一个新的面向对象数据库系统的动态数据模型*

黄刘生, 陈华平, 郑启龙, 陈国良

(中国科学技术大学 计算机科学与技术系,安徽 合肥 230027);
(国家高性能计算中心,安徽 合肥 230027)
E-mail: lshuang@ustc.edu.cn
http://cs.ustc.edu.cn

摘要: 现有的OODBMS(object-oriented database management systems)的数据模型基本上可以分为两类:传统的强类型和非传统的弱类型.前者的特征是所有具有同样结构(模式)和行为(方法)的对象组成一个类,对象的含义由对象所属类的模式解释,故类的定义必须先于其对象的定义;后者的特征是"无模式",即对象是"自我描述"的,因此无须为对象预先定义模式.指出了这两类模型的主要优缺点,并证实:为了充分适应有效而灵活的动态对象操作,有必要在传统的强类型模型和非传统的弱类型模型之间寻求某种折衷.为此,提出了一个基于概念聚类机制(conceptual clustering mechanism,简称CCM)的数据模型.这种新的动态模型已成功地应用于一个面向对象的VDBMS(video database management system).

关键词: 概念聚类机制(CCM);强类型数据模型;弱类型数据模型;簇;角色;扮演者

中图法分类号: TP311 **文献标识码**: A

一个OODBMS(object-oriented database management systems)可被定义为直接支持面向对象的数据模型的DBMS[1].多数人认为一个面向对象的数据模型必须支持:(1) 对象标识(Oid),即现实世界的实体被模型化为对象后均有一个惟一的标识;(2) 复杂对象,即一个对象可以是其他对象的聚合(aggregation);(3) 继承性,即一个类的性质(属性和方法)可被子类继承[1,2].由此可见,OODBMS是面向对象的程序设计技术与DBMS相结合的产物.受面向对象程序设计语言的影响,共享同样属性结构(模式)和行为(方法)的对象集合构成一个类[2],在定义对象之前必须先定义其类型(即类).因此,我们将这种传统的数据模型称为强类型模型,它较为适合表达良结构及同质对象的集合.

但Ullman认为,支持Oid的数据模型即是面向对象的,否则是面向值的[2].Gattell则认为一个面向对象的数据模型只需支持Oid和对象的嵌套(类似于复杂对象)[3].因此,Papakonstantinou等人提出了一个新模型OEM(object exchange model)[4],它仅支持Oid和对象嵌套,不支持类和继承性.由于无须为对象预先定义类型,因此我们把这种模型称为弱类型模型.它适合于表达病态结构和异质对象的集合,尤其是当对象结构并不完全已知时,便于动态地增加和删除对象的属性.预先定义类的强类型数据模型对此没有提供足够的支持[4].本文将分析这两类模型的特征及存在的问题,然后提出一个新的模型.

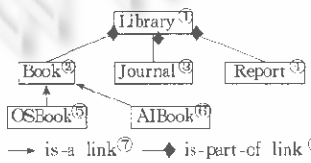
* 收稿日期: 1999-11-24; 修改日期: 2000-03-17
基金项目: 国家863高科技发展计划资助项目(863-306-ZD06-2)

作者简介: 黄刘生(1957-),男,安徽宿松人,教授,主要研究领域为OODB,KDD,分布式算法;陈华平(1965-),男,江苏江阴人,博士,教授,主要研究领域为数据库,并行分布计算;郑启龙(1969-),男,四川成都人,讲师,主要研究领域为并行软件工具与环境;陈国良(1938-),男,安徽颍上人,教授,博士生导师,主要研究领域为并行分布计算,并行机体系结构,并行算法.

1 强类型数据模型

强类型数据模型中有两个层次:(1) 由超类和子类构成的类层次(class hierarchy),它捕获了子类和超类间的"is-a"语义,说明二者间是特殊与一般的关系;(2) 由类和其成分类构成的组合层次(composition hierarchy),它捕获了二者间的"is-part-of"语义,说明二者间是整体与部分的关系[1,5]. 强类型模型中有实例化和继承两种复用机制. 前者可用类定义来产生具有同样结构和行为的对象,后者使子类可继承其超类的性质.

