STUDIA UNIV. "BABEŞ-BOLYAI", INFORMATICA, Volume I, Number 1, March 1996

UNE APPROCHE SUR LA MODÉLISATION D'HÉRITAGE DANS UN SYSTÈME ORIENTÉ-OBJET

G. MOLDOVAN AND C. BOBOILĂ

Résumé. Les domaines d'applications tels que: la conception assistée par ordinateur (CAO), la production de documents incorporant textes (images et graphique), le genie logiciel, nécessitent une représentation d'objets complexes.

L'approche orienté objet offre de nombreux avantages pour modéliser ces domaines d'applications. A partir de constructeurs tels que les constructeurs de n-uplet, ensemble, tableau, liste, et en imbriquant arbitrarement ceux-ci, les systèmes orientés-objet supportent la notion d'objet complexes avec identité d'objet.

Cet article présente un modèle de définition d'objets complexes avec identité d'objet au moyen des graphes. On définit les notions de type, valeur structurée, objet, la relation de sous-typage, le graphe des types, graphe de composition d'objets et graphe d'héritage des classes pour mettre en évidence les connexions inter-objet.

1. Introduction

Les domaines tels que: la conception assistée par ordinateur (CAO), la production de documents incorporant textes, images et graphiques, le génie logiciel, nécessitent la gestion d'une grande variété de types de données ainsi que les liens entre ces données (souvent imbriquées).

Le modèle relationnel [7] ne permet pas de modéliser efficacement ces types variés de données ainsi que leur imbrication. Dans ce modèle, les données sont représentées sous forme de relations "plates", c'est la contrainte de première forme normale. Les attributs d'un n-uplet étant nécessairement des valeurs atomiqurs (entiers, caractères, réels, booléens), il est difficile de représenter un objet complexe dans son ensemble.

Les premières évolutions du modèle relationnel ont consisté à enrichir son pouvoir de modélisation. La difficulté de représenter des structures complexes par des relations a

Received by the editors: October 9, 1996.

1991 *Mathematics Subject Classification.* 68P15, 68P05, 68Q65.

1991 *CR Categories and Descriptors.* H.2.1 [Database Management]: Logical Design – data models; H.2.3 [Database Management]: Languages – data description languages; D.3.2 [Programming Languages]: Language Classifications – object-oriented languages.

conduit à l'élaboration de modèles ne se restreignant plus à de seules valeurs atomiques, mais acceptant des valeurs structurées.

Plusieurs tentatives d'extension du modèle relationnel ont été effectuées au cours de ces dernières années [4].

Le rest du papier est organisé de la manière suivante: la Section 2 décrit un modèle à objets complexes avec identité d'objet, la Section 3 présente la rélation de sous-typage pour définir le graphe des types, le graphe de composition d'objets et le graphe d'héritage des classes, nous concluons ensuite Section 4.

2. Un modèle à objets complexes avec identité d'objet

Nous avons considéré les ensembles suivants. Un ensemble fini de domaines $D_1, \ldots, D_n$ $n \geq 1$ (par exemple l'ensemble Z de nombres entiers). Notons D l'union des domaines $D_1, \ldots, D_n$. Nous supposons que les domaines sont disjoints. Un ensemble dénombrable et infini A, qui s'appelle **univers d'attributs.** Les éléments de A sont des noms pour les champs de la structure. Un ensemble dénombrable et infini I d'identificateurs. Les éléments de I seront utilisés comme d'identificateurs pour d'objets.

Les types sont récursivement construits de la manière suivante:

Définition 2.1. *(Types) Soit A un univers d'attributs et T un ensemble de types prédéfinis (***integer, real, boolean, string, char***, etc).*

(i): *Tout élément de T est un type (dit* **atomique***).*

(ii): *Si $t_1, \ldots, t_n$ sont des types et $a_1, \ldots, a_n$ des attributs de A, alors $t = [a_1 : t_1, \ldots, a_n : t_n]$ est un type de structure n-uplet.*

(iii): *Si t_1 est un type alors $t = \{t_1\}$ est un type ensemble.*

(iv): *Si t_1 est un type alors $t = (t_1)$ est un type liste.*

(v): *Notons T l'ensemble des types.*

Les valeurs sont récursivement construites de la manière suivante.

