4
Descrição do Modelo de Dados

4.1
Introdução

Neste capítulo nós propomos um novo modelo de dados orientado a objetos denominado *BioConceptual* . O objetivo desta proposta é oferecer ao projetista do banco de dados uma lingaugem mais expressiva para projetar esquemas de dados para aplicações biológicas facilitando a sua construção. O *BioConceptual* tenta trazer o modelo de dados mais perto para o domínio da biologia mantendo distante dos aspectos de implementação. Desta forma, o modelo de dados *BioConceptual* é definido como um modelo de dados conceitual visto que não utiliza qualquer tipo de construtor que sugere a forma como este modelo será implementado.

O *BioConceptual* foi projetado para preenchar todos os requisitos acima. A seguir, nós descreveremos as principais propriedades do *BioConceptual* e justificaremos, quando necessário, as decisões que foram tomadas.

4.2
Requisitos para um Modelo de Dados Biológico

Nesta seção, nós listamos os requisitos para um novo modelo de dados para a biologia molecular. Esses requistos foram coletados no Capítulo 3. Eles são baseados nos problemas encontrados durante a modelagem de dados da biologia molecular utilizando linguagens de modelagem conceitual de dados tradicionais. A Tabela 4.1 descreve os requisitos para um novo modelo de dados conceitual.

4.3
BioConceptual e o Modelo de Dados ODMG

Após analisar diversos modelo de dados(e.g. ER[1], EER[8], ORM[52] or OO[43] etc), decidimos usar um modelo de dados orientado a objetos como base para o nosso modelo de dados. São dois os maiores benefícios para o uso de uma abordagem orientada a objetos:

– Um modelo orientado a objetos tem bastante expressividade para especificar representações complexas e permite a unificação do desenvolvimento da aplicação e do

Tabela 4.1: Resumo dos requisitos para um modelo de dados conceitual para biologia molecular

Requisito	Descrição
#1	Representar relacionamentos com ordem complexa e padrões.
#2	Incluir propriedades em conceitos denominados de alto-nível sem necessidade de alterar os conceitos de base ou de baixo-nível
#3	Enriquecer a semantica dos modelos conceituais usando ontologias
#4	Permitir explicitar as semantica dos relacionamentos
#5	Representar funções e estruturas biológicas de forma distinta
#6	Representar herança não-monotônica
#7	Representar relacionamentos probabilísticos e impíricos
#8	Representar restrições complexas as quais garantam a qualidade dos dados de entrada e mantenham a sua consistências
#9	Permitir a definição de visões sobre o modelo conceitual
#10	Permitir representar hierarquias grandes
#11	Representar aspectos espaciais e temporais do processos biológicos
#12	Permitir a definição de construtores de alto-nível baseados nas definições de construtores de baixo nível

banco de dados em um ambiente sem divisões entre o banco de dados e a linguagem de programação;

– As aplicações requerem menos código, usam mais naturalmente a modelagem de dados, e os códigos são mais fáceis de manter. Desenvolvedores de sistemas orientados a objeto podem criar aplicações completas de banco de dados sem muito esforço.

Como consequência desta decisão, decidimos usar e estender a especificação do padrão ODMG 3.0[43] para incorporar as necessidades do novo modelo de dados. Acreditamos que a adoçõe do padrão ODMG facilitará a compreensão do modelo e a compatibilidade com outras linguagens de modelagem orientadas a objetos como UML, linguagens de programação usando objetos (ex. Java), bancos de dados objeto-relacional e bancos de dados puramente orientados a objetos. Neste contexto, nós realizaremos as seguintes extensões no modelo de dados da ODMG.

4.4 Tipo de Dado Objeto

Uma característica importante de modelos de dados conceituais é a facilidade de especificar coisas do mundo real ou do universo do discurso (UoD) usando tipos de abstração. Representação é uma saída de um processo de abstração, seguido por um processo de classificação (grupando todas as entidades que pelo ponto de vista da aplicação podem ser consideradas similares). As classes que eventualmente resultam deste processo de classificação são descritas como tipos de objetos em um esquema de banco de dados. Suas descrições consistem em dar um nome para tipo de objeto, o qual distingue ele de todos os outros tipos de objetos, e definir as propriedades de interesse que um projetista deseja associar com o tipo de objeto. O termo genérico propriedade inclui tanto atributos

(dados que podem ser recuperados, atualizados, e visualizados) e métodos (operações que podem ser executadas sobre os objetos do tipo definido).

No *BioConceptual* um tipo de dado objeto é um tipo de abstração que permite o projetista representar o domínio da aplicação em termos dos dados. Por exemplo, se um projetista de dados especificar em seu esquema o conceito Gene, ele usuará o construtor de tipo de dado objeto para definir este conceito.

A Figura 4.1 ilustra a criação de uma instância de um tipo de objeto do*BioConceptual* associando a ele o nome Gene e descrevendo seus atributos como name, authority, e sequence. O projetista de dados está neste momento criando um novo tipo de dado chamado Gene no esquema de dados. A definição do tipo Gene ilustrada na Figura 4.1 mostra dois níveis de representação: nível do meta-modelo e nível do modelo. No nível do modelo é ilustrado o construtor tipo de dado objeto e no nível do esquema é apresentada a definição do tipo Gene, a qual é representada graficamente por um retângulo com uma texto interno definindo o nome do tipo.

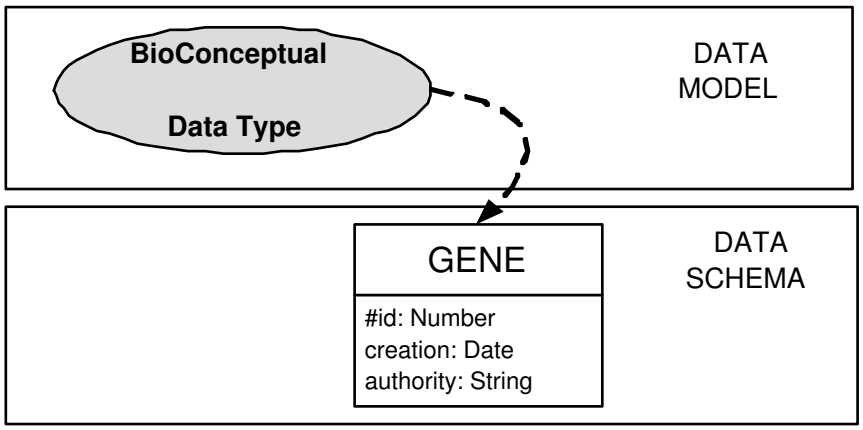

Figura 4.1: instanciação de um Tipo de Dado Objeto do *BioConceptual*

As entidades agrupadas no tipo de dado objeto são chamadas de objetos ou instâncias deste tipo de dado objeto. Uma instância do tipo acima pode guardar um valor (17456, 25/03/2003, 'HUGO_NC'), i.e., uma composição de três valores atômicos, um para cada atributo: id (codificado como um inteiro), creation (codificado como data), e authority (codificado com um caracter string). Cada instância representa um gene do mundo real relevante para a aplicação. Uma associação de instâncias com um tipo de dado objeto é referida como uma população deste tipo de dado objeto. Populações mudam com o tempo visto que novas instâncias podem se tornar relevantes ou antigas instâncias podem se tornar irrelevantes. Um exemplo de população de tipo de Gene é ilustrada na Figura 4.2 onde existem duas instâncias identificadas pelos números 17456 e 16775 com os respectivos valores de atributos.

Uma extensão de um tipo de dado objeto define um conjunto de todas as instância para um tipo de dado objeto em um banco de dados particular (o *BioConceptual* não sugere nenhum tipo de armazenamento físico). Se um objeto é uma instância do tipo de dado objeto Gene, então será necessário ser um membro da extesão do tipo Gene. Se o tipo Gene é um subtipo do tipo CodingRegion, então a extensão do tipo Gene é um subconjunto da extensão do tipo CodingRegion.

