Volně přístupné objektově orientované databáze

Jana Kuklová, Luboš Popelínský
Katedra matematické informatiky, Masarykova universita
Brno

Objektově orientovaný přístup se v současnosti stále více prosazuje i v oblasti databází. Přestože mezi odborníky neexistuje shoda v definici objektově orientovaného datového modelu, **objektově orientované databáze** [1], [5], [8] se stávají vážnou konkurencí relačním databázím.

První prototypy objektově orientovaných databází se objevují v polovině 80. let. Jedná se například o systém ORION (1987), implementovaný v LISPu [9], systém IRIS (1987) vyvinutý v laboratořích Hewlett-Packard a další. V této době se objevují i první komerční objektově orientované databáze jako je G-Base (1987) používající LISPovskou syntaxi, GemStone (1987) vycházející ze Smalltalku, Ontos (1988) založený na knihovnách implementovaných v jazyce C++ nebo O_2 (1990). Vedle čistě objektově orientovaných databází vznikají i **rozšířené relační databáze**. Rozšíření převážně umožňuje vytváření vlastních uživatelských funkcí, abstraktních datových typů a nabízí podporu pro multimediální datové typy. Další relační databáze se prohlašují za objektově orientované, ale ve skutečnosti nabízejí pouze objektově orientované uživatelské prostředí (MAGIC od Magic Software Enterprises pro operační systémy MS-DOS a UNIX, 4thDIMENSION od Analyses Conseils Informations pro počítače Macintosh). Základní strategie vývoje objektově orientovanách databází jsou tyto:

- Vytvoření zcela nového čistě objektově orientovaného databázového systému
- Rozšíření existujícího standardu SQL o vlastní objektově orientovaný dialekt
- Rozšíření existujícího objektově orientovaného jazyka (např. Smalltalk) o databázové vlastnosti
- Vytvoření knihoven pro objektově orientovaný databázový systém a jejich využití v objektově orientovaném programovacím jazyce (např. C++)
- Vnoření objektově orientovaného databázového jazyka do hostitelského programovacího jazyka (např. C)

- Speciální aplikace vytvořené nad objektově orientovanou správou databáze

Nelze určit přesnou dělící čáru mezi rozšířenými relačními a objektově orientovanými databázemi. Například v [9] se za objektově orientovanou považuje pouze ta databáze, která umožňuje polymorfismus.

Kromě objektově orientovaných databází se v posledních patnácti letech objevují přinejmenším dva další směry výzkumu - **aktivní databáze** a **deduktivní databáze**. Typickým příkladem chování aktivní databáze je reakce na modifikaci některého údaje, například atributu, v databázi. Popis takové akce je přímo součástí databáze. Něco podobného se běžně používá v operačních systémech(démony). Za aktivní můžeme považovat i POSTGRES. Rozšíření ODE o aktivní rysy je popsáno v [3].

Soupeření mezi objektově orientovanými výzkumníky a deduktivními databázisty z poloviny 80. let stále častěji ustupuje snaze o integraci obou proudů. Ta vyústila ve vznik nové řady konferencí DOOD, které se konají od roku 1989 každé dva roky. Ta letošní proběhne od 6. do 8. prosince ve Phoenixu v Arizoně.

Vytvoření databázového systému, který bude obsahovat všechny tři rysy, tedy aktivní deduktivní objektově orientovné databáze, je cílem jednoho z ESPRIT projektů řešených na universitě v Bonnu.

V naší přednášce chceme databázovou veřejnost seznámit se třemi objektově orientovanými databázemi, které lze jako public domain získat v počítačových sítích. Všechny uvedené databáze pracují pod systémem UNIX na počítačích SUN. Budeme zkoumat jejich objektovou orientaci (abstraktní datové typy, dědičnost, identita objektů) a databázovou podporu (persistence, dotazovací jazyk, verse dat, transakce, integritní omezení, paralelní zpracování, bezpečnost, zotavení).

1 POSTGRES 4.0

POSTGRES (vedoucí projektu Michael Stonebraker, Univ. of California, Berkeley) [10] je rozšířením relační databáze INGRES. POSTGRES pro komunikaci s uživatelem používá jazyk POSTQUEL, který mimo jiné nabízí definici vlastních datových typů a funkcí, dědičnost, složené objekty a verze objektů.

Příkazy jazyka POSTQUEL uživatel interaktivně zadává terminálovému procesu `monitor`. Komunikaci mezi tímto procesem a příslušnou databází zabezpečuje proces `postmaster` spuštěný na pozadí. Pro přístup do databáze přímo z jazyka C je určena sada funkcí obsažená v knihovně LIBPQ.