Définition 2.2. *(Valeurs) Soit A un univers d'attributs et D un domaine de valeurs atomiques. Une valeur simple est prise dans l'un de types prédéfinis que sont les entiers (***integer***), les valeurs réelles (***real***), les valeurs logiques (***boolean***), les caractères (***char***) et les chaînes de caractères (***string***).*

(i): *Tout élément de D est une valeur (dite atomique).*

(ii): *Un identificateur d'objet est une valeur.*

(iii): *Si $v_1, \ldots, v_n$ sont des valeurs et $a_1, \ldots, a_n$ des attributs de A, alors $v = [a_1 : v_1, \ldots, a_n : v_n]$ est une valeur structurée de type n-uplet.*

(iv): *Si $v_1, \ldots, v_n$ sont des valeurs distinctes alors $v = \{v_1, \ldots, v_n\}$ est une valeur structurée de type ensemble.*

(v): *Si $v_1, \ldots, v_n$ sont des valeurs alors $v = (v_1, \ldots, v_n)$est une valeur structurée de type liste.*

(vi): *Notons V l'ensemble des valeurs.*

Les objets sont construits de la manière suivante:

Définition 2.3. *(Objets) Un objet est un couple $o =< i, v >$ où i est un élément de I (un identifiant) et v est une valeur.*

3. La rélation de sous-typage

Notons $n..m$ le sous-type du type **integer** correspondant à l'ensemble des entiers de n à m bornes comprises. On définit récursivement la relation de sous-typage de la façon suivante:

Définition 3.1. *(Sous-types)*

(i): *$n..m \preceq p..q$ si et seulement si $p \leq n$ et $q \leq m$.*

(ii): *Si $t_1, \ldots, t_m, u_1, \ldots, u_n$ sont des types et $a_1, \ldots, a_m$ sont des attributs alors $[a_1 : t_1, \ldots, a_m : t_m] \preceq [a_1 : u_1, \ldots, a_n : u_n]$ si et seulement si $t_i \preceq u_i$ pour tout i de 1 à n et $n \leq m$.*

(iii): *Si t_1, t_2 sont des types alors $\{t_1\} \preceq \{t_2\}$ si et seulement si $t_1 \preceq t_2$.*

(iv): *Si t_1, t_2 sont des types alors $(t_1) \preceq (t_2)$ si et seulement si $t_1 \preceq t_2$.*

Définition 3.2. *(Classes) Une classe est un couple $C =< O, T, M >$ où O est l'ensemble d'objets (instances) de la classe (**extension** de la classe), T est le type d'objets de la classe, et M, l'ensemble de méthodes appliquables sur l'objets de la classe.*

Le type T, associé à une classe C reflète la hiérarchie de composition des objets de cette classe, c'est-à-dire les liens que possèdent ceux-ci avec d'autres objets [2, 5, 6].

Le type T peut être représenté par un graphe orienté étiqueté, dont la définition suit:

Définition 3.3. *(Graphe de type GTC)*

(i): *$GTC = (N_T, E_T)$ où:*

(ii): *N_T est l'ensemble des sommets étiquetés. Chaque sommet représente un type et est étiqueté au moyen de $\beta : N_T \to T$*

(iii): *Si t est un type atomique alors $t \in N_T$ et $\beta(t) = t$.*

(iv): *Si $t_1, \ldots, t_n \in N_T$ et $t = [a_1 : t_1, \ldots, a_n : t_n]$ alors $t \in N_T$ et $\beta(t) = []$.*

(v): *Si $t_1 \in N_T$ et $t = \{t_1\}$ alors $t \in N_T$ et $\beta(t) = \{\}$.*

(vi): *Si $t_1 \in N_T$ et $t = (t_1)$ alors $t \in N_T$ et $\beta(t) = ()$.*

(vii): *E_T est l'ensemble des arcs orientés et étiquetés au moyen de $\gamma : E_t \to A$ où A est l'ensemble des noms d'attributs.*