在强类型模型中,静态定义的类不能灵活而有效地适应动态的应用环境. 例如,早期建立的图书馆类只需包括书、杂志和技术报告等成分类(如图 1 所示),但现在它应该包括 CD 盘、微缩胶片和 PS 文件等成分类,而将来它所包括的成分类现在无法预料. 因此要修改 Library 类结构才能适应新的需求,这对一个已交付使用的数据库系统来说,显然是不可取的. 当面向对象的模型应用于某些系统时,常需要从已存在的对象(可能是异质对象)中动态地产生某些临时性的组合对象,并要求能为其中的成分对象动态地增加或删除某些附加性质的描述. 显然,静态的类定义不能描述这种临时对象. 虽然某些支持视图(view)的对象模型允许从现有的对象中派生出视图,但视图仍然是一种特殊的类,它也不能适应这种应用要求.



(①图书馆,②书,③杂志,④报告,⑤操作系统类书籍,⑥人工智能类书籍,⑦is-a链,⑧is-part-of链.

Fig. 1 Cass-and composition-hierarchy
图1 类层次和组合层次

在传统的类层次上,类的属性结构被该类及其子类的全体对象共享,但属性值却不能被这些对象共享. 故有必要扩充类层次,使之支持值共享属性. 例如,设 Book 类有一属性 TimeLimit 表示借阅期限,若将其定义为结构共享属性,则每本书都持有同样的 TimeLimit 值,这将浪费存储空间和修改该值的时间,而且不利于保持数据的一致性和完整性. 因此,应将其定义为值共享属性,使所有书对象共享同一个 TimeLimit 值为宜.

有的数据模型为了区别对象性质和类性质,将属性、方法区分为对象和类的属性、方法[1]. 类属性是一个聚合(aggregation)属性,可用来表示类对象的数目、某个对象属性的平均值等,故它类似于 C++的类的静态数据成员. 虽然类属性和值共享属性对类而言均只有一个值,但二者含义不同. 例如,设 Book 类有一个 Num 属性表示各种书的总数,显然它是一个类属性,不属于类的任何对象,也不能被子类继承;而值共享属性 TimeLimit 是对象的属性,它刻画了书对象的性质,可被子类继承.

在类的组合层次上,类的实例是复杂或组合对象(complex/composite object). Won Kim 认为组合层次通常没有继承性[5],然而我们认为选择部分属性继承是有必要的. 例如,在一个 Video 数据库中,一个节目类由若干个片段类组成,而一个片段类由若干个帧类组成,这 3 种类构成了一个组合层次. 假设节目类具有导演、摄影、片长等属性,显然,导演、摄影等属性的值能够被片段类和帧类继承,而片长等属性值不能被片段类和帧类继承. 现有的强类型模型不支持这种组合层次上的选择继承性.

2 弱类型数据模型

弱类型数据模型是无模式的,它无须预先定义对象类型. 每个对象都包含自身的模式,故它是一种"自我描述"的模式,现以 OEM 为例说明其特点. 在 OEM 中,每个对象是一个四元组(Oid,

Label,Type,Value〉.这里,Oid 和 Label 相当于对象的物理标识和逻辑标识;Value 和 Type 分别表示对象的值及其类型,对象的值类型可以是原子或集合,集合值通过 Oid 引用其他对象.例如,图 1在 OEM 中可表示如下:

〈Library,set,{cmp_1,cmp_2,cmp_3}〉// Library 是对象标号,它是集合对象,cmp_1,cmp_2,cmp_3 是成分对象的 Oid

cmp_1:〈Book,set,{cmp_{11},cmp_{12},cmp_{13},cmp_{14},...}〉 //Book 是集合对象

cmp_{11}:〈Num,int,700 000〉 //书数目的总数,原子对象

cmp_{12}:〈TimeLimit,int,90〉 //借阅时间为 90 天,原子对象

cmp_{13}:〈OsBook,set,{cmp_{131},...,cmp_{13m}}〉 //cmp_{131},...,cmp_{13m}对应 m 本 OS 书

cmp_{131}:〈BookRecord,set,{cmp_{1311},cmp_{1312}}〉 //cmp_{132},...,cmp_{13m}的定义与 cmp_{131}相似

cmp_{1311}:〈Title,string,"Operating System"〉 //原子对象

cmp_{1312}:〈Author,string,"Ullman"〉 //原子对象

cmp_{14}:〈AIBook,set,{cmp_{141},...,cmp_{14n}}〉//cmp_{141},...,cmp_{14n}对应 n 本 AI 书,其定义与 cmp_{131}相似