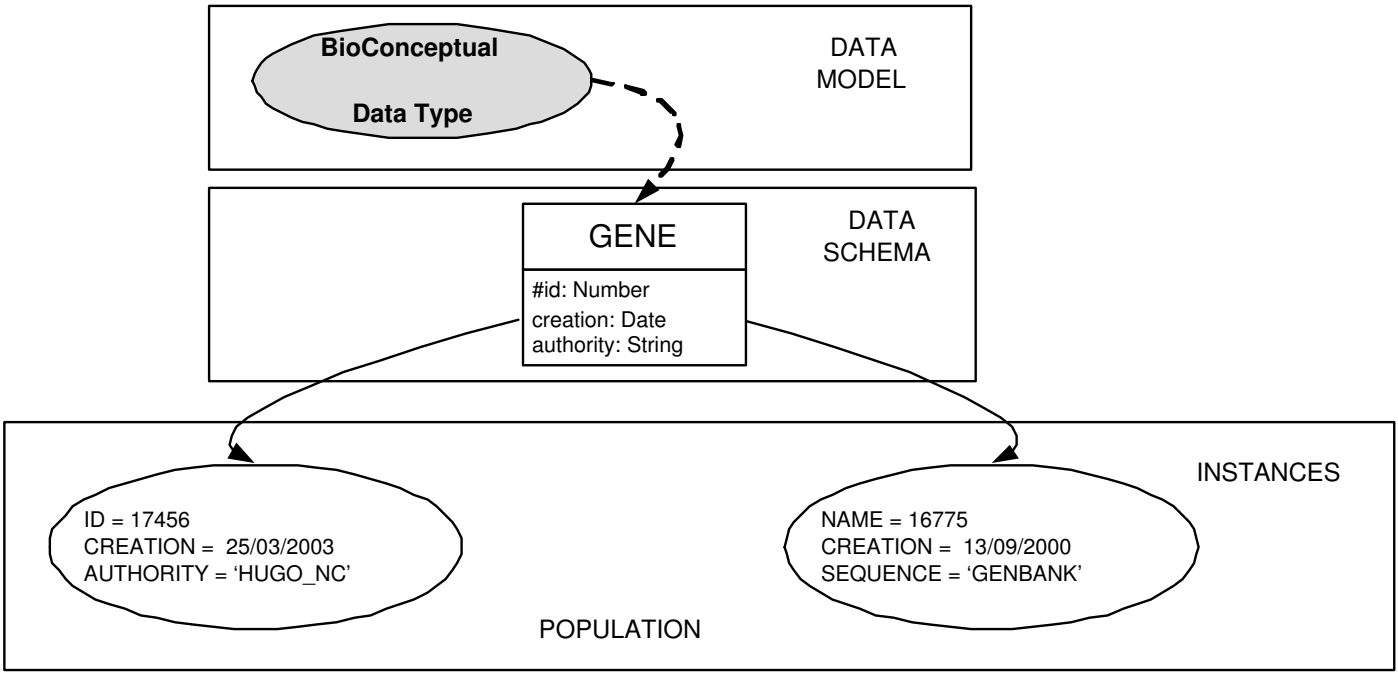

Figura 4.2: Exemplo de População do Tipo de Dado Objeto Gene

Para termos a certeza que duas instâncias diferentes são distinguíveis, mesmo que elas guardem o mesmo valor, é assumido que cada instância de qualquer tipo de dado objeto é identificada pelo SGBD usando um atributo adicional, definido pelo sistema, chamado de identificador do objeto, ou simplesmente oid. Este oid gerado pelo sistema é imutável e único, garantindo que dois objetos não possuem o mesmo oid. O formato completo de um instância de um dado tipo de dado objeto é então <oid, valor>, onde o valor denota o valor da instância, formado juntando um valor por atributo e formado de acordo com a definição do tipo.

Oids existem principalmente para uso interno pelo sistema, normalmente não podem ser apresentados ou atualizados pelos usuários. Porém, as suas existências são aparentes nas interações entre os usuários e o sistema, na forma como as instâncias foram recuperadas pelo sistema e na execução de operaçoes executadas pelos usuários (ex., criando um link para esta instância, inserindo uma instância com o mesmo oid em outro tipo). Por esta razão a existência de objetos identificadores é uma questão relevante no nível conceitual (mas no modo como são implementadas não é). Representando um objeto via seu oid garante que é de fato aquele objeto que será usado na operação, e não qualquer objeto que possua o mesmo valor em seus atributos.

Identificadores definidos pelos usuários (denominados "chave-primária" ou "atributos únicos" em SGBDs relacionais), tal como um número para nosso tipo de objeto Gene, podem ser realmente usados pelas aplicações para identificar objetos em um tipo (ex., existe um único gene na aplicação que possui o número 17456). No entanto, esses atributos não podem ser seguramente utilizados como oids, porque isto poderia confundir identidade com valores dos dados. Por exemplo, sistemas relacionais, os quais não usam objetos identificadores, não podem permitir atualização em um atributo que é parte da chave-primária, porque existe uma ambiguidade na intenção de atualizar, se é para um novo

objeto ou simplesmente alterar o valor de uma atributo de um objeto já existente (sem mencionar o problema de manter as ligações que outros objetos podem ter sobre o objeto atualizado). O fato dos valores de um atributo serem únicos em um tipo objeto (ex., duas instâncias não podem compartilhar o mesmo valor para este atributo) é apropriadamente expressada como um restrição de integridade específica que tem um formato pré-definido. A Figura 4.2 apresenta um diagrama de tipo de objeto que inclui a especificação de um identificador definido pelo usuário, identificado pelo símbolo '#'.

4.5 Atributos de um Tipo de Dado Objeto

Em aplicações reais podem existir uma grande quantidade de atributos para um tipo de objeto. Apresentá-los como uma longa lista desestruturada não é melhor maneira para representar a suas semântica. Por exemplo, o conceito publicação é extensivelmente usado para abstrair da informação detalhada que faz uma publicação sobre genes a qual possuiria conjunto dos autores, título, local da publicação e data É muito importante que um modelo conceitual de dados suporte esta maneira muito natural de organizar informação. Em outros termos, existem dois tipos de atributos, Aqueles que diretamente guardam um valor, e.g., número de identificação do Gene (Figura 4.1), os quais são chamados de atributos simples ou atômicos. E aqueles contendo um conceito com estrutura, e.g., publicação, são chamados de atributos complexos. Um atributo complexo é o nome que denota um conjunto de atributos que são tanto complexos como simples. Esta definição recursiva permite arbitrariamente construir estruturas complexas de atributos para lidar com a complexidade dos objetos do mundo real, onde a complexidade muda de um tipo de objeto para outro. Um tipo de objeto contendo atributos complexos é chamado de tipo de objeto complexo. Tipos de objetos complexos são suportados por SGBDs orientados a objetos mas não são suportados por SGBDs relacionais. SGBDs relacionais objetos são classificados entre esses dois, visto que eles suportam construtores para descrever atributos complexos.

Como somente atributos simples realmente guardam valores, o valor de um atributo complexo em uma instância é definido como a composição dos valores de seus atributos componentes. Desta forma, o valor de uma instância de um tipo de objeto complexo é um valor composto, interativamente composto de valores atômicos anexados às folhas da árvore de atributos.

Atributos mantém informação sobre os objetos que eles descrevem. Parte desta informação pode ser vista como inerente aos objetos em um tipo, i.e., poderia não fazer sentido para as aplicações lidarem com objetos que não contém esta informação. Por exemplo, pode ser uma regra da aplicação que objetos do tipo Gene sejam somente relevantes se a identificação do gene é conhecida (em outras palavras, a aplicação não pode propriamente gerenciar genes se a identificação do gene não está disponível no banco de dados). É importante ressaltar que ser inerente é dependente da aplicação; isto não necessariamente refere à uma propriedade que é instrinsicamente anexada às entidade no

mundo real. Por exemplo, aplicações estatísticas podem não requerer que uma identificação de gene seja armazenada no banco de dados. Em aplicação biológicas, identificações são dadas aos genes, as quais são usadas com um simples mecanismo de referência para se referir as tais objetos. Identificações de genes não são alguma coisa do mundo real. Elas são identificadores artificiais, definidos pelos homens para facilitar comunicação.