(viii): *Si $t = [a_1 : t_1, \ldots, a_n : t_n]$ alors $(t, t_i) \in E_T$ et $\gamma(t, t_i) = a_i$ pour tout i de 1 à n.*

(ix): *Si $t = \{t_1\}$ alors $(t, t_1) \in E_T$ et $\gamma(t, t_1)$ n'est pas définie.*

(x): *Si $t = (t_1)$ alors $(t, t_1) \in E_T$ et $\gamma(t, t_1)$ n'est pas définie.*

La totalité des liens entre tous les types présents dans un système S est donnée par le **Graphe des Types GTS.**

Définition 3.4. *(Graphe des types GTS) Soit S, l'ensemble des classes du système. Le graphe des types GT est défini comme suit: $GTS = \cup_{T \in S} GTC = (\cup_T N_T, \cup_T E_T)$.*

Etant donné l'ensemble O d'objets de la classe C, les liens entre ces objets pourront être représentés par un graphe orienté. Nous appelons ce graphe: **Graphe de Composition d'Objets** GCO. La définition d'un tel graphe suit:

Définition 3.5. *(Graphe de composition d'objets GCO) Soit I un ensemble d'identificateurs d'objets et V l'ensemble de valeurs atomiques ou structurées. Le GCO est défini pour les objets/valeurs par:*

(i): *$G_I = (V_I, E_I)$ où:*

(ii): *V_I est l'ensemble des nœuds chaque nœud représente une valeur et est étiqueté par $\alpha : V_I \to I$ et $\beta : V_I \to V$.*

(iii): *Si v est associée à l'objet identifié par i alors $v \in V_I$ et $\alpha(v) = i$.*

(iv): *Si v est une valeur atomique alors $v \in V_I$ et $\beta(v) = v$.*

(v): *Si $v_1, \ldots, v_n \in V_I$ et $v = [a_1 : v_1, \ldots, a_n : v_n]$ alors $v \in V_I$ et $\beta(v) = []$.*

(vi): *Si $v_1, \ldots, v_n \in V_I$ et $v = \{v_1, \ldots, v_n\}$ alors $v \in V_I$ et $\beta(v) = \{\}$.*

(vii): *Si $v_1 v_n \in V_I$ et $v = (v_1, \ldots, v_n)$ alors $v \in V_I$ et $\beta(v) = ()$.*

(viii): *E_I est l'ensemble des arcs étiquetés au moyen de $\gamma : E_I \to A$, où A est l'ensemble des noms d'attributs.*

(ix): *Si* $v = [a_1 : v_1, \ldots, a_n : v_n]$ *alors* $(v, v_k) \in E_I$ *et* $\gamma(v, v_k) = a_k$ *, pour tout* k *de* 1 *à* n.

(x): *Si* $v = \{v_1, \ldots, v_n\}$ *alors* $(v, v_k) \in E_I$ *et* $\gamma(v, v_k)$ *n'est pas définie, pour tout* k *de* 1 *à* n.

(xi): *Si* $v = (v_1, \ldots, v_n)$ *alors* $(v, v_k) \in E_I$ *et* $\gamma(v, v_k)$ *n'est pas définie, pour tout* k *de* 1 *à* n.

Le *GCO* peut être vu comme une "instanciation" du graphe des types *GT*. La relation d'héritage entre classes peut être représentée par une relation de sous-typage entre types.

Définition 3.6. *(La relation d'héritage des classes) Soit* C *l'ensemble des toutes les classes et* $C_1 =< O_1, T_1, M_1 >$, $C_2 =< O_2, T_2, M_2 >$, *deux classes de* C. $C_2 \preceq C_1$ *si et seulement si,* $T_2 \preceq T_1$ *et* $M_1 \subseteq M_2$.

Si deux classes sont en relation d'héritage (une relation d'ordre partiel), on appelle C_1la super-classe de C_2 et C_2 la sous-classe de C_1.