Příkaz `create` jazyka POSTQUEL vytváří novou třídu a ukládá ji do databáze. Nová třída může být odvozena z několika základních tříd, definice třídy zahrnuje pouze definici jejích atributů. Jméno třídy může být automaticky použito jako typ při definici dalších tříd.

```
create Employee (name = text, dep = char16) \g
create Department (name = char16, head = text) \g
```

Uživatel může vytvářet nové datové typy, operace a funkce a ukládat je do databáze, jejich definice však nejsou zapouzdřeny v definici příslušné třídy. Uživatelskou funkci definuje příkaz `define function`, funkce vrací hodnotu libovolného typu nebo množinu objektů daného typu. Tělo funkce je psáno v jazyce POSTQUEL nebo v jazyce C.

```
define function chief
 (language = "postquel", returntype = Employee)
  arg is (Employee)
  as "retrieve (E.all)
          from E in Employee
          where E.name = Department.head
                and Department.name = $1.dep" \g
```

Příkaz `append` způsobí rozšíření databáze o nové objekty, příkaz `retrieve` umožňuje vyhledání objektů uložených v databázi podle zadaných kritérií. V uvedeném příkladě jsou vybráni všichni zaměstnanci, jejich šéfem je `Tom`.

```
append Employee (name = "Tom", dep = "KMI") \g
append Employee (name = "Mojmir", dep = "KMI") \g
append Department (name = "KMI", head = "Tom") \g
retrieve (Employee.name) where Employee.chief.name = "Tom" \g
```

POSTGRES automaticky uchovává v databázi historii všech objektů. Následující dotaz vybere z databáze všechny vedoucí katedry `KMI` od roku 1989.

```
retrieve (D.head)
    D in Department ["Jan 1 00:00:00 1989", "now"]
    where D.name = "KMI" \g
```

Příkaz `define rule` slouží k definici pravidel aktivujících se při vytvoření, zrušení, změně nebo výběru příslušného objektu. Tato pravidla umožňují snadnější uchování databáze v konsistentním stavu a chrání před neoprávněným přístupem.

```
define rule no_access is
   on retrieve to Department.head
   where current.name = "KMI"
            and pg_username() = "Jan"
   do instead nothing \g
```

Předcházející příkaz zakazuje uživateli jménem `Jan` vyhledat vedoucího katedry `KMI`. Klíčové slovo `current` znamená objekt, při jehož výběru bylo pravidlo `no_access` aktivováno, `do instead` uvádí akci, která se vykoná místo příkazu zadaného uživatelem.

Transakce jsou uvozeny slovem `begin`, přerušení transakce provede příkaz `abort`, provedení transakce je potvrzeno příkazem `end`.

POSTGRES nabízí tvorbu bezpečnostních kopií databází, zotavení z chyb a multiuživatelský přístup. Speciální podporu poskytuje pro práci s objekty, jejichž velikost přesahuje délku 8 kB.

POSTGRES je možno získat anonymním ftp na adrese `toe.cs.berkeley.edu` včetně zdrojových textů v jazyce C.

2 ODE 2.0

ODE (AT&T Bell Laboratories) [2] vychází z objektů definovaných v jazyce C++ a přidává jim databázovou podporu. Je tvořeno překladačem jazyka O++ a knihovnami pro práci s objekty. Překladač jazyka O++ přeloží zdrojový text do jazyka C++ a výsledný kód předá překladači C++. Předpokládá se, že uživatel má instalován překladač jazyka C++ verse 2.1 nebo vyšší.

Využití databázové podpory programem v O++ je zpřístupněno zadáním příkazu

```
#include <ode.h>
```

V souboru `ode.h` je obsažena definice třídy `database`, která definuje funkce pro práci s databázemi. Program napsaný v O++ může pracovat s několika databázemi, v každém okamžiku však smí být otevřena pouze jedna z nich.

Veškerý přístup k databázi musí být uzavřen v bloku transakce. Transakce mohou být trojí – pro zápis, pro čtení a hypotetická. Pokud je v programu použita transakce pro čtení, překladač při spuštění požaduje nastavení speciální volby. Příkaz `tabort` přeruší vykonávání transakce a provádění programu pokračuje prvním příkazem za touto transakcí.

V následujícím příkladě je popsáno otevření databáze `test`, případné hlášení chyby a uzavření databáze.

```
database *db;
main{
if ((db = database::open("test")) == NULL {
  cout << "error" << endl;
```

```
  exit(1);
};
trans{
// ... access to  database ...
};
db -> close;
};
```

Pro označení typů, jejichž hodnoty budou uloženy v persistentní paměti, jazyk O++ používá klíčové slovo `persistent`. Způsob definice tříd objektů je převzat z jazyka C++. Definice třídy zapouzdřuje deklaraci atributů objektu i definice metod spojených s objektem. Třída může být vytvořena odvozením z jedné nebo více základních tříd.

```
persistent class Employee;
persistent class Department;

class Department;
public:
 char name[20];
 persistent Employee *head;
 Department (char *xname, persistent Department *xdept){
  strcpy(name, xname);
  head = xhead;
 };
 };

class Employee {
public:
 char name[20];
 persistent Department *dep;
 Employee(char *xname, persistent Department *xdep){
  strcpy(name, xname);
  dep = xdep;
 };
 persistent Employee *chief(){
   return(dep->head);
 };
 };
```

Funkce `chief()` vrací ukazatel na šéfa konkrétního zaměstnance.

