

TÍTULO: ESTENDENDO AS ABSTRAÇÕES DE
GENERALIZAÇÃO E SUBCONJUNTO NO MODELO
ENTIDADE RELACIONAMENTO

AUTOR: Luiz Tucherman e Marco Antonio Casanova

ENDEREÇO PARA CONTATO: Centro Científico Rio - IBM Brasil
Caixa Postal 4624
20.001, Rio de Janeiro, RJ.

RESUMO: Este trabalho propõe extensões das abstrações de generalização e subconjunto propostas na literatura e que são muito utilizadas na modelagem conceitual de bancos de dados. Este trabalho apresenta também algumas otimizações para esquemas entidade-relacionamento que utilizam estes tipos de abstrações.

PALAVRAS-CHAVE: modelo Entidade-Relacionamento, hierarquias de generalização e subconjunto, projeto lógico de banco de dados, otimização de esquemas.

ÁREA E SUB-ÁREA: software e software para banco de dados.

PÚBLICO ALVO: Pesquisadores em Banco de Dados, projetistas de banco de dados e administradores de banco de dados.

1. INTRODUÇÃO

O projeto de bancos de dados é tipicamente dividido em duas fases, denominadas de conceitual e física, respectivamente. O objetivo da fase conceitual é obter uma descrição de alto nível do banco de dados, a qual deve ser a mais independente possível dos detalhes de implementação. O resultado desta descrição é chamado de esquema conceitual do banco de dados. Em oposição, a fase de projeto físico se concentra na escolha da representação interna das estruturas conceituais.

Entre as principais abstrações usadas na fase de projeto conceitual encontram-se as hierarquias de generalização e de subconjunto. A maioria das definições encontradas na literatura para a hierarquia de generalização se baseiam na existência de uma condição de seleção (predicado) que divide, de acordo com o resultado desta seleção, uma classe em várias subclasses. Esta divisão pode particionar exaustivamente uma classe em todas as suas possíveis subclasses.

Para a hierarquia de subconjunto, a maioria das definições não dependem de nenhuma condição de seleção e estabelecem que a especialização de uma classe de entidade E não implica na partição de E . Isto significa que alguma entidade em E pode não estar em qualquer uma das suas subclasses. Assim, a única condição imposta por este tipo de hierarquia é que todas as entidades de uma subclasse devem também ocorrer em E . Alguns autores também diferenciam a hierarquia de subconjunto da hierarquia de generalização pela possibilidade da existência de sobreposição ou não dos subconjuntos, respectivamente.

Este trabalho propõe dois construtores que juntos capturam e estendem as abstrações de generalização e subconjunto que são propostas na literatura, e investiga algumas possíveis otimizações em esquemas conceituais utilizando os construtores propostas.

Diversas extensões ao Modelo Entidade-Relacionamento original [5] encontram-se em [3,4,6,7,10,11,12]. Em particular, propostas que capturam hierarquias de generalização e de subconjuntos encontram-se em [3,6,8,9,11,12].

Este trabalho está dividido da seguinte maneira. A seção 2 descreve a sintaxe dos construtores. Na seção 3 apresentamos exemplos do uso dos construtores.

A seção 4 discute otimizações e verificações de consistência para esquemas entidade-relacionamento definidos com estes construtores, e finalmente a seção 5 contém as conclusões.

2. CONSTRUTORES PARA CAPTURAR HIERARQUIAS DE GENERALIZAÇÃO E SUBCONJUNTO

Esta seção apresenta uma coleção de construtores que juntos capturam e estendem a maioria das formas propostas para as hierarquias de generalização e subconjunto. Inclui também, por uma questão de completude, a descrição de um número mínimo de construtores adicionais.

2.1 Sintaxe da linguagem

2.1.1 Declaração de um esquema ER

Um esquema conceitual entidade-relacionamento é descrito por uma *declaração de esquema-ER*, da seguinte forma:

```
define esquema S
    L1; ...; Lk;
fimesquema
```

onde *S* é o *nome* e L_1, \dots, L_k é a lista de *comandos* do esquema-ER.

2.1.2 Declaração de domínios

Os tipos de domínios dos atributos são baseados nos domínios existentes nas linguagens de definição de dados do modelo relacional.

Assim, um *tipo básico de domínio* é qualquer expressão da forma:

```
T [multivalorado] [nao nulo]
```

onde *T* é qualquer uma das seguintes expressões:

```
decimal(m,n)          flutuante
inteiro                caracter(p)
```

caractvar(p)

caractvar_longo

onde m e n são inteiros, com $m < n$, e p é um inteiro positivo menor do que 254.

Se "multivalorado" é especificado, dizemos que o tipo do domínio é *multivalorado*; caso contrário, dizemos que ele é *unicamente-valorado*. Se "nao nulo" é especificado dizemos que o tipo do domínio *não admite valores nulos*; caso contrário dizemos que ele *admite valores indefinidos*.

Entretanto, assumimos que:

A0. um domínio multivalorado nunca admite valores nulos.

Isto é, não é necessário especificar "nao nulo" juntamente com "multivalorado". Esta premissa é justificada por evitar a manipulação de valores nulos nos domínios multivalorados, que é mais complicada do que para os domínios unicamente-valorados.