Définition 3.7. *(Le graphe d'héritage des clases GHC)*

(i): $G = (X, U) = (X, \Gamma)$ *où:* X *est l'ensemble des sommets, ou classes, et* U *la relation d'heritage* ($\preceq$), *un ensemble de* $X \times X$. G *est muni d'une racine, notée* C_0.

(ii): Γ *est la relation multi-valuée des successeurs d'un sommet, équivalente à* U. *On utilisera aussi* Γ^{-1} *qui représente les prédécesseurs.*

On note $\lambda = (C_0, C_1, \ldots, C_n)$ *et on appelle* λ *un chemin de* G *si et seulement si* $\{(C_0, C_1), (C_1, C_2), ..., (C_{n-1}, C_n)\} \subseteq U$.

Définition 3.8. *(Sous-graphe de descendants) Soit* C_r *un sommet du graphe d'héritage.* $G_1 = (X_1, U_1)$ *où:* $X_1 = \{C_X \in X \mid \exists \lambda = (C_0, \ldots, C_X)$ *un chemin dans* $G\}$. C_1 *est la racine de sous-graphe* G.

4. Conclusion

Nous avons présenté ici les concepts d'un modèle à objets complexes avec identité. Dans les SGBDOO le concept d'objet est fondamental. Nous avons défini le concept de type complexe à partir de types atomiques et de constructeurs qui s'appliquent récursivement.

La relation de sous-typage entre types peut être considérée comme une possibilitée d'établir une hiérarchie d'héritage sur l'ensemble des classes.

On représent souvent les liens entre types et objets au moyen d'un graphe. Nous avons construit le graphe des types GT, le graphe de composition d'objets GCO, et le graphe d'héritage GHC.

Le graphe des types, le graphe de composition d'objets et le graphe d'héritage jouent un rôle prépondérant dans la mise en œuvre des stratégies de regroupement d'objets sur disque dans un système de bases de données orienté-objet [5, 6].

Bibliographie

[1] S. Abiteboul, P. Kanellakis, *Object Identity as Query Language Primitive*, In Proc. of the ACM SIGACT- SIGMOD Symp. on Principles of Database Systems, Juin, 1989.

[2] M. Adiba, C. Collet, *Objets et Bases de Données. Le SGBD O2*, Hermès, Paris, 1993.

[3] M. Atkinson, P. Buneman, *Types and Persistence in Database Programming Languages*, ACM Computing Surveys, June, 1987.

[4] A. Bancilhon, S. Khoshafian, *A calculus for Complex Objects*, ACM PDDS Conference 1986.

[5] V. Benzaken, *Regroupement d'objets sur disque dans un système de bases de données orienté-objet*, Thèse de doctorat, Paris, 1990.

[6] V. Benzaken, A. Doucet, *Bases de Données orientées objet. Origines et principes*, Armand Colin, 1993.

[7] E.F. Codd, *A Relational Model of Data for Large Shared Data Banks*, CACM Vol 13 No 6, June, 1970.

[8] M. Humbert, *Les Bases de Données*, Hermès, 1989.

[9] S. Khoshafian, G. Copeland, *Object Identity*, OOPSLA 86, Portland Oregon, Sept. 1986.

[10] C. Lecluse, P. Richard, F. Velez, *O2, an Object- Oriented Data Model*, ACM 3/1988.

[11] C. Lecluse, P. Richard, *Modeling Complex Structures in Object-Oriented Databases*, Proc. of the ACM PODS Conference, Philadelphie, 1989.

[12] GR. Moldovan, *Modelul Relational pentru Baze de Date, Metodologii si tehnici moderne de proiectare si scriere a programelor*, Bucuresti, 1981, pag. 213-226.

[13] P. Richard, *Des Objets Complexes aux Bases de Données Orientées - objets*, Thèse de doctorat, Paris, 1991.

"Babeş-Bolyai" University, Faculty of Mathematics and Computer Science, RO-3400 Cluj-Napoca, Romania

E-mail address: moldovan@cs.ubbcluj.ro

University of Craiova, Faculty of Mathematics – Computer Science, RO-1100 Craiova, Romania