Vytvoření a zrušení persistentního objektu provádějí funkce `pnew` a `pdelete` podobným způsobem jako `new` a `delete` v C++. Funkce `pnew` vrací ukazatel na nově vytvořený persistentní objekt. S objekty a s ukazateli na ně se pracuje obdobně jako v C++.

```
persistent Department *de;
persistent Employee *em;

de = pnew Department("KMI", NULL);
em = pnew Employee("Tom", de);
de -> head = em;
em = pnew Employee("Mojmir", de);
pdelete em;
```

Jednotlivé persistentní objekty mohou být pojmenovány. Toto jednoznačné jméno je uloženo v databázi a umožňuje rychlejší přístup k pojmenovanému objektu.

```
db -> set_name("boss", em);
de -> head = db -> get_obj("boss");
```

ODE nemá dotazovací jazyk typu SQL. Pro práci s objekty uloženými v databázi slouží `for` cyklus, ve kterém jsou postupně zpřístupněny všechny objekty dané třídy vyhovující zadaným podmínkám. Cykly `for` lze vnořovat do sebe.

```
for (em in Employee)
suchthat (!strcmp(em -> chief() -> name, "Tom")){
 cout << em -> name << endl;
 };
```

Výsledkem předcházejícího příkladu je seznam jmen všech zaměstnanců, jejichž vedoucím je `Tom`.

Jazyk O++ nabízí vytváření versí jednotlivých objektů, přičemž objekty stejného typu mohou mít různý počet versí. Jednotlivé verse téhož objektu jsou setříděny podle doby vzniku a podle toho z které verse byly odvozeny. V těchto uspořádáních je lze i postupně zpřístupňovat.

```
persistent Employee *x;
x = newvers(em);
```

Po násilném ukončení práce s databází je databáze automaticky uvedena do posledního konsistentního stavu při novém startu Ode. Další databázová podpora ve verzi 2.0. chybí. Nejsou implementovány `constraints` a `triggers`, kopie databázových souborů je nutno provést systémovými prostředky, není podporován víceuživatelský přístup. Systém se však stále vyvíjí, je využíván AT&T pro podporu multimédií v projektu interaktivní TV.

ODE v binárním tvaru je přístupné anonymním ftp na adrese `research.att.com`, zároveň je třeba zaslat e-mailem žádost Narain Gehanimu na adresu `nhg@allegra.att.com` a podepsat licenční smlouvu.

3 ConceptBase 3.2

ConceptBase [6], vyvinutá skupinou Manfreda Jeusfelda, je deduktivní objektově orientovaná databáze, jejíž datový model spojuje objektově orientované modelování s temporální a deduktivní databázi. Běžné vlastnosti objektově orientovaných databází jako je dědičnost a hierarchie objektů obohacuje o deduktivní pravidla a integritní omezení. Předchozí verse tohoto databázového systému byly úspěšně aplikovány v takových oblastech jako je databázová podpora při tvorbě informačních systémů [7] nebo vytváření hypertextových souborů paralelně více autory [4]. System je implementován převážně v BIM Prologu, částečně v jazyku C. K jeho běhu jsou nezbytná X11 Windows verse 5.

Architektura klient-server využívá sítě Internet, což umožňuje klientům jedoucím na různých počítačích spolupracovat s týmž serverem bez ohledu na vzdálenost. Přátelský uživatelský interface nad X11 obsahuje též GraphBrowser, program pro grafické zobrazování databáze.

Vlastním jazykem databáze je jazyk pro reprezentaci znalostí Telos umožňující vyjádřit znalosti jak ve tvaru objektů tak deduktivních pravidel. Pro ilustraci uvádíme krátký příklad. Definujme třídy `Employee` a `Manager` následujícím způsobem

```
Class Employee with

   attribute
      name : String;
      salary : Integer;
      dep : Department;
      boss : Manager
end

Class Manager
      isA Employee with
end
```

Prvek třídy `Mojmir` má atributy s následujícími hodnotami

```
Individual Mojmir in Employee with
   dep
      MojmirDepartment : kmi
   salary
      MojmirSalary : 12345
end
```