Exemplos de expressões de tipos básicos de domínios (note que por A0 os dois últimos exemplos são equivalentes):

```
inteiro
decimal(10,2) nao nulo
caracter(20) multivalorado
caracter(20) multivalorado nao nulo
```

Permitimos também que os tipos de domínios sejam nomeados por *declaração de domínios* da seguinte forma:

```
defina dominio  $D_1 B_1, \dots, D_n B_n$ 
```

onde, para $i=1, \dots, n$, B_i é um tipo básico do domínio e D_i é um *nome* para B_i . Dizemos que D_i é multivalorado, etc., se B_i também for.

Exigimos que:

A1. duas declarações de domínio em um esquema-ER não podem especificar o mesmo nome.

O uso de nomes para os tipos básicos dos domínios é uma maneira conveniente de se impor uniformidade e também obter flexibilidade. Por exemplo, declarando

```
defina dominio CEP_TIPO_DOMINIO decimal(5,0)
```

e usando CEP_TIPO_DOMINIO em todo o esquema-ER, todos os códigos postais parecerão os mesmos. Além disso, se o tipo do domínio do código postal tiver de ser modificado, será necessário somente alterar um único comando e recompilar o esquema-ER.

Um *tipo de domínio válido* de um esquema-ER S é ou um tipo básico de domínio ou um nome para o tipo de domínio declarado em S.

2.1.3 Declarações de Esquemas de Entidades

A definição de um conjunto de entidades é bastante padronizada. Permitimos a especificação de um conjunto de chaves alternativas, em adição a uma chave primária. A chave primária, as chaves alternadas e mesmo a lista de atributos são em princípio opcionais. Mas, assumindo que desejamos selecionar cada entidade por um dos seus valores de chave, esta liberalidade somente faz sentido quando um esquema de entidade especializa um outro esquema de entidade do qual ele herda suas chaves e a lista de atributos (veja na seção 2.1.5 o conceito de especialização). Finalmente, permitimos ao projetista do banco de dados definir que os valores dos atributos de cada entidade em um dado conjunto devem satisfazer certas restrições de integridade. Esta facilidade está em linha com a noção de especialização qualificada (veja seção 2.1.5 e 4).

Portanto, introduzimos uma *declaração de esquema de entidade* como qualquer expressão da seguinte forma:

```
defina entidade E
  [atributos A1 D1, ..., An Dn]
  [chave K0] ... [chave Kp]
  [verifique Q0] ... [verifique Qq]
```

Dizemos que E é o *nome*, A₁, ..., A_n é a lista de *nomes de atributos*, K₀ é a *chave primária* e K₁, ..., K_p são as *chaves alternativas* do esquema. Dizemos também que

D_i é o tipo do domínio de A_i , para $i=1,\dots,n$, e que A_i é *multivalorado* ou *aceita valores nulos* se D_i também o for. Finalmente, dizemos que Q_0,\dots,Q_q são as *assertivas* definidas para E .

Exigimos que:

- E0. E deve ser um nome válido e distinto de todos os outros nomes de esquemas de entidade ou relacionamento do esquema-ER;
- E1. A_1,\dots,A_n devem ser nomes válidos e distintos de atributos;
- E2. D_1,\dots,D_n devem ser tipos de domínios válidos no esquema-ER;
- E3. para cada i em $[0,p]$, K_i deve ser uma lista de nomes de atributos de E tal que
 - E3.0. todos os nomes de atributos em K_i são distintos;
 - E3.1. cada nome de atributo em K_i é unicamente-valorado e não admite valores nulos;
 - E3.2. K_i não está contido em K_j , para qualquer j em $[0,p]$ com $i \neq j$;
- E4. Q_0,\dots,Q_q devem ser qualificações sobre atributos do esquema (herdados ou não, veja seção 2.1.5).

A sintaxe exata da qualificação foi deixada indefinida. Dizemos que uma entidade e *satisfaz* uma qualificação Q_i quando os valores dos atributos de e fazem com que Q_i seja verdadeiro.

2.1.4 Declarações de Esquemas de Relacionamentos

Antes de introduzirmos o construtor que descreve conjuntos de relacionamentos, convém fazer algumas considerações. Primeiro, permitimos definir conjuntos de relacionamentos sobre conjuntos de entidades ou de relacionamentos. Deixamos também que um conjunto de entidades ou de relacionamentos participe mais de uma vez de um conjunto de relacionamentos, desde que um nome de papel distinto seja designado para cada ocorrência do conjunto participante. Permitimos que um conjunto de relacionamento seja opcionalmente definido como total [12] em relação ao papel de um conjunto de entidade ou de relacionamento. Introduzimos também a noção de identificador para os conjuntos de relacionamentos como um correspondente às chaves. Escolhemos o termo 'identificador' para chamar a atenção para o fato de que agora podemos ter uma lista de participantes no relacionamento, e não uma lista de atributos. E, por último, permitimos que um relacionamento seja definido com mais de um

identificador, uma liberalidade que é necessária, por exemplo, para a definição de relacionamentos binários 1-1.

Os identificadores capturam a anotação usual que rotula com "1" ou "n" os arcos que deixam o losango que representa um conjunto de relacionamento em um diagrama ER, mas eles são mais gerais do que esta convenção de anotação.