cmp_2:〈Journal,set,{cmp_{21},cmp_{22},...,cmp_{2i}}〉 //Journal 是集合对象

cmp_3:〈Report,set,{cmp_{31},cmp_{32},...,cmp_{3j}}〉 //Report 是集合对象

虽然这种模型简单,但它能灵活、有效地适应动态、异质的环境,也易于动态地从现有的对象中产生临时性的组合对象.例如,当要往 Library 对象中加入 CD 盘对象时,只要在 Library 的值集合中加入 cmp_4,并建立相应的对象 cmp_4:〈CD,set,{cmp_{41},...,cmp_{4k}}〉即可.若在 OSBook 对象中,某书(如 cmp_{13i})的作者是两个人,则在建立 Author 对象时只需指出它的值是集合类型,并建立相应的对象即可:

cmp_{1312}:〈Author,set,{cmp_{13121},cmp_{13122}}〉; cmp_{13121}:〈Author1,string,"Aho"〉; cmp_{13122}:〈Author2,string,"Ullman"〉.

由此可见,即使是同一类对象 Author,也允许是异质的.若要为 OS 课程建立一个临时指定的参考书目,则可建立临时对象:〈OSReferences,set,{cmp_{121},cmp_{13i}}〉,它表示参考书来自于两个 OSBook 的对象.

该模型的另一特点是标号代替了模式,这使得解释和操作对象更为直接,语法分析无须参考额外的模式,查询方式更为直观.例如,要查询 Ullman 的 OS 课程参考书,则可用类似于 Unix 的路径表达式表示为

select OSReferences. BookRecord. Title **Where** OSReferences. BookRecord. Author = "Ullman".

虽然 OEM 可用嵌套集合中的对象和子对象层次来模拟强类型模型中的类层次和组合层次,但它仍然不能区分结构共享属性、值共享属性和类属性,也没有支持组合层次上的选择继承性.此外,由于缺乏类及其性质继承机制,故它不支持强类型模型中的两种复用机制.因此,对于具有同样结构的对象,需要重复说明其标号和类型,这将增加存储空间以及修改标号和类型的时间开销.

3 基于概念聚类机制(CCM)的数据模型

3.1 簇和角色

CCM(conceptual clustering mechanism)的一个显著特征是允许同质或异质对象被动态地聚集到簇(cluster)中,簇中的对象可以扮演不同的角色(role)[6,7].一个簇可用四元组 $C_i=(N;A;M;$

X)来表示,其中 N,A 和 M 分别为 C_i 的名字、属性集和方法集;X 是"角色-扮演者(role-player)"的集合,$X=\{\langle R_i:P_i\rangle \mid 1\leqslant i\leqslant n\}$,其中 R_i 是角色,由名字、属性集、方法集和超角色(可为空)构成;P_i 是一个同质或异质对象的集合:$\{P_{i1},P_{i2},\ldots,P_{ik}\}$,其中的对象既是角色 R_i 的扮演者,又是簇 C_i 的成员.

簇是一个动态的集合对象,其属性、方法、角色(包括属性和方法)和扮演者均是动态地加入和删除的.从集合的观点看,簇是由成员对象子集构成的集合 $S=\bigcup P_i(1\leqslant i\leqslant n)$,每个子集 P_i 扮演角色 R_i.这里,角色就像线索一样将语义相关的对象组合在一起,形成了集合 S 的一个划分.但由于同一对象可能同时扮演不同的角色,故 $P_i\bigcap P_j$ 不一定是空集.簇的属性和方法刻画的是集合 S 的性质,它相当于强类型模型中的类属性和类方法,或者相当于组合层次上的属性和方法.角色 R_i 的属性和方法刻画的是子集 $P_i(1\leqslant i\leqslant n)$ 中成员对象的性质,它相当于实例对象的属性和方法.