Aplicações reais fazem uso estensivo de tais identificadores artificiais, os quais são sempre considerados como inerentes ao objeto que eles identificam. Atributos que são inerentes aos objetos de um tipo são ditos como atributos obrigatórios: nenhuma instância de objeto pode ser criada sem tais atributos. Atributos que não são inerentes aos tipo de objetos são chamados de atributos opcionais, i.e., a criação de uma instância do tipo será aceita pelo sistema mesmo que os dados para esses atributos não sejam providos. A data de criação é um exemplo de um atributo opcional. Um objeto do tipo gene pode ser criado sem que um valor para criação do atributo data será dado, seja porque não existia data de criação para este gene ou porque no momento da criação do objeto a data de criação era desconhecida.

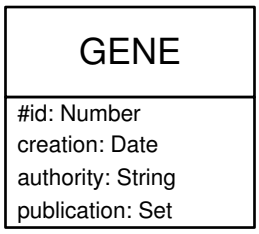

Figura 4.3: Atributos Multivalorados

Enquanto vários atributos simples guardam no máximo um valor, alguns são intensionados para guardar vários valores. Data de criação no tipo de obejto Gene é um exemplo de um atributo que na aplicação biológica é definido para guardar um único valor (i.e., a nenhum objto gene pode ser dado mais do que uma data de criação). Reciprocamente, o atributo publicação (o qual provê informação sobre publicações sobre os respectivos genes) é definido para receber diversos valores. De fato devem existir várias publicações para o mesmo gene. O valor deste atributo é uma coleção de valores onde cada valor pertence ao conjunto de valores válidos para publicações como ilustrado na Figura 5.2. Atributos que podem guardar mais do que um valor são chamados de atributos multivalorados.

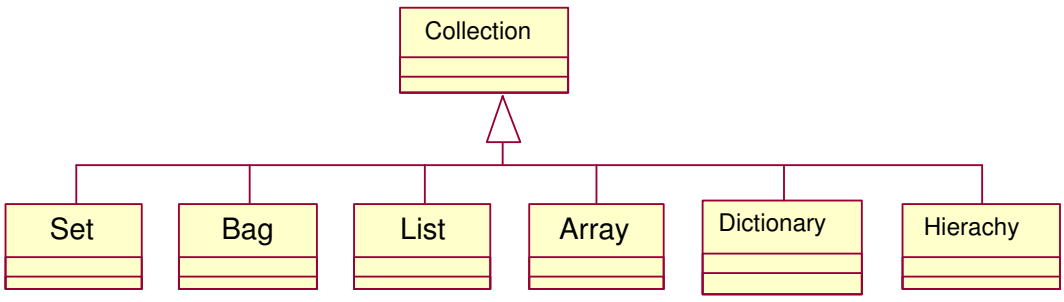

Figura 4.4: Coleção de Dados

Atributos multivalorados podem ser divididos em categorias distintas, dependendo do tipo de coleção que eles guardem. A Figura 4.4 apresenta um diagrama UML que

ilustra as possíveis categorias. Os vários valores podem assumir a forma de um conjunto (grupo desordenado sem valores duplicados), um sacola (grupo desordenado onde o mesmo valor aparece várias vezes), uma lista (grupo ordenado), um array (grupo de elementos ordenados que pode ser localizados pela posição), dicionário (sequência desordenada de pares chave-valor sem chaves duplicadas)e hierarquia (grupo ordenado hierárquico). Por exemplo, definindo a publicação como um atributo multivalorado do tipo conjunto é apropriado se a intenção semântica é simplesmente saber quais publicações falam sobre gene. A definição multivalorado-lista é apropriada para a classificação de publicações através de uma determinada ordem (e.g., relevância da publicação). Publicações ainda podem ser definidas como multivalorado-sacola para os casos onde queremos armazenar a mesma publicação várias vezes. Finalmente, a especificação de multivalorado-hierarquia poderia ser útil para se aplicada para organizar publicações que tem alguma relação entre si.

4.5.1 Cardinalidade do Atributo

A cardinalidade de atributos denota o número de valores que um atributo pode guardar. Cardinalidade é definida por dois números que definem o número de valores permitidos para a extensão do objeto. O primeiro número define o número mínimo de valores; o segundo número define o número máximo de valores. Por exemplo, a cardinalidade (1,n) define um atributo obrigatório e multivalorado, e a cardinalidade (0,1) define um atributo opcional e monovalorado. O uso do 'n' como cardinalidade máxima indica que qualquer número de valores é aceito.

4.6 Tipo de Relacionamento

Os objetos manipulados por uma aplicação não vivem isolados. Eles são interrelacionados por relacionamento que provêm caminhos para completar suas informações. Por exemplo, o objeto do tipo Gene pode conter atributos tais como sequência de ADN, data de cadastro, etc. No entanto, a sua ligação com o objeto do tipo Proteína que provê a informação sobre a função do objeto do tipo Gene. Esta informação não foi modelada como um atributo de Gene, porque a aplicação é também interessada em proteínas, independente da sua relação com Gene.

Relacionamentos em banco de dados representam ligações do mundo real que são do interesse da aplicação (da mesma forma que objetos representam entidades de interesse). A definição de um relacionamento passa pelo mesmo processo de percepção, abstração, e classificação que é usado para identificar tipos de objetos relevantes para a aplicação. O resultado do processo é um conjunto de tipos de relacionamentos, onde cada tipo de relacionamento determina uma ligação entre dois ou mais tipos de objetos, como mostrado na Figura 4.5.

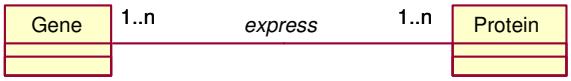

Figura 4.5: Um diagrama mostrando um tipo de relacionamento ligando dois tipos de objetos

Relacionamentos em um modelo conceitual podem lidar com uma variedade de semanticas. Isto pode levantar uma questão de quantos tipos diferentes de relacionamentos deveriam ser suportados por um modelo de dados. Apesar desta questão não ter uma resposta precisa, sabemos que o excesso de categorias de relacionamentos pode confundir o usuário e e a ausência de categorias pode diminuir o poder de expressão das mesmas.

Por causa disto, o *BioConceptual* utiliza a abordagem de permitir a complementação semântica das tipos de relacionamentos existentes, i.e., o *BioConceptual* disponibiliza os seguintes tipos de relacionamentos: "é-um", "parte-todo" e associação. A partir desses relacionamentos, o projetista do banco de dados pode usar um construtor Constraint 4.9.1 especificado para enriquecer a semântica dos outros construtores do *BioConceptual* .

4.7 Relacionamentos de Associação

Tradicionalmente, as associações são os tipos mais comuns de relacionamentos. Um tipo de relacionamento de associação é um relacionamento que liga duas ou mais instâncias de tipos de objetos sem impor qualquer semântica específica à ligação. No caso do modelo *BioConceptual* obrigatoriamente todo tipo de relacionamento deverá possuir uma semântica associada dada a complexidade apresentada no domínio da biologia molecular. Desta forma, o *BioConceptual* possui somente um tipo básico de relacionamento o qual deverá possuir uma semântica associada. A associação da semântica será obtida através do uso de um construtor especial denominado de Constraint 4.9.1.

Como veremos adiante, a semântica pode ser simplesmente um nome associado ao relacionamento. Este nome será interpretado formalmente pelo modelo como um predicado onde os parâmetros deste predicado são os tipos de objetos relacionados. Por exemplo, o tipo de relacionamento Express na Figura 4.5 modela relacionamentos de associação que representam fatos sobre a expressão gênica. A linha na Figura 4.5 ilustra o nome do relacionamento. A linha conectando os retângulos que emolduram os nomes dos tipos de objetos mostram quais os tipos de objetos que são ligados por este relacionamento. Por exemplo, Express na Figura 4.5 tem dois papéis: afinal é um relacionamento binário, um Gene exercendo o papel de agente que expressa e a proteína exercendo o papel de de agente expressado. Da mesma forma que os tipos de objetos, o conjunto de instâncias de um tipo de relacionamento constitui a sua população.