Assim como para os esquemas de entidades, podemos omitir o identificador primário, os identificadores alternativos e toda a lista de atributos. Da mesma forma, o identificador primário é sempre o primeiro a ser especificado.

Uma *declaração de um esquema de relacionamento* é um construtor da forma:

```
defina relacionamento R
  sobre  $O_1$  [como  $N_1$ ] [total], ...,  $O_m$  [como  $N_m$ ] [total]
  [atributos  $A_1$   $D_1$ , ...,  $A_n$   $D_n$ ]
  [identificador  $I_0$ ] ... [identificador  $I_p$ ]
  [verifique  $Q_0$ ] ... [verifique  $Q_q$ ]
```

Dizemos que R é o *nome*, A_1, \dots, A_n é a lista de *nomes de atributos*, I_0 é o *identificador primário* e I_1, \dots, I_p são os *identificadores alternativos* do esquema. Dizemos também que o esquema de relacionamento R tem m *papéis*. O esquema no $i^{\text{ésimo}}$ papel é O_i e o *nome* do $i^{\text{ésimo}}$ papel é N_i , se especificado, ou o próprio O_i , caso contrário. Quando "total" é especificado para " O_i [como N_i]", dizemos que R é *total* no $i^{\text{ésimo}}$ papel do esquema.

A partir de agora, quando nos referirmos a um esquema queremos dizer ou um esquema de entidade ou um esquema de relacionamento.

Exigimos que:

R0. R deve ser um nome válido e distinto de todos os outros nomes de esquemas do esquema-ER;

R1, R2. (como E1 e E2);

R3. O_1, \dots, O_m devem ser nomes de esquemas válidos, não necessariamente distintos, mas definidos no esquema-ER, com $m \geq 2$;

R4. os nomes dos papéis de R devem ser distintos;

R5. para cada i em $[0, p]$, I_i deve ser uma lista de nome de papéis de R tal que

R5.0. todos os nomes de papéis em I_i são distintos;

R5.1. I_i não está contido em I_j , para qualquer j em $[0,p]$ com $i \neq j$;

R6. (como E4).

Note que, por R4, os nomes dos papéis são únicos. Portanto, podemos nos referir a um papel específico de R pelo seu nome ao invés de sua posição.

Como permitimos relacionamentos sobre relacionamentos, devemos também evitar definir um esquema de relacionamento R que seja sobre si mesmo, uma vez que de outra maneira a semântica de R seria indefinida. Mais precisamente, exigimos adicionalmente que:

R7. o esquema-ER não pode ter uma cadeia de esquemas de relacionamentos $O_0, O_1, \dots, O_{n-1}, O_0$ tal que O_i é sobre O_{i+1} , para $i=0, \dots, n-1$, onde '+' é módulo n.

2.1.5 Declaração de Especializações

Permitimos ao projetista do banco de dados organizar os esquemas de entidades e de relacionamentos de um dado esquema-ER como um grafo de especialização com as seguintes características.

Em primeiro lugar, o grafo de especialização não necessita ser uma árvore, isto é, uma hierarquia estrita. Na verdade, nem exigimos que o grafo seja acíclico, embora argumentaremos na seção 3 que os grafos de especialização cíclicos devem ser evitados.

Um esquema de entidade E pode ser definido como uma especialização *qualificada* de outro esquema de entidade F, com qualificação C, o que implica que E denotará sempre o conjunto de entidades de F que satisfazem C. Por conseguinte, exigimos que um esquema de entidade E seja uma especialização qualificada de no máximo um outro esquema de entidade.

Um esquema de entidade E pode também ser definido como uma especialização *simples* de F, indicando que E deve sempre denotar um subconjunto do conjunto de entidades associada com F. Assim, um esquema de entidade pode ser uma especialização simples de mais de um esquema e, talvez ao mesmo tempo, ser uma especialização qualificada de um esquema de entidade.

Permitimos também definir um esquema de relacionamento como uma especialização simples de um outro esquema de relacionamento. Proibimos para este caso especializações qualificadas porque elas iriam criar interrelacionamentos complexos entre os conjuntos de entidades e de relacionamentos, que seriam difíceis de garantir. Exigimos que, se R é um esquema de relacionamento sobre P_1, \dots, P_n e S é um esquema de relacionamento sobre Q_1, \dots, Q_n tal que S é uma especialização de R , então Q_i ou é igual a ou é uma especialização de P_i . Esta exigência garante que é semanticamente possível comparar S and R .

Introduzimos também o conceito de totalidade e exclusão mútua para especializações simples. Se declararmos que O é *totalmente especializado* em O_1, \dots, O_n , então qualquer objeto em O deve pertencer a O_i , para algum i . Ortogonalmente, se declararmos que, O_1, \dots, O_n são *mutuamente exclusivos* então nenhum objeto em O pode pertencer a mais do que um O_i , para todo i . Estas são então restrições de integridade que relacionam O, O_1, \dots, O_n .

Permitimos especificar totalidade e exclusão mútua apenas para as especializações simples pelas seguintes razões. Primeiro, como especializações qualificadas definem conjuntos de objetos baseados nos valores dos atributos, o projetista do banco de dados pode, para este caso, expressar a exclusão mútua através do uso de uma definição apropriada para as qualificações e a totalidade definindo uma assertiva para o esquema generalizado. Entretanto, não podemos aplicar a mesma estratégia para especializações simples uma vez que os conjuntos de objetos neste segundo caso são definidos "manualmente", ou seja, quando o usuário adiciona os objetos nos seus conjuntos.