簇中的成员本身又可能是另一个集合对象,该对象是簇的子簇.超簇和子簇形成的层次主要用来表达强类型模型中的组合层次[6,7].但从集合的观点看,类层次上除了"is-a"语义之外,也往往蕴含了"is-part-of"语义.例如,OSBook 是 Book 的子类,但它又是 Book 的组成成分,因此,簇层次同样可以表达类层次.一般地,子簇中的所有角色是该子簇在其超簇中扮演的角色(称为超角色)的子角色,子角色和超角色之间也形成了一个层次.在簇层次上,属性和方法均采用选择继承,因为簇是一个组合对象,所以这种继承是对象间的属性值和方法的继承;在角色层次上,子角色自动继承所有超角色的属性和方法,而其中的属性又可区分为结构共享属性和值共享属性,相应地,子角色可分别继承超角色的属性结构和属性值.

在最初的 CCM 模型中,簇仅可动态地创建和删除子簇和角色,而不能创建新的非簇对象(称为普通对象),这些普通对象仍作为类的实例存储在数据库中,簇只能动态地引用它们.从派生的观点看,簇是从现有对象中派生出来的组合对象,它类似于视图,但可通过增加簇和角色中的属性及方法为这些已存在的对象赋予新的语义,故它比视图更为灵活、有效[6,7].我们引入了主动簇(active cluster)的概念,它无须依赖类模型即可动态地创建和删除普通的及子簇的成员对象,而将只能动态地加入或删去已存在对象的引用的簇称为被动簇(passive cluster).图 1 中的类层次和组合层次可用簇层次和角色层次描述如下:

```
Cluster Library{
    // Library 的属性集
      Name:"USTC Library":I;
      Num:700 000:NI; //总册数
    // Library 的方法集
      getNum():NI;
      getName():I;
    // Library 的 Role-Players Set
      Publication:{Book,Journal};//出版物
      Exchange:{Report};//交换资料
}// End of Cluster Library
Cluster Book:Library{ //Library 的子簇
    //属性集
      Num:500 000:NI; //总册数
    //方法集
      getNum():NI;
    //Role-Players Set
      Book:{OSBook,AIBook,...};
```

```
Role Publication{
    //结构共享属性
      string PubHouse;
      string Date;
    //方法集
      getPubHouse();
      getData();
}// End of Role Publication
Role Book:Publication{ //Publication 的子角色
    //结构共享属性
      string Title;
      string Author[ ];
    ValueSharing;//以下是值共享属性
      int TimeLimit:90;
    //方法集
      getTimeLimit();
}//End of Role Book
Role OSBook:Book{//Book 的子角色
```

```
}// End of Cluster Book
Cluster OSBook:Book{//Book 的子簇
    //Cluster OSBook 的属性集
      Num:2 000:NI;
    //Cluster OSBook 的方法集
      getNum():NI;
    //Cluster OSBook 的 Role-Players Set
      OSBook:⟨b1,b2,...,b2000⟩
}//End of Cluster OSBook
```

```
    ValueSharing://值共享属性
      string Description:"Operating System";
    //Method Set
      getDescription();
}// End of Role OSBook
```

簇 Library 中的属性和方法均省略了类型和返回类型的说明.由"I"和"NI"来决定它们是否被子簇继承.Name 和 getName()被指定为"I",表示该属性的值和方法可被该簇的3个子簇 Book,Journal 和 Report 继承(这里只给出了 Book 的说明);表示书、杂志和技术报告的总册数的聚合属性 Num 的值不能被子簇继承,故 Num 和 getNum()被指定为"NI".该簇的3个成员对象均是子簇,它们扮演了两个不同的角色 Publication 和 Exchange,分别表示出版物和馆际交流资料.角色 Publication 的属性和方法分别被它的两个扮演者 Book 和 Journal 共享,故该角色可看作为其扮演者动态定义的类说明,反之,扮演者可看作角色的实例对象.

Library 的子簇 Book 又有两个子簇 OSBook 和 AIBook.簇 Book 的属性 Num 描述了 OSBook,AIBook 等的总册数,该簇只有一个与其同名的角色 Book,其中的 Title 和 Author 是结构共享属性,它们是其扮演者的模式;而定义在 ValueSharing 区中的 TimeLimit 是值共享属性,其值由所有扮演者共享.因为簇 Book 是角色 Publication 的扮演者,故簇 Book 中所有角色均应是 Publication 的子角色.而簇 Book 的两个子簇中的角色均应为角色 Book 的子角色.由簇 Library 及其子簇构成的簇层次是一棵类似于如图1所示的树;角色 Publication 及其子角色形成的角色层次是一棵如图2所示的树;类似地,角色 Exchange 及其子角色也构成了另一棵树.