Associações no *BioConceptual* são ligações direcionadas dada a necessidade de indicar qual é a ordem dos parâmetros dentro do predicado que representa o relacionamento. As associações podem ter qualquer número de papéis e cada papel correspondente a um tipo de objeto contribui para uma instância que formará cada instância da extensão do tipo de

relacionamento. Em outras palavras, instâncias dos relacionamentos não podem ter papéis pendentes. Outras restrições estão relacionadas com as cardinalidades que caracterizam cada um dos papéis envolvidos. Para cada papel, suas cardinalidades mínimas e máxima são definidas. Essas cardinalidades definem o número de instâncias do relacionamento que, tomando um instante arbitrário na vida do banco de dados, podem ligar uma instância do tipo do objeto associado ao papel.

Por exemplo, a Figura 4.5 ilustra que no tipo de relacionamento Express o papel do Gene tem cardinalidades (0,n), e o papel da Proteína tem cardinalidades (1,n). A primeira cardinalidade significa que uma proteína pode nunca ter sido expressada por um gene, e uma proteína pode ter sido expressada por um ou mais genes. A última cardinalidade diz que um Gene no banco de dados ter que expressar pelo menos uma proteína. Ela também diz que vários genes podem expressar a mesma proteína.

4.8 Ligações "É-UM" entre tipos de objetos

Após termos discutido os três elementos fundamentais de todos os modelos conceituais - objetos, atributos e relacionamentos - o último construtor básico que resta a ser discutido é conhecido com ligação "É-UM", também referenciado como relacionamento de generalização/especialização. Enquanto relacionamentos lidam com o requisito em modelagem de dados em como relacionar objetos, as ligações "É-UM" lidam com o requisito de classificação. Como apresentado anteriormente, o processo de projeto passa por um processo de classificação que agrupa entidades do mundo real em conjuntos homogêneos, baseada na similaridade de sua criação. Por exemplo, todos os genes são classificados como formando um único conjunto com entidades similares, separados das proteínas que formam um outro conjunto. Características de cada conjunto são descritas em um esquema de banco de dados por um tipo de objeto ou tipo de relacionamento.

Ligações "É-UM" são diferentes dos tipos de relacionamentos. Relacionamentos normalmente ligam dois ou mais objetos representando diferentes entidades do mundo real. Como elas representam um fenômeno do mundo real, têm uma identidade e podem possuir propriedades. Ao contrário, ligações "É-UM" juntam duas instâncias que são diferentes representações da mesma entidade do mundo-real. Elas não tem identidade e não tem propriedades. Além disso, ligações "É-UM", por causa de suas semânticas que é de refinamento de classificação, são ligações diretas e por definição possuem um restrição de inclusão de população forçando que todo objeto instaciado em um subtipo também instânciado no supertipo. A Figura 4.6 ilustra a estrutura de dados que suporta classificações múltiplas em diferentes níveis de abstração de macromoléculas.

4.8.1 Multi-instanciação

Dizemos que modelos de dados e sistemas que permitem a mesma entidade do mundo real ser instânciada em diversos tipos de dados são ditas que suportam multi-

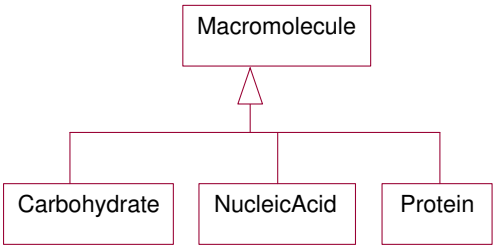

Figura 4.6: Um exemplo de ligação É-UM

instanciação. Nós chamamos de objeto global o conjunto de instâncias que representam o mesmo fenômeno do mundo real. As essas instâncias são dadas as mesmas identidades para preservar os mapeamentos 1:1 entre as entidades do mundo real e objetos globais[1]. Cada instância de um objeto global descreve a entidade do mundo real através de um ponto de vista particular do seu tipo de objeto. Similarmente, um relacionamento global é um conjunto de relacionamentos que representam o mesmo fenômeno do mundo real e tem a mesma identificação do relacionamento.

As estruturas "É-UM" são apropriadas para multi-instanciação. Na Figura 4.6, por exemplo, todas as populações são por definição instânciadas em ambas populações de *Protein* e *Macromolecule*. Porém, isto dependerá da necessidade da aplicação, ou seja se uma dada instância de um supertipo deve ser necessariamente uma instância de um subtipo ou não. Na Figura 4.6, esta questão poderia ser levantada e determinaríamos se cada macromolecula é necessariamente um carboidrato, um ácido nucleico, ou uma proteína. Se este for o caso, a restrição é feita explicitamente pela associação de uma restrição de cobertura para os tipos correspondentes.

Outra questão relevante com relação multi-instanciação é determinar se um objeto pode ser instânciado em mais de um subtipo. Na Figura 4.6, isto depende novamente da aplicação se uma macromolécula pode ser ao mesmo tempo um carboidrato e um ácido nucleico.

Podemos afirmar que dois tipos de objetos que pode ter instâncias para o mesmo objeto global são sobrepostos como ilustrado na Figura 4.7(b). Caso contrário, podemos dizer que existe uma restrição de disjunção entre os dois objetos como mostrado na Figura 4.7(a). Um subtipo e seu supertipo são por definição definições sobrepostas.

A sobreposição pode ser automaticamente inferida entre dois tipos de objetos que têm o mesmo subtipo, como é o caso apresentado na Figura 4.8, como todas as instâncias de *Human&Mouse* são necessariamente representadas em ambos tipos *Human* e *Mouse*. Quando um agrupamento de subtipos possui a restrição de disjunção e a restrição de cobertura são mantidas ao mesmo tempo, é dito que este agrupamento possui uma restrição de partição, onde a população do supertipo é particionada nas populações do subtipo.

[1] Este é um dos princípios fundamentais que foram definidos pelo famoso manifesto de orientação a objetos[11], destinado a definir as essências do modelo orientado a objetos