Finalmente, permitimos especificar no esquema-ER mais de uma declaração de especialização para o mesmo esquema O , mas não permitimos existir mais de uma declaração indicando que um esquema O é uma especialização de um esquema P .

Mais precisamente, introduzimos uma *declaração de especialização* como qualquer expressão da forma:

especialize O [totalmente] [exclusivamente]
em O_1 [onde C_1], ..., O_m [onde C_m]

Dizemos na declaração que O é o esquema *especializado* e que O_i é uma *especialização simples*, se a cláusula onde C_i não for especificada, e que O_i é uma *especialização qualificada* em caso contrário. Também dizemos que O é uma *generalização* de O_1, \dots, O_m , que é total se "totalmente" for especificado. Da mesma forma, se "exclusivamente" for especificado, dizemos que O_1, \dots, O_m são especializações *mutuamente exclusivas* de O .

Antes de prosseguirmos, introduziremos algumas definições auxiliares. Seja S um esquema-ER. O *grafo de especialização* de S , $G(S) = (V, A)$, é um grafo dirigido tal que V é o conjunto de nomes de esquemas de S e um par (O, P) está em A se e somente se O é uma especialização simples ou qualificada de P em S .

Dado $F, G \in V$, dizemos que G é uma *especialização transitiva* de F e que F é uma *generalização transitiva* de G se e somente se existe um caminho de G para F em $G(S)$; dizemos que G é uma *especialização qualificada transitiva* de G se e somente se existe um caminho de G para F em $G(S)$ tal que, para todos os arcos (H, I) do caminho, H é uma especialização qualificada de I .

Finalmente, dado $E, F, G \in V$, dizemos que E está *entre* G e F se e somente se G é uma especialização transitiva de E e E é uma especialização transitiva de F .

Dizemos que um atributo A (ou uma chave K) é *nativo* em O se e somente se A é o nome de um atributo (ou K é uma chave) definida no esquema cujo nome é O . Dizemos que um atributo A (ou uma chave K) é *herdado por O de P* se e somente se A é o nome de um atributo de P (ou K é a chave de P) e O é uma especialização transitiva de P . Note que devemos especificar o esquema originário de A (ou K) uma vez que O pode ser uma especialização transitiva de esquemas cujos atributos podem ter o mesmo nome (ou cujas chaves podem ser iguais). Assim, no contexto do esquema O e sempre que necessário, usaremos a notação $P.A$ (ou $P.K$) para nos referirmos ao atributo (ou chave) herdado por O de P .

Para a especialização exigimos que:

- S0. O, O_1, \dots, O_m sejam nomes distintos de esquemas definidos no esquema-ER;
- S1. para cada i em $[1, m]$, C_i , se especificado, deve ser uma qualificação sobre atributos de O , envolvendo pelo menos um atributo nativo de O ;

- S2. para cada i em $[1,m]$, não pode existir outra declaração de especialização indicando que O_i é uma especialização de O ;
- S3. para cada i em $[1,m]$, se O_i é uma especialização qualificada de O então O_i não pode ser uma especialização qualificada de qualquer outro esquema de entidade;
- S4. se 'totalmente' ou 'exclusivamente' são especificados então O_1, \dots, O_m devem ser especializações simples;
- S5. se O é o nome de um esquema de relacionamento então nenhuma qualificação pode ser especificada;
- S6. se O é o nome de um esquema de relacionamento sobre P_1, \dots, P_n então, para cada i em $[1,m]$, O_i deve ser o nome de um esquema de relacionamento sobre esquemas Q_1, \dots, Q_n tal que O_i ou é igual a ou é uma especialização transitiva de P_j , para cada j em $[1,n]$.

Para o Requisito S1, consideramos que C_i deve envolver pelo menos um atributo nativo de O uma vez que, em caso contrário, O_i poderia ser redefinido como uma especialização qualificada de alguma generalização transitiva de O , isto é, o esquema onde todos os atributos em C_i são definidos. Em vista de S4, é necessário especificar duas declarações separadas de especialização para o esquema O quando as especializações simples de O modelam uma classificação total ou exclusiva de O , mas O possui ainda uma ou mais especializações qualificadas.

2.1.6 Declaração de Exclusão Mútua

Esta seção introduz uma forma bem geral para se definir que vários conjuntos de objetos são mutuamente exclusivos, generalizando assim a palavra chave *exclusivamente* da declaração de especialização.

Em geral, o projetista do banco de dados pode indicar que dois esquemas O e P devem sempre denotar conjuntos disjuntos de objetos, desde que as seguintes condições sejam satisfeitas. Primeiro, O e P devem ser comparáveis, o que pode ser formalizado exigindo-se que O e P especializem transitivamente um terceiro esquema Q . Segundo, exigimos que O e P não sejam especializações qualificadas de outros esquemas O' e P' uma vez que a exclusão mútua neste caso criaria iterações complexas com as qualificações.