簇 OSBook 的成员 $b_1,b_2,\ldots,b_{2000}$ 不再是子簇对象,而是普通对象,因为它们扮演的角色 OSBook 继承了其超角色的性质,故这些对象的属性和方法均来自于 Publication,Book 和 OSBook 这3个角色.此外,还可为普通对象动态地加入或删去附加属性,这些属性是一个由属性名、类型和值构成的三元组集合.例如,对象 $b_i(1\leqslant i\leqslant 2000)$ 的附加属性集合为 $\{(AttrName_k,AttrType_k,AttrValue_k)\mid 0\leqslant k\leqslant m_i\}$,这里 $m_i\geqslant 0$,不同的对象 b_i 和 b_j 其附加属性的名字、类型、值和二元组的数目均可以不同.

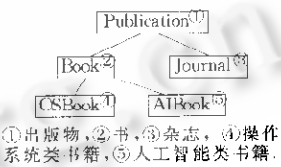


Fig. 2 Role tree
图2 Role树示例

以上的簇均为主动簇,簇中的成员对象均可直接由簇动态地创建或删除.下面我们给出一个被动簇的示例,其中的成员对象不能由簇创建或删除,而只能是动态地引用其他主动簇中已存在的成员对象.

```
Cluster OSReference{//被动簇
  //Attr-Set
      Course:"Operating System";
      Instructor:"Prof. Chen";
      ReverseTime:"18 Weeks";
  //Role-Players Set
      Major:⟨b100 from OSBook,j2,j200 from
            Journal⟩;
      Minor:⟨...⟩
```

```
Role Major {
  ValueSharing:
    string TimeLimit:"2 Hours"; //借阅时间
    ...
}//End of Role Major
Role Minor {
  ValueSharing:
    string TimeLimit:"4 Hours";
    ...
```

//Method Set
}//End of OSReference

}//End of Role Minor

这里,被动簇 OSReferences 是为 OS 课程指定的参考文献的集合,图书馆为其保留18周供短期借阅,其中主要参考文献来自于主动簇 OSBook 中的 b_{100} 以及 Journal 中的 j_2 和 j_{200}. 故需将簇 OSBook 中的成员对象 b_{100} 增加一个附加属性〈Reserve,string,"OSCourse Reference"〉,用来说明该书已被保留为课程所用.需按课程参考书的规则借阅.对于 j_2,j_{200} 亦需作类似处理.注意,角色 Major 的扮演者是异质对象,分别引自两个不同的主动簇,角色自的属性是其扮演者的临时属性. 18周后,OSReferences 及其角色均将被删去,b_{100},j_2,j_{200} 等书和杂志也将恢复正常借阅,即 b_{100} 等对象的附加属性 Reserve 也将被删去.

3.2 与强类型和弱类型模型的比较

与基于类的强类型模型相似,簇有一个属性集、方法集及对象的集合.但簇是动态定义的集合对象,而类是静态定义的对象类型.虽然可以通过实例化动态地创建和删去类对象,但它们囿于类定义的同质对象,而簇中的对象可以为异质的.类中的属性和方法不能动态地加入或删除,而簇则可以.因此,CCM 模型具有弱类型动态模型化的特征,可灵活地适应动态、异质、对象结构不完全已知的应用环境.例如,若要收藏 CD,VCD 等资料,则只需在簇 Library 中增加角色 MultiMedia 及扮演者 CD,VCD 等子簇,无须修改原模式.

类强调的是对象的共性,而簇中成员可通过它们所扮演的不同角色表达其个性.同一对象在同一个簇中可扮演不同的角色,而同一角色的扮演者可以是异质对象的集合.这些特征可方便地从已存在的对象中动态地创建具有病态结构的临时性组合对象(即被动簇),其中的成员对象可通过加入簇和角色的属性及方法而被赋予新的性质.因此,被动簇比基于类的视图具有更强的表达能力.

簇层次与强类型模型中的组合层次相似,但由于子类对象集通常是超类对象集的子集,故簇层次亦可表达类层次,此时,子、超类间的继承性由簇层次相应的角色层次来表示.类层次不支持值共享属性,而角色层次则区分并支持结构共享和值共享属性,也支持值共享属性的继承.组合层次缺乏的继承性则由簇性质的选择继承来解决.此外,簇和角色的性质严格区分了强类型模型中类和对象的性质.因为簇是一个对象,故簇层次上的属性继承通常是对象间的值继承,强类型模型不支持这种继承,但它在 VDBMS 中特别有效[6].