Následující pravidlo `bossrule` definuje hodnotu atributu `boss` v třídě `Employee`.

```
Class Employee with rule

  bossrule :

    $ forall e/Employee  m/Manager
        ((exists d/Department
            (e dep d) and (d head m))
             ==>
            (e boss m)) $

  constraint SalaryBound :

    $ forall e/Employee b/Manager x/Integer y/Integer
        (( (e boss b) and (e salary x) and (b salary y))
             ==>
            (y >= x)) $

end
```

tzn. pro každého zaměstnance `e` se odvodí hodnota atributu `boss`, existuje-li oddělení `d`, v němž zaměstanec `e` pracuje a manažer `m` je vedoucím tohoto oddělení. Ovšem pouze za podmínky, že jeho plat je vyšší nebo roven platu všech zaměstnanců, což vyjadřuje integritní omezení `SalaryBound`.

ConceptBase není vhodná pro manipulaci s velkými databázemi. Již s objektovou bází o velikosti 1 MB je práce velmi neefektivní. Kromě integritních omezení a deduktivních pravidel není možno definovat metody. Programové rozhraní pro přístup z jiných programovacích jazyků sice existuje, ale je velmi omezené. Není podporován víceuživatelský přístup.

ConceptBase umožňuje libovolně hlubokou hierarchii tříd dovolující popsat objekty, třídy, nadtřídy, nad-nadtřídy atd. stejným způsobem. Modelování a manipulace s takovými objekty je velmi usnadněna grafickým browserem. K nezanedbatelným přednostem patří též přátelské uživatelské rozhraní.

Vývoj ConceptBase byl zahájen v roce 1987 na universitě v Passau a první verse se objevila na začátku roku 1988. Verse 2.x již obsahovala integritní omezení a měla architekturu klient-server. Později byla doplněna o intervalový temporální kalkul. Verse 3.x se liší novým uživatelským rozhraním (X11) a definicí dotazů jako tříd. Od roku 1992 probíhá výzkum a vývoj na technické universitě RWTH v Aachen.

ConceptBase získáte na adrese `ftp.informatik.rwth-aachen.de` pomocí anonymního ftp v adresáři `pub/CB/CB_3.2`. Po získání klíče ji můžete používat čtyři týdny. Potom máte dvě možnosti: znovu zkopírovat pomocí ftp nebo požádat o dlouhodobou licenci.

4 Závěr

Objektově orientované databáze jsou intenzívně zkoumány a vyvíjeny ne déle než deset let. Stále častěji se však úspěšně uplatňují i na softwarovém trhu, kde soupeří s tradičními relačními databázovými systémy obohacenými o objektově orientované rysy. Dobývají úspěšně ty oblasti aplikací, na něž doposud databázové technologie nestačily. Což neznamená, že překonávají relační databáze ve všech směrech. Jejich přednosti se však projevují u aplikací, které jsou složité co do struktury a/nebo chování jako jsou CAD, CASE systémy nebo multimedia.

Jsou-li pro vás profesionální produkty nedosažitelné (neúplná verse databáze O_2 stojí asi \$2000), vyzkoušejte některou z nahoře uvedených. Možná, že naše teoretické i praktické znalosti pomohou i vám najít cestu k objektově orientovaným databázím.

Literatura

[1] Atkinson M. et al.: The Object-Oriented Database System Manifesto. In: *1st International Conference on Deductive & Object-Oriented Databases, Kyoto, 1989.*

[2] Biliris A., Gehani N. H., Jagadish H. V., Roome W. D.: *ODE Object Database and Environment.* AT&T Bell Laboratories, Murray Hill, New Jersey 07974, 1992.

[3] Gehani N., Jagadish H. V.: *Ode as an Active Database: Constraints and Triggers.* In: VLDB'91.

[4] Hahn U., Jarke M., Eherer S., Kreplin K.: CoAUTHOR: a hypermedia group authoring environment. In: *Benford J.M., Bowers S.D.(eds.): Studies in Computer-Supported Cooperative Work* North-Holland 1990

[5] Hughes: *Object-Oriented Databases.* Prentice Hall 1991

[6] Jarke M.(ed.): *ConceptBase V3.2 User Manual.* RWTH Aachen 1993.

[7] Jarke M., Mylopoulos J., Schmidt J. W., Vassiliou Y.: Information systems development as knowledge engineering: a review of the DAIDA project. *Programirovanie 17(1) 1990 Academy of Science, Russia*; also report MIP-9010, Universiteat Passau, Germany.

[8] Khoshafian S.: *Object Orientation: concepts, languagues, user interfaces.* John Willey & Sons, Inc., 1990.

[9] Kim W.: *Introduction to Object-Oriented Databases.* The MIT Press Cambridge, Massachusets, 1990.

[10] Rhein J., Kemnitz G. and The POSTGRES Group: *The POSTGRES User Manual.* University of California, Berkeley, 1990.