Assim, uma *declaração de exclusão mútua* é qualquer expressão da forma

exclua $(0_{11}, \dots, 0_{1n_1}) \mid \dots \mid (0_{k1}, \dots, 0_{kn_k})$

Para cada $p, r \in [1, k]$, com $p \neq r$, para cada $q \in [1, n_p]$ e cada $s \in [1, n_r]$, dizemos que 0_{pq} and 0_{rs} são *mutuamente exclusivos*.

Exigimos que:

M0. $0_{11}, \dots, 0_{1n_1}, \dots, 0_{k1}, \dots, 0_{kn_k}$ devem ser nomes de esquemas definidos no esquema-ER e que não sejam especializações qualificadas de qualquer esquema;

M1. exista um esquema Q tal que $0_{11}, \dots, 0_{1n_1}, \dots, 0_{k1}, \dots, 0_{kn_k}$ são especializações transitivas de Q ;

M2. 0_{pq} e 0_{rs} devem ser distintos, para cada $p, r \in [1, k]$, com $p \neq r$, para cada $q \in [1, n_p]$ e cada $s \in [1, n_r]$.

3. UM EXEMPLO INFORMAL

Ilustraremos nesta seção o uso dos construtores apresentados na seção 2. Considere o diagrama ER (figura 1) representando um fragmento do esquema conceitual apresentado em [1]:

No diagrama ER acima citado, a hierarquia de generalização da entidade TRABALHO resulta em dois conjuntos disjuntos TRABALHO_REJEITADO e TRABALHO_SELECIONADO, que são produzidos particionando TRABALHO a partir dos dois possíveis valores do atributo SUBMISSAO. A hierarquia de subconjunto da classe de entidade ESPECIALISTA resulta em dois subconjuntos que se sobrepõe, MEMBRO e REVISOR, que podem ser produzidos através do particionamento da classe genérica ESPECIALISTA ou pelos valores de diferentes atributos ou baseados na aplicação.

Podemos definir estes dois tipos de hierarquias de diferentes maneiras, uma vez que não distinguimos a hierarquia de generalização e a hierarquia de subconjuntos, usando a declaração de especialização.

Para definir a hierarquia de generalização de TRABALHO, usaremos a seguinte declaração de especialização qualificada:

especialize TRABALHO

em

```
TRABALHO_REJEITADO onde SUBMISSAO = 'REJEITADO',  
TRABALHO_SELECIONADO onde SUBMISSAO = 'SELECIONADO';
```

Uma vez que a maioria das definições de hierarquia de generalização implica em totalidade e exclusão mútua das subclasses e desde que totalmente não pode ser usado em uma declaração de qualificação, somos levados a especificar na declaração do esquema da entidade TRABALHO a seguinte assertiva, que enumera todos os possíveis valores para o atributo SUBMISSAO:

...

```
verifique SUBMISSAO = 'REJEITADO' ou SUBMISSAO = 'SELECIONADO'
```

...

Em geral, quando a cláusula onde de uma declaração de especialização qualificada nunca é simultaneamente satisfeita, ela define subclasses exclusivas.

A hierarquia de subconjunto de ESPECIALISTA pode ser definida de duas maneiras diferentes dependendo da existência de algum atributo que pode ser usado para classificar as instâncias de ESPECIALISTA nas subclasses MEMBRO e REVISOR. Assumindo a existência dos atributos MEMBRO e REVISOR na entidade ESPECIALISTA, a seguinte declaração de especialização qualificada captura a hierarquia desejada:

especialize ESPECIALISTA

```
em MEMBRO onde MEMBRO = 'SIM',  
REVISOR onde REVISOR = 'SIM';
```

De outra maneira, se estes atributos não existirem podemos definir a hierarquia de subconjunto de ESPECIALISTA através de uma declaração de especialização simples:

especialize ESPECIALISTA

```
em MEMBRO, REVISOR;
```

Note que nem a opção totalmente nem a opção exclusivamente foram especificadas porque, para a hierarquia de subconjuntos, conforme definido na literatura, podemos ter instâncias da classe genérica que não pertencem a

qualquer subclasse e, além disso, podemos ter classes que se sobrepõe. No exemplo anterior, podemos ter instâncias de ESPECIALISTA que pertencem a ambas subclasses, MEMBRO e REVISOR, a uma delas ou a nenhuma delas.

Entretanto, podemos ir além e combinar as hierarquias de especialização simples e qualificadas de forma a capturar conceitos bastantes complexos do mundo real. Vamos dar um exemplo.

Neste exemplo (figura 2), classificamos EMPREGADO em duas especializações qualificadas, ALTAMENTE_GRADUADO e ALTAMENTE_ESPECIALIZADO, e em uma especialização simples, INTERNO. Além disso, categorizamos INSTRUTOR em duas especializações qualificadas, INTERNO e EXTERNO, que são subconjuntos disjuntos. Assim, INTERNO representa um empregado que ensina algum curso. As declarações de especialização apropriadas são:

```
especialize INSTRUTOR
```

```
  em  EXTERNO  onde TIPO = 'EXTERNO',  
      INTERNO  onde TIPO = 'INTERNO';
```

```
especialize EMPREGADO
```

```
  em  INTERNO,  
      ALTAMENTE_GRADUADO      onde EDUCACAO = 'PHD' ou  
                               EDUCACAO = 'MSC',  
      ALTAMENTE_ESPECIALIZADO onde ESPECIALIZACAO = 'TECNICA' e  
                               EXPERIENCIA >= 10;
```

Observe que podemos ter instâncias de EMPREGADO que não são instâncias das duas especializações qualificadas, mas que são instâncias de INTERNO. Além disso, podemos também ter instâncias de EMPREGADO que não são classificados em qualquer das três subclasses.