通常可将对象共享的较为稳定的属性抽象到它们所扮演的角色中进行定义,角色和这些扮演者就相当于形成了类和实例的关系,故无须像弱类型模型那样重复地为每个对象说明相同的结构.而对象的不稳定的、相异的属性则作为对象的附加属性加以说明,这类属性与弱类型模型的对象说明相似,是一种"自我描述"的无模式属性.因此,CCM 模型既具有弱类型模型动态模型化的优点,又由于角色和扮演者集合保留了强类型模型的类和实例机制,簇层次和角色层次保留并扩充了强类型模型的性质继承机制,因此它也克服了弱类型模型因缺乏这两种复用机制而产生的缺点.

4 结 论

本文对目前的 OODBMS 数据模型的基本特征进行抽象,将它们区分为强类型和弱类型两大类.通过详细分析这两类模型的优缺点,首次提出了区分值共享、结构共享属性和类属性以及支持组合层次上选择继承的必要性.在对这两类模型折衷的基础上提出了一个新的基于 CCM 的动态数据模型,它既保留了强类型的复用机制,又具有弱类型动态模型化的特征,较好地解决了这两类模型中存在的问题.因此,它可用来对更高级、更复杂的系统进行模型化,以降低系统设计的复杂

性,它已在我们的VDBMS项目中起到了关键作用[6,7].

References:

[1] Bertino, E., Martino, L. Object-Oriented database management systems: concepts and issues. IEEE Computer, 1991,24(4):33~47.

[2] Ullman, J. D. Principles of Database and Knowledge-Base Systems. Rockville, MD: Computer Science Press, 1988.

[3] Catell, R. G. G. Object Data Management. Reading, MA: Addison-Wesley Publishing Company, 1991.

[4] Papakonstanntinou, Y., Garica-Molina, H., Widom, J. Object exchange across heterogeneous information sources. In: Proceedings of the IEEE ICDE. IEEE Computer Society Press, 1995. 251~260.

[5] Kim, W. Object-Oriented databases: definition and research direction. IEEE Transactions on Knowledge and Data Engineering, 1990,2(3):327~341.

[6] Huang, L. S., Lee, C. M., Li, Q., *et al*. An experimental video DBMS based on advanced object-oriented techniques. In: Sethi, I. K., Jain, R. C., eds. Proceedings of the SPIE. San Jose, CA: IS&T/SPIE Society Press, 1996. 158~169.

[7] Li, Q., Huang, L. S. A dynamic data model for a video database management system. ACM Computing Surveys, 1995,27(5):602~606.

A New Dynamic Data Model for Object-Oriented Database Systems*

HUANG Liu-sheng, CHEN Hua-ping, ZHENG Qi-long, CHEN Guo-liang

(*Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China*);
(*National High Performance Computing Center, Hefei 230027, China*)

E-mail: lshuang@cs.ustc.edu.cn

http://cs.ustc.edu.cn

Abstract: The current data models used in OODBMS (object-oriented database management systems) are basically divided into two categories: traditional 'type-strong' and non-traditional 'type-weak'. The feature of the former is that the objects sharing the same structure (schema) and behaviors (methods) are grouped into a class. The schema of class to which the objects belong interprets the meaning of the objects. Therefore a class must be predefined before its objects are defined. The latter is characterized by schema-less, i.e., the meaning of the objects is 'self-describing', and hence the schema need not to be defined in advance for the objects. In this paper, after outlining the main advantages and disadvantages of these two kinds of models, the authors prove that it is necessary to find some tradeoffs between traditional 'type-strong' data models and non-traditional 'type-weak' ones in order to adequately accommodate the efficient and flexible manipulation of dynamic objects. For this reason, a novel dynamic data model is proposed based on CCM (conceptual clustering mechanism) which has been successfully used for the development of an object oriented VDBMS (video database management system).

Key words: conceptual clustering mechanism; type-strong data model; type-weak data model; cluster; role; player

* Received November 15, 1999; accepted March 17, 2000
Supported by the National High Technology Development Program of China under Grant No. 863-306-ZD06-2