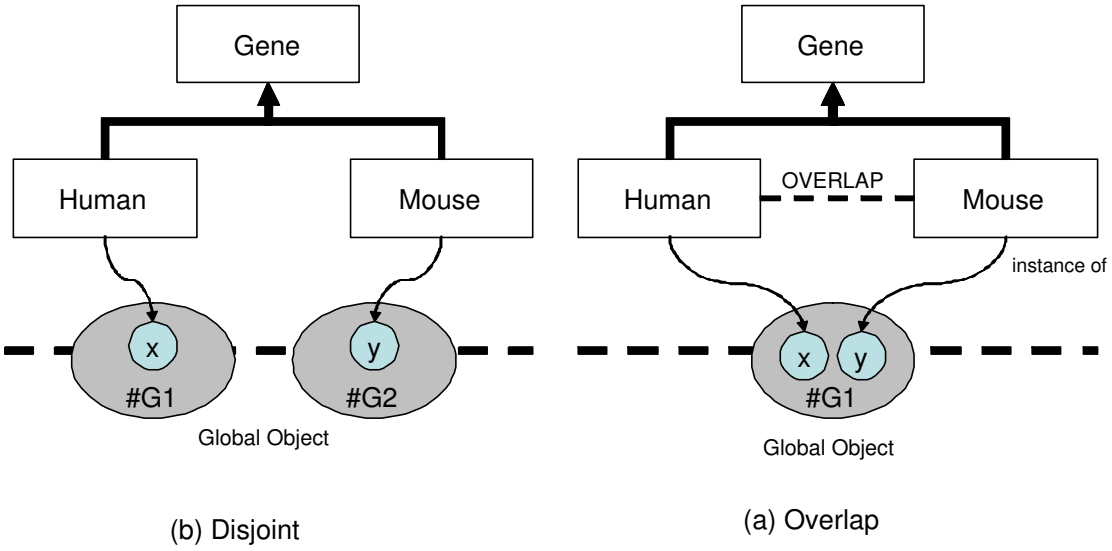

Figura 4.7: Exemplos de Sobreposição e Disjunção

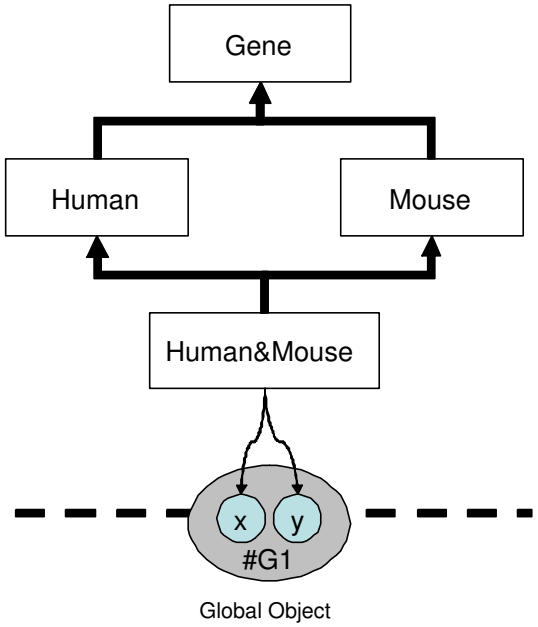

Figura 4.8: Neste exemplo é possível inferir sobreposição automaticamente

4.8.2 Ligações "É-UM" entre tipos de relacionamentos

Na maior parte dos modelos de dados, ligações "É-UM" podem somente ligar tipos de objetos. No entanto não existe razão essencial para impedir que ligações "É-UM" sejam usadas entre tipos de relacionamentos. Aumentar a expressividade dos construtores sem comprometer a clareza do modelo pode trazer benefícios importantes ao processo de modelagem. Supomos que dada familiaridade com o construtor de ligação "É-UM", é razoavelmente fácil entender seu uso com tipos de relacionamentos.

Faz sentido no BioConceptual suportar ligações "É-UM" entre tipos de relacionamentos porque eles possuem identidades da mesma forma que os objetos possuem. Além disso, este mecanismo propriciará a representação de heranças não-monotônicas, requisito importante levantado na seção 3.8.

A Figura 4.9 ilustra um possível uso desta facilidade. É assumido que o projetista de

um banco de dados biológico deseja criar um tipo de relacionamento separado para anotações sobre as sequências de ADN que são realizadas pelos cientistas durante a montagem do genoma. O sub-relacionamento pode ter propriedades adicionais com respeito ao super-relacionamento. O diagrama apresenta que ambos papéis nos sub-relacionamentos Manual e Automatizado tem a mesma cardinalidade com no super-relacionamento Annotate (papéis com cardinalidades redefinidas são redesenhados anexados ao tipo da sub-relação para apresentar as novas cardinalidades).

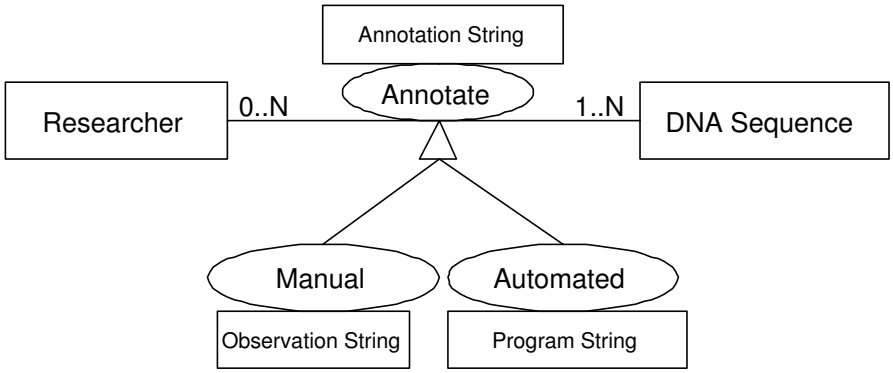

Figura 4.9: Refinando os tipos de relacionamentos

Como apresentado na seção 3.8, algumas vezes desejamos diferenciar os relacionamentos presentes em uma hierarquia. A Figura 4.10 apresenta o exemplo motivador da discussão sobre herança não-monotônica, a qual é um requisito para modelos de dados conceituais para biologia molecular. O diagrama ilustra o caso onde uma aplicação precisa representar a situação onde toda biomolécula pode ter múltiplos componentes estruturais. No entanto, nos casos de biomoléculas do tipo proteína só são permitidos ocorrerem no máximo um componente estrutural do tipo alpha helix. A Figura 4.10 apresenta o refinamento do tipo de relacionamento hasComponente e as cardinalidades que são herdadas e modificadas. O projeto garante que as proteínas terão somente uma estrutura alpha helix. Note que a cardinalidade (0,N) no papel do tipo StructuralComponent foi restrito para cardinalidade (0,1) no papel do tipo AlphaHelix.

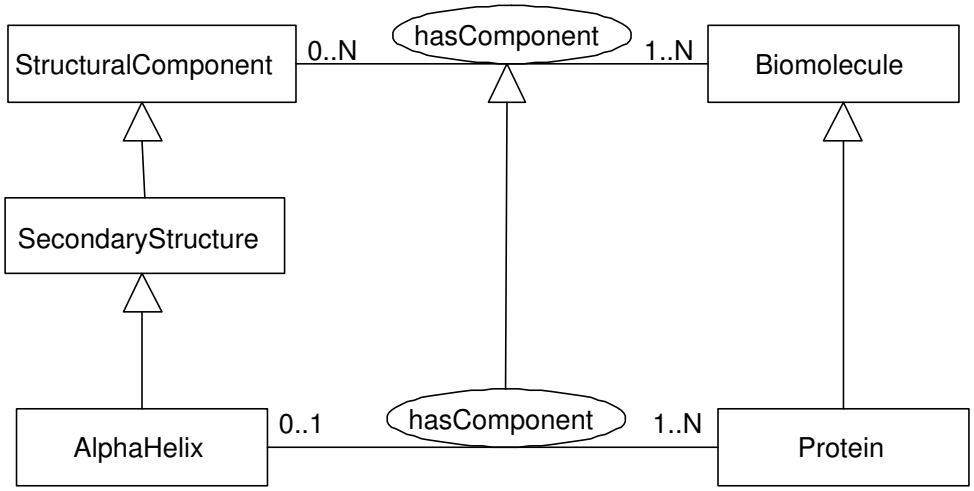

Figura 4.10: Representando Herança Não-Monotônica usando ligação "É-UM" entre tipo de relacionamento

Para exemplificar a possibilidade de adicionar papéis locais, vamos assumir que uma aplicação biológica tem diferentes políticas baseadas, por exemplo, na validação de anotações. A regra da aplicação é a seguinte: alguns processos de anotação são divididos em processos validados (por uma pessoa autorizada) e ainda não validados. Para representar esta diferença, um tipo de relacionamento ValidatedAnnotate é criado como subtipo do tipo de relacionamento Annotate. Neste caso, é necessário especificar o autor da validação, sendo necessária a ligação com o papel do tipo Validator. O diagrama final é apresentado na Figura 4.11. Pode-se observar que ValidatedAnnotate é um tipo de relacionamento ternário. Suas instâncias serão extensões das instâncias do tipo de relacionamento Annotate, no sentido de que elas adicionam o objeto Validator nas instâncias Annotate.

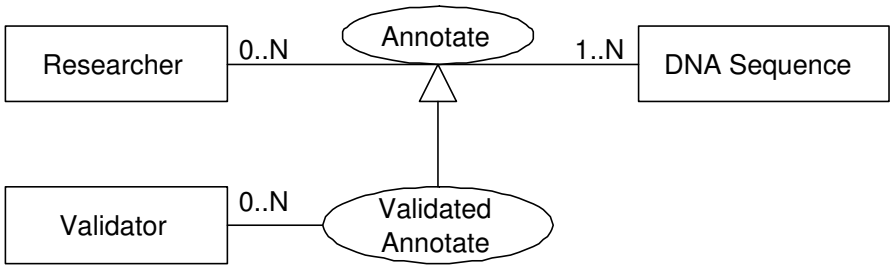

Figura 4.11: Um diagrama exemplificando o refinamento de um papel através do uso de ligações É-UM

4.9 Restrições de Integridade

Restrições de integridade provêem um meio mais preciso para definir a semântica dos dados e possuem um papel essencial no estabelecimento da qualidade de um banco de dados e na sua correta evolução. Restrições de integridade são assertivas que restringem como os dados podem aparecer em um banco de dados, para previnir a inserção de dados que são obviamente incorretos com as regras que governam o mundo real e sua representação no banco de dados. Diferentes tipos de restrição podem ser especificados. Restrições sobre os valores do domínio são possivelmente as mais simples tipos de restrições de integridade. Algumas aplicações exigem a especificação de um limite permitido para o intervalo de valores permitido pelo domínio em questão. Essa restrições objetivam limitar a possibilidade de entrada de dados errados, porém não evitam entrada de valores dentro dos limites permitidos mas que estao incorretos (e.g., entrando 0.53 ao invés de 0.25 não pode ser detectado por restrições simples).