4. RESTRIÇÕES E OTIMIZAÇÕES PARA AS DECLARAÇÕES DE ESPECIALIZAÇÃO E EXCLUSÃO MUTUA

Discutiremos nesta seção vários casos de redundância e inconsistência que podem ocorrer em um esquema-ER como consequência da definição de declarações de especialização e exclusão mútua.

4.1 Restrições adicionais

Informalmente, dizemos que um esquema-ER S é *redundante* se e somente se o esquema tem comandos que são parcialmente implicados por outros e dizemos que um esquema é *mal-definido* se e somente se todos os estados consistentes de S , ou seja, todos os estados do esquema que satisfazem todas as restrições de integridade, atribuem o conjunto vazio a algum esquema de S .

Denotaremos por $I(O)$ o conjunto de objetos associados a um esquema O por um estado do banco de dados. Seja S um esquema-ER e seja $G(S) = (V, A)$, um grafo de especialização de S . Podemos provar que:

Proposição 1: Seja I um estado consistente de S . Se existir um caminho de O para P em $G(S)$, então $I(O) \subseteq I(P)$.

Observe então que todos os esquemas participantes em um ciclo (O_0, \dots, O_n, O_0) de $G(S)$ denotam sempre o mesmo conjunto de objetos em qualquer estado consistente uma vez que, pela proposição (1), teríamos, $I(O_0) \subseteq \dots \subseteq I(O_n) \subseteq I(O_0)$. Portanto, podemos otimizar S colapsando O_0, \dots, O_n em um único esquema O que tem todos os atributos originalmente definidos para cada esquema colapsado e modificando a definição dos outros comandos do esquema-ER de forma a refletir tal transformação.

Entretanto, este tipo de ação tem consequências profundas na estrutura do esquema-ER e na semântica das operações. Portanto, ela é abandonada, exigindo-se que:

G1. $G(S)$ deve ser acíclico.

Similarmente, se S especifica que O e P são mutuamente exclusivos e existe um caminho em $G(S)$ de O para P então S é mal-definido uma vez que $I(O)$ deve ser vazio, para qualquer estado consistente I de S . Da mesma forma, se O e P são mutuamente exclusivos e tem em comum uma especialização transitiva Q , então S é mal-definido já que $I(Q)$ deve ser vazio, para qualquer estado consistente I de S . Portanto, exigimos adicionalmente que:

G2. se S especifica que O e P são mutuamente exclusivos então não deve existir nenhum caminho em $G(S)$ de O para P .

G3. se S especifica que O e P são mutuamente exclusivos então não pode existir nenhuma especialização transitiva comum a O e P .

Todas as condições discutidas acima são facilmente verificadas pois elas se reduzem a descobrir caminhos em um grafo. Vamos discutir em seguida alguns critérios adicionais que dependem da análise das qualificações e assertivas.

Seja α um estado consistente de um esquema-ER S . Seja O um esquema de S e o um objeto de $I(O)$. Então, o satisfaz todas as assertivas associadas a O . Além disso, se O é uma especialização simples de P , o também deve pertencer a $I(P)$, o que implica que o também satisfaz todas as assertivas associadas a P . Em adição, se O é uma especialização qualificada de P com qualificação C , o também satisfaz C . Todas estas observações podem ser capturadas da seguinte maneira.

Seja S um esquema-ER e assumamos que $G(S)$ é acíclico. Para cada esquema O de S , defina $\bar{Q}(O)$ como a seguir:

$$\bar{Q}(O) = C, \quad \text{se existir um esquema } P \text{ tal que } O \text{ é uma especialização qualificada de } P \text{ com qualificação } C$$

$$\bar{Q}(O) = \textit{verdadeiro}, \quad \text{caso contrário}$$

Uma vez que um esquema pode ser uma especialização qualificada de no máximo um outro esquema, \bar{Q} é bem definido. Definimos também $\bar{A}(O)$ como a conjunção de todas as assertivas definidas para O ou induzidas por aquelas definidas para os domínios usados pelos atributos de O .

O *grafo de especialização aumentado* de S , $H(S) = (V, A, m)$, é um grafo dirigido com nós rotulados tal que V é o conjunto dos nomes de esquemas de S e um par (O, P) está em A se e somente se O é uma especialização simples ou qualificada de P em S . A função m que rotula os nós é definida da seguinte maneira:

- 1) Se O tem grau-saida 0 então $m(O) = \bar{A}(O)$;
- 2) Se O tem grau-saida maior do que 0 e os arcos que deixam O são $(O, O_1), \dots, (O, O_n)$ (isto é, se O especializa O_1, \dots, O_n) então $m(O) = (m(O_1) \wedge \dots \wedge m(O_n) \wedge \bar{Q}(O) \wedge \bar{A}(O))$

Observe que m é bem definido porque assumimos que $G(S)$, e por conseguinte $H(S)$, são acíclicos. Além disso, $m(O)$ é definido somente sobre os atributos de O

(e não sobre atributos de especializações transitivas de 0). Podemos então provar que:

Proposição 2: Seja I um estado consistente de S . Então todo $o \in I(0)$ satisfaz $m(0)$.

