem orientada a objetos é que os objetos são capazes de conservar algumas propriedades de dependência funcional, estrutural e hierárquica. Além disso, a modelagem através de objetos nos permite encapsular dados e abstrair a informação em níveis superiores. Quando falamos em Biologia Molecular Computacional, dificilmente conseguiremos fugir de estruturas fortemente relacionadas por níveis de hierarquia e com dados agregados advindos de várias fontes. Neste sentido, o estudo métodos e métricas de objetos para representação de dados de Biologia Molecular representam um importante fator de sucesso tanto na modelagem quanto na implementação de um sistema de dados estável e eficiente.

Referências

- [1] LIU, L. *A formal approach to structure, algebra and communication of complex objects*, 1992. Tese de doutorado, Proefschrift Katholieke Universiteit Brabant Tilburg - the Netherlands.
- [2] ELMAGARMID, A., RUSINKIEWICZ, M., SHETH, A. *Management of heterogeneous and autonomous database systems*. 1st. ed. Morgan-Kaufman, 1998.
- [3] ELLIS, G. R. *Managing complex objects*, 1995. Tese de doutorado, Department of Computer Science - University of Queensland.
- [4] DATE, C. J., DARWEN, H. *A guide to sql standard*. 4th. ed. Addison-Wesley, 1996.
- [5] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. *Introduction to algorithms*. 2nd. ed. MIT Press, 2001.
- [6] GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley, 1995.
- [7] BOURNAUD, I., GANASCIA, J.-G. Conceptual clustering of complex objects: A generalization space based approach. *International Conference on Conceptual Structures*, p. 173–187, 1995.
- [8] PETER BUNEMAN, S. D., WATTERS, A. A semantics for complex objects and approximate queries. *7th Symposium on Principles of Database Systems*, p. 305–314, 1998.
- [9] ELMASRI, R., NAVATHE, S. B. *Fundamentals of database systems*. 3rd. ed. Addison-Wesley, 1999.
- [10] SILBERSCHATZ, A., KORTH, H. F., SUDERSHAN, S. *Database system concepts*. 3rd. ed. McGraw-Hill, 1997.
- [11] TOM KELLER, G. G., MAIER, D. Efficient assembly of complex objects. *1991 ACM-SIGMOD Conference*, 1991.
- [12] SCHÖNING, H., SIKELER, A. Cluster mechanisms supporting the dynamic construction of complex objects. *3rd. International Conference of Foundations of Data Organization and Algorithms*, p. 31–46, 1989.

- [13] VOET, D., VOET, J. G. *Biochemistry*. 2nd. ed. John-Wiley, 1994.
- [14] STRYER, L. *Biochemistry*. 4th. ed. W.H. Freeman, 1994.
- [15] BUNEMAN, P., S.B.DAVIDSON, HART, K., OVERTON, C., WONG, L. A data transformation system for biological data sources. *Proceedings of the XXI International Conference on VLDB*, 1995.
- [16] CHAWATHE, S., GARCIA-MOLINA, H., HAMMER, J. The tsimmi project: Integration of heterogeneous information sources. *Proceedings of 16th Meeting of Information Processing Society of Japan*, 1994.