Restrições de integridade podem oferecer complexidade arbitrária, envolvendo operações para recuperação dos dados e cálculos sobre os valores armazenados no banco de dados. A maioria delas restrigem valores, forçando conformidade com as regras da aplicação. As restrições de integridade podem também evitar a criação de instâncias de objetos ou relacionamentos. Por exemplo, restrições sobre cardinalidades associadas aos papéis nos relacionamentos limitam o número de instâncias de relacionamentos que podem ser criados.

4.9.1
Definição de Restrições

O *BioConceptual* propõe um construtor específico para especificacão de restrições de integridade denominado *Constraint*. A notação utilizada para este construtor será um quadrado com a letra "C" na sua parte superior. A linguagem utilizada para especificação da restrições será lógica de primeira ordem pela sua natureza universal pode ser diretamente mapeada para quase todas linguagens lógicas usadas em métodos formais. No *BioConceptual* o construtor *Constraint* pode ser aplicado a todos os outros tipos de construtores: tipos de objetos, tipos de associação, papéis, ligações "É-UM", atributos e inclusive sobre outro construtor *Constraint*. Por exemplo, a Figura 5.5 apresenta um diagrama que representa um relacionamento de homologia entre sequências de ADN. Normalmente em aplicações biológicas é necessário restringir quais tipos de homologias são de interesse da pesquisa realizada. Isto é representado criando uma restrição sobre o tipo de relacionamento Homologue. O tipo de restrição criado determina que instâncias do tipo ADNSequence, as quais participam da associação, devem obrigatoriamente ser diferentes e serem 80% similares. Além disso, o exemplo apresenta um outro tipo de restrição o qual é aplicado sobre o tipo de objeto ADNSequence. Esta restrição determina que somente as instâncias de ADNSequence sem erro e mutação podem ser aceitas como instâncias válidas.

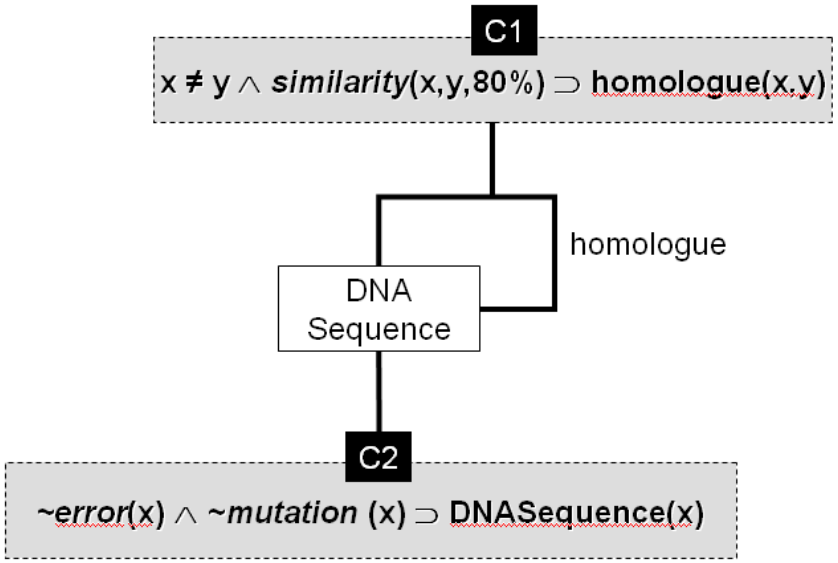

Figura 4.12: Exemplo de uso do construtor *Constraint*

4.10
Múltiplas Percepções e Representações

As especificações do modelo de dados *BioConceptual* que nós discutimos até então endereçam os requisitos clássicos de modelagem de dados. Nesta seção nós analisamos e desenvolvemos especificações para suportar múltiplas representações. Inicialmente discutimos porque múltiplas representações são essenciais para a modelagem de dados, principalmente no domínio da biologia molecular. Em seguida procedemos com a análise de como múlti-

plas representações pode ser suportadas a partir da perspectivas da modelagem de dados conceitual.

O objetivo principal de um banco de dados é armazenar representações de fenômenos identificados do mundo real, os quais são de interesse para um dado conjunto de aplicações. Quais representações devem ser armazenadas é determinado durante o processo de projeto do banco de dados, onde os requisitos da aplicação são analisados e convertidos em descrições de estruturas de dados formalizadas. Um dificuldade conhecida neste processo é como reconciliar requisitos divergentes das aplicações compartilhando o mesmo banco de dados. Enquanto o mundo é suposto como único, sua representação depende da intenção de como queremos percebê-lo.

Cada aplicação tem sua própria percepção do mundo real, e suas tarefas de processamento levam a requisitos específicos, ambos em termos de quais informações devem ser mantidas e em termos de como a informação deve ser representada. Diferentes aplicações que têm sobreposição de conceitos sobre o fenômenos do mundo real normalmente requerem diferentes representações do mesmo fenômeno. Por exemplo, duas aplicações que analisam proteínas podem possuir percepções diferentes. A primeira utiliza simplesmente a sequência de amino ácidos para realizar suas anotações. A segunda aplicação por sua vez necessita visualizar graficamente a estrutura tridimensional da proteína visando o reconhecimento da áreas ativas da mesma. Ambas aplicações percebem o mesmo fenômeno, a proteína, porém com percepções diferentes e representações diferentes.

Para suportar tal heterogeneidade de percepções, o banco de dados tem que ser hábil para lidar com múltiplas representações do mesmo fenômeno do mundo real, como apresentado na Figura 4.13.

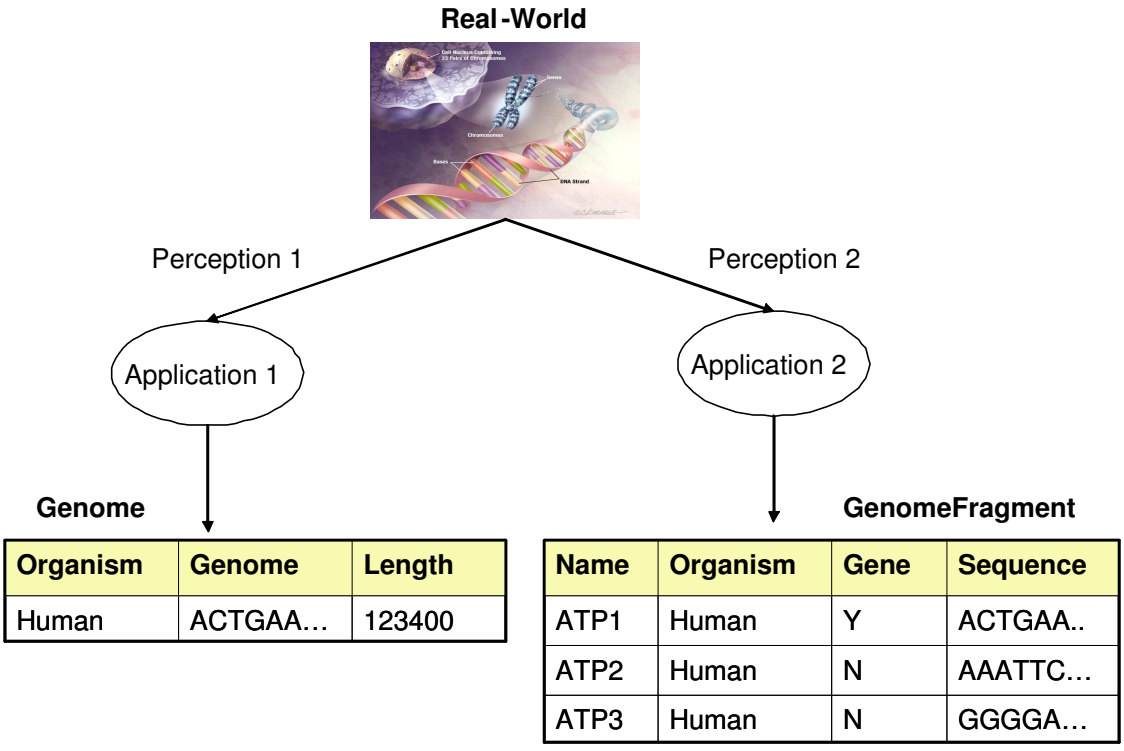

Figura 4.13: Exemplo ilustrando as diferentes percepcões possíveis do mesmo fenômeno do mundo real, e suas possíveis representações.