Nossa exigência final é:

G4. para nenhum esquema 0 definido em S , $m(0)$ é falso.

Note que esta última condição é muito mais difícil de implementar uma vez que ela requer um provador de teoremas para fazer frente às fórmulas que expressam as qualificações e as assertivas.

Finalmente, podemos mostrar que se levarmos totalidade em consideração ela não auxilia a detectar novas inconsistências.

4.2 Otimizações Simples

Vamos agora sugerir algumas transformações simples que otimizam um esquema-ER mas que devem ser vistas como parte da definição dos construtores da seção 2.5.

Usando o grafo de especialização, obtemos três otimizações simples:

Otimização de Especializações Simples:

Suponha que $(0,P)$ seja um arco em $G(S)$ correspondendo a uma especialização simples e que exista um caminho $(0,Q,\dots,P)$ em $G(S)$. Então, substitua S por um novo esquema-ER S' obtido eliminando-se de S a especificação que define 0 como uma especialização simples de P .

O novo esquema-ER S' é de fato equivalente a S porque o caminho alternativo de 0 para P já determina que $I(0)$ deve ser um subconjunto de $I(P)$ em qualquer estado consistente I , tornando portanto redundante a especialização simples.

Otimização de Especializações Qualificadas (1):

Seja $(0,P)$ um arco de $G(S)$ correspondendo a uma especialização qualificada de S com qualificação C . Suponha que existe um caminho em S da forma $(0,Q,\dots,P)$. Então, substitua S por um novo esquema-ER S' obtido:

- 1) eliminando-se de S a especificação que define O como uma especialização qualificada de P ;
- 2) definindo-se que O é agora uma especialização qualificada de Q com qualificação C' , obtida substituindo-se cada ocorrência de um atributo A em C por $P.A$, de forma a refletir que $P.A$ é agora um atributo de Q que foi herdado de P .

A correção sintática do novo esquema-ER S' segue primeiro porque Q herda todos os atributos de P , o que implica que C pode na verdade ser redefinido sobre Q . Segundo porque, pela restrição da linguagem, O não pode ser uma especialização qualificada de Q em S uma vez que O já é uma especialização qualificada de P . Portanto, o passo 2 pode na verdade ser executado sem violar as restrições da linguagem. Finalmente, S' é equivalente a S porque, em qualquer estado consistente I de S , a especialização qualificada requer que $I(O)$ seja o conjunto de objetos de $I(P)$ que satisfaça C ; mas o caminho alternativo de O para P determina que $I(O) \subseteq I(Q) \subseteq I(P)$; assim sendo, $I(O)$ é o conjunto de objetos em $I(Q)$ que satisfaz C .

Otimização de Exclusão Mutua (1):

Suponha que S especifique que O e Q são mutuamente exclusivos com P e que exista um caminho em $G(S)$ de O para Q . Então, substitua S por um novo esquema-ER S' obtido pela eliminação em S da especificação que define que O é mutuamente exclusivo com P .

Esta otimização está correta porque, em qualquer estado consistente I de S , a existência de tal caminho implica em que $I(O) \subseteq I(Q)$; assim sendo, $I(Q) \cap I(P) = \emptyset$ implica em $I(O) \cap I(P) = \emptyset$.

Seja S um esquema-ER e $H(S) = (V, A, m)$ o grafo de especialização aumentado de S . Usando $H(S)$, obtemos duas otimizações adicionais cuja correção resulta da proposição (2):

Otimização da Especialização Qualificada (2):

Se O é uma especialização qualificada de P , com qualificação C , e se $m(P)$ implica C , então substitua S por um novo esquema-ER S' obtido pelo colapsamento de O em P .

Otimização de Exclusão Mutua (2):

Se O e P são mutuamente exclusivos em S e a conjunção de $m(O)$ e $m(P)$ é sempre falsa, então substitua S por um novo esquema-ER S' obtido pela eliminação da especificação que indica que O e P são mutuamente exclusivos.

5. CONCLUSÕES

Os construtores apresentados na seção 2 formalizam e estendem as abstrações de generalização e subconjunto dentro do contexto do modelo entidade-relacionamento. Foram apresentadas também algumas otimizações adicionais que podem ser feitas sobre as hierarquias de especialização utilizadas em um esquema conceitual do banco de dados.

É importante chamar a atenção para a necessidade de se especificar o comportamento das operações sobre as extensões propostas. É apresentado em [13] um estudo detalhado de diversas alternativas para o desenvolvimento de uma linguagem de manipulação que leva em consideração tais abstrações.

BIBLIOGRAFIA

- [1] P. Bertaina, A. Di Leva, P. Giolito, Logical design in Codasyl and relational environment, in S. Ceri (ed.), Methodology and Tools for Data Base Design, North-Holland, 1983.
- [2] A. Borgida, "Features of languages for the development of Information Systems at the Conceptual level", IEEE Software 2(1), January 1985, 63-73.
- [3] U. Bussolati, S. Ceri, V. De Antonellis and B. Zonta, Views Conceptual Design, in S. Ceri (ed.) Methodology and Tools for Data Base Design, North-Holland, 1983.
- [4] S. Ceri (ed.), Methodology and tools for data base design, North-Holland, 1983.
- [5] P. P. Chen - The entity-relationship model: toward a unified view of data - ACM TODS, 1, 1 (1976) 9-36.
- [6] A. Dogac, P.P. Chen, Entity-Relationship Model in the ANSI/SPARC Framework, in Entity-Relationship Approach to System Analysis and Design, P.P. Chen (ed.), North-Holland (1981), 361-378.

- [7] R. Elmasri, J. Weeldreyer, A. Heuner, The Category Concept: An extension to the entity-relationship model, in *Data and Knowledge Engineering 1*, North-Holland, 1985, 75-116.
- [8] H.F. Korth and A. Silberschatz, *Database System Concepts*, McGraw-Hill Book Co. (1986).
- [9] W. Kozaczynski, L. Lillien, An Extended Entity-Relationship (E²R) Database Specification and its Automatic Verification and Translation into the Logical Relational Design, Proc. of the Sixth Int. Conf. on Entity-Relationship Approach, New York (Nov. 1987), 497-513.
- [10] M. Lenzerini, G. Santucci, Cardinality constraints in the E-R model, *Entity-Relationship Approach to Software Engineering*, North-Holland, 1983.
- [11] P. Scheuermann, G. Schiffner, H. Weber, Abstraction capabilities and invariant properties modelling within the Entity-Relationship approach, P.P. Chen (ed.) *Entity-Relationship approach to System Analysis and Design*, North-Holland, 1980.
- [12] T.J. Teorey, D. Yang, J.P. Fry, A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, *ACM Computing Survey*, 18, 2 (June 1986), 197-222.
- [13] L. Tucheran, M.A. Casanova, P.M. Gualandi, A. Pacheco, A Proposal for Formalizing and Extending the Generalization and Subset Abstractions in the Entity-Relationship Model, relatório técnico do Centro Científico Rio, IBM Brasil, a ser publicado.

ILUSTRAÇÕES

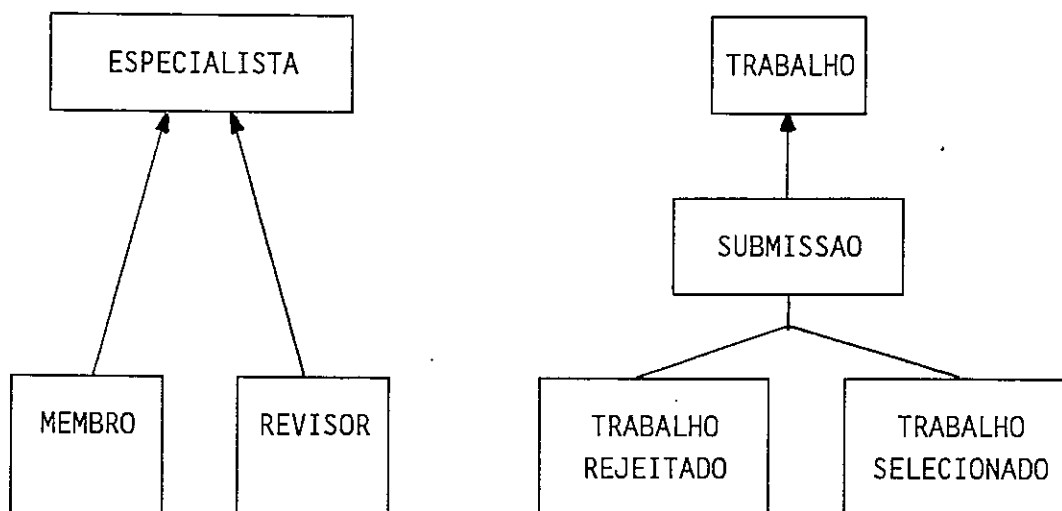


Figura 1

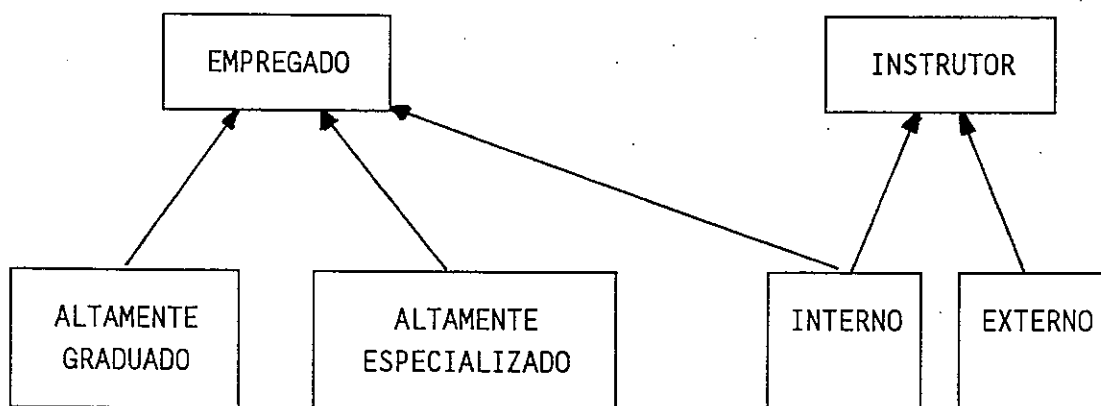


Figura 2