Note que múltiplas representações podem ser necessárias também em uma única percepção. Suponha uma única aplicação que precise de diversas representações em diferentes níveis de detalhe. Por exemplo, visualizar a sequência de ADN através de uma cadeia de caracteres ou de uma representação das ligações químicas presentes.

Os SGBDs atuais suportam parcialmente a multirepresentação, através de mecanismos de visões, os quais permitem derivar novas representações a partir de representações previamente definidas. SGBDs orientados a objetos também provêem hierarquias "É-UM" que permitem representações múltiplas como um refinamento progressivo de uma representação mais genérica para uma representação mais especializada.

Esta abordagem, no entanto, é insuficiente (em termos de poder de expressão, praticidade e clareza) para prover flexibilidade no suporte a múltipla representação. Mais importante ainda é o fato do conceito de percepção, i.e., o conhecimento do qual agrupamento de representações formam um conjunto consistente para uma aplicação, não ser suportado. Isto é indiretamente suportado através de um outro mecanismo, a identificação dos direitos de acesso, i.e., fornecendo a cada aplicação acesso para todas as visões e somente para aquelas visões que pertencem a sua percepção.

Uma importante diferença entre visões e multi-representação como proposto no BioConceptual é que as visões são unidades isoladas e anônimas de descrição de dados. Elas não definem um esquema que consistemente representa uma dada percepção do mundo de interesse. Além disso, visões perdem a informação que identifica as percepções que elas representam. Uma percepção da aplicação pode ser reconstruída a partir de especificações de controle de acesso, assumindo que cada transação é somente permitida acessar às estruturas de dados (estruturas básicas ou views) que correspondem a sua percepção. No entanto, misturar elementos de controle de acesso com elementos de representação é perigoso e conceitualmente confuso.

Em resumo, gerenciamento de dados modernos requerem um novo paradigma de representação, tal qual múltiplas representações de um mesmo fenômeno possam coexistir em um banco de dados, e isto deveria ser explicitamente descrito e conhecido pelo sistema de tal forma que pudesse ser gerenciado apropriadamente. Em outras palavras, suportar múltiplas percepções e representações significa que ambos usuários e sistemas estão cientes de que duas ou mais representações armazenadas estão descrevendo o mesmo fenômeno do mundo real, e que juntas formam uma percepção. Para atingir este objetivo, modelos de dados existentes precisam ser estendidos com novos conceitos possam identificar percepções e suas representações, e com uma semântica bem definida a qual diga que "esta representação descreve o mesmo fenômeno do mundo real que esta outra representação", complementada com restrições associadas e operadores.

4.10.1 Uso de percepções e múltiplas representações no *BioConceptual*

BioConceptual propõe um construtor chamado Perception que objetiva a especificação das percepções dos cientistas. Veremos que o conrtrutor Perception é uma forma de representar as percepções do interesse do cientista em termos de dados. Por exemplo, uma

aplicação de anotação usada por dois grupos distintos de pesquisa, os quais trabalham sobre o mesmo projeto genoma, precisam lidar com conceitos biológicos similares (e.g. genes, proteínas, etc), mas cobrem regiões diferentes do genoma. Cada laboratório possui um grupo de análise o qual estuda suas regiões, fazem anotações e sugerem funções associadas a cada gene. Usando a noção de percepção, podemos representar as percepções desejadas por cada laboratório e associá-las às representações existentes.

Para refletir as discrepâncias nas anotações, uma organização lógica dos termos biológicos e laboratoriais devem ser descritos. Normalmente, laboratórios usam suas próprias terminologias para descrever seus experimentos. Cada percepção oferece uma ontologia de termos, representando suas diferenças e similaridades. O desafio é misturar as duas hierarquias de entidades de uma forma não-obstrusiva que permita um integração mais suave entre as percepções. A mediação de percepções entre esses diferentes esquemas pode resolver todas as heterogeneidades semânticas.

Para ilustrar o uso de percepções no *BioConceptual* , assumamos que existam três laboratórios que percebam o conceito de Gene de três formas diferentes. A Figura 4.14 ilustra a representação do conceito Gene e a especificação destas três percepções (i.e.LaboratoryA, LaboratoryB e LaboratoryC). Um construtor Perception é na verdade uma estrutura hierárquica onde cada nó desta estrutura é um percepção. A informação das percepções são organizadas hierarquicamente onde cada nó interno representa uma especificação de percepção diferente e cada nó folha da árvore representa um elemento de esquema de dados, tal como: tipo de dado objeto, tipo de relacionamento ou atributo. A relação entre duas percepções é uma ligação do tipo "É-UM". Desta forma, a estrutura e o comportamento de cada uma das percepções apresentadas são herdadas pelas percepções subordinadas. O nó raiz da árvore da percepção é uma percepção também, os quais contém algumas estruturas genéricas assim como comportamento que é compartilhado por seus herdeiros, exceto se sobrescritos.

A idéia geral é associar tipos de objetos, atributos ou tipos de relacionamentos com percepções. Associações entre os elementos de esquema de dados e hierarquia de percepções podem ser implícitas ou explicítas. Um notação baseada em ponto é usada para descrever nomes de atributos de tipo de dados de objetos ou tipos de relacionamentos. Por exemplo, o tipo de dado objeto Gene está implicitamente associado com duas percepções: LaboratoryA e LaboratoryB através duas percepções definida como Gene como ilustrado na Figura 4.14. A associação explícita é feita através do uso de linhas pontilhadas como representada na Figura 4.14 entre atributo de Publication e a percepção Gene.Publication. Uma associação explícita indica que um elemento de esquema de dados (atributo Publication) somente existe na percepção associada (percepção Gene.Publication) enquanto uma associação implícita, quando existe, denota uma nova percepção do elemento do esquema de dados na percepção associada.

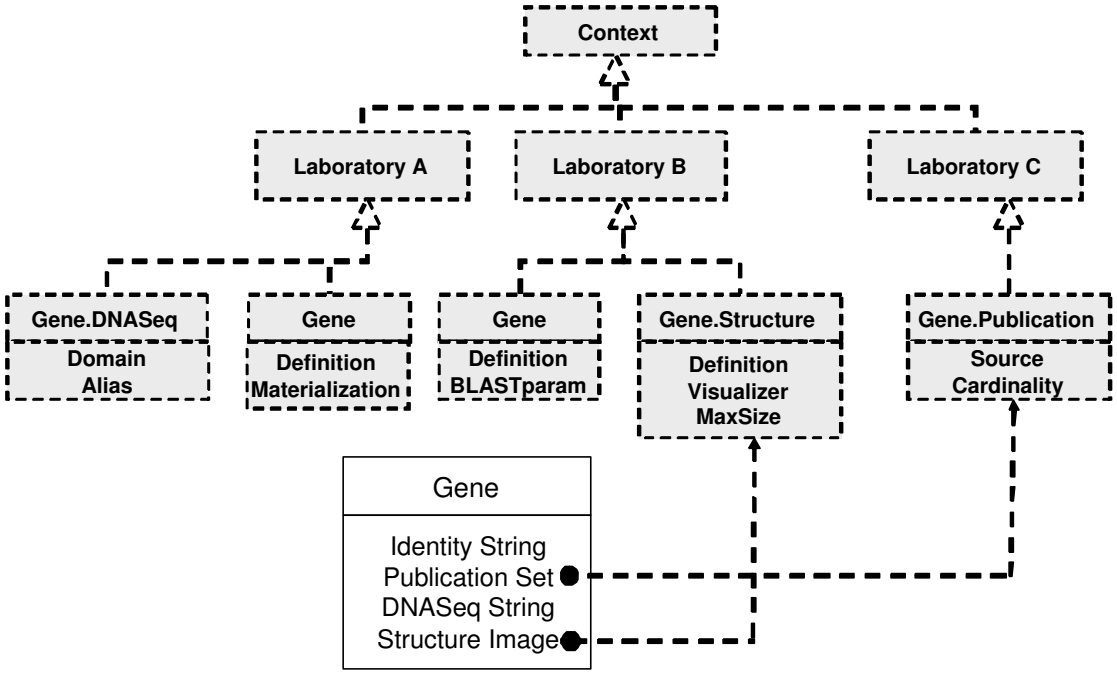

Figura 4.14: Usando percepções para associar a diferentes representações com o mesmo tipo de objeto Gene.

4.10.2 Propriedades da Percepção

O conceito de percepção no *BioConceptual* é definido como um conjunto de propriedades. Cada propriedade é representada por uma expressão que pode ser um restrição, domínio ou especificação de valor. Existem dois tipos de propriedades: definidas pelo usuário e definidas pelo sistema. Propriedades definidas pelo sistema são prefixadas com caracter '$' e seu propósito é alterar as características do atual esquema de dados. Por exemplo, na Figura 4.14 o atributo ADNSeq tem seu nome e domínio redefinida na percepção Gene.ADNSeq. Essas propriedades pode ser definidas como apresentado abaixo:

$$\$Domain = Bitmap;$$
$$\$Alias = Sequence;$$

BioConceptual define diversos atributos de forma a facilitar a modificação dos elementos de um esquema de dados de acordo com a percepções necessárias. A Tabela 4.2 apresenta as propriedades definidas no *BioConceptual* , seus objetivos e exemplos de uso.

Propriedades definidas pelo usuário são expressões que podem ser referenciadas pelas aplicações que usam este esquema de dados. Por exemplo, na Figura 4.14 a propriedade *BLASTparam* é uma propriedade definida pelo usuário para a percepção Gene, a qual estipula quais são os parâmetros do BLAST que devem ser usados quando a percepção LaboratoryB é definida. Neste caso particular, a percepção LaboratoryB armazena informação que pode ser usada pela aplicação que executa um programa BLAST.

Tabela 4.2: Exemplos de propriedades definidas pelo sistemas

Propriedade definida pelo sistema	**Objetivo**	**Exemplo**
$Definition	Especifica a definição de uma percepção	$Definition = Uma unidade que carrega informação para a biosíntese de um produto específico na célula
$DefinitionRef	Especifica um conjunto de referências de definição	$DefinitionRef = {ISBN:1234, EC:1.2.7.2, GO:0018482}
$Alias	Altera um tipo de objeto, atributo ou nome do tipo de relacionamento	$Alias = Sequence
$Cardinality	Especifica cardinalidade de um tipo objeto, atributo e tipo de relacionamento	$Cardinality = (0,N)
$Domain	Define um novo domínio de atributo	$Domain = Bitmap

4.10.3 Identidade de Percepções e Instâncias de Percepções

Cada contexto tem um único identificador, o qual é escolhido a partir de espaço de percepções. A idéia geral da identidade da percepção é que toda instância possui um atributo adicional com a identidade da percepção. Por isso, cada instância do atributo no *BioConceptual* é representada por um tipo, um conjunto de valores ou uma percepção. Por exemplo, a Figura 4.15 mostra as três possíveis instâncias do tipo Gene que variam de acordo com diferentes percepções (e.g. Laboratory A, Laboratory B e Laboratory C). Cada instância de atributo tem um identificador de percepção associado. Associando identidade de percepções a atributos permite resolver problemas relacionados com heterogeneidade semântica. Por exemplo, embora o atributo ADNSeq tenha diferentes nomes e domínios na percepção LaboratoryA, é possível saber que este atributo armazena a mesma informação sobre sequências de ADN em todas as diferentes percepções.

4.10.4 Referenciando Percepções

As identidades de contexto são importantes porque elas permitem ser referenciadas durante a população e consulta ao banco de dados. Por exemplo, a Tabela 4.3 ilustra três comando DML que inserem três tuplas no tipo de dado objeto Gene. Neste exemplo, os usuários podem indicar em qual percepção cada tupla será inserida. Neste exemplo, o símbolo '@' é usado para prefixar a identidade da percepção.

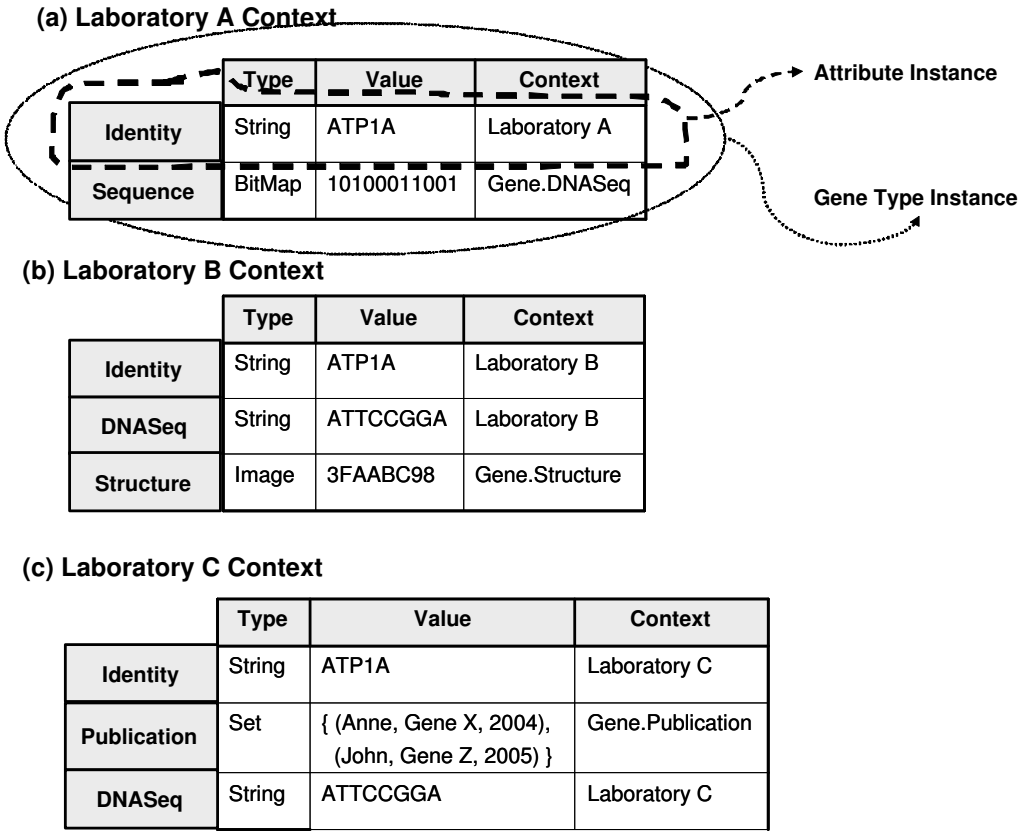

Figura 4.15: Três tipos de instâncias que variam de acordo com diferentes percepções

Tabela 4.3: Populando o banco de dados usando diferentes percepções

Insert into Gene@LaboratoryA (Identity, Sequence) values ('ATP1A',10100011001)	
Insert into Gene@LaboratoryB (Identity, ADNSeq, Structure) values ('ATP1A','ATTCCGGA', 3FAABC98)	
Insert into Gene@LaboratoryC (Identity, ADNSeq, Structure) values ('ATP1A',(Anne, Gene X, 2004),(John, Gene Z, 2005),'ATTCCGGA',)	