doi:10.11959/j.issn.1000-436x.2015151

数据库形式化安全策略模型建模及分析方法

王榕[1,2]，张敏[1,3]，冯登国[1]，李昊[1]
（1. 中国科学院 软件研究所 可信计算与信息保障实验室，北京 100190；2. 中国科学院大学，北京 100190；
3. 中国科学院 软件研究所 计算机科学国家重点实验室，北京 100190）

摘 要：目前数据库形式化安全策略模型存在抽象层次较高、缺乏对数据库状态与约束的充分描述等问题，难以辅助用户发现商用数据库设计中的微小缺陷。提出了一种基于 PVS 语言的数据库形式化安全策略模型建模和分析方法，该方法较以往模型能够更加贴近实际数据库，应用范围更广，安全属性描述更加完整，描述的模型具有灵活的可扩展性，并且保证了建模与验证的效率。最后，将该方法应用于数据库管理系统 BeyonDB 的安全策略建模分析中，帮助发现了系统若干设计缺陷，证明了方法的有效性。

关键词：形式化建模；数据库；定理证明；安全策略模型
中图分类号：TP309 **文献标识码**：A

Formal modeling and analyzing method for database security policy

WANG Rong[1,2], ZHANG Min[1,3], FENG Deng-guo[1], LI Hao[1]
(1. Trusted Computing and Information Assurance Laboratory, Institute of Software,Chinese Academy of Sciences, Beijing 100190, China;
2. University of Chinese Academy of Sciences, Beijing 100190, China;
3. State Key Laboratory of Computer Science, Institute of Software,ChineseAcademy of Sciences, Beijing 100190, China)

Abstract: Because of the high-level abstraction, insufficient description of database states and constraints, it was difficult to find the tiny flaws in design and implementation. Based on PVS, a method for formal description and analysis of database security policy was proposed, which was more close to the actual database, more widely used in reality, and more complete in describing the safe properties, more extendible of the model, and ensure the efficiency of modeling and verification. Finally, this method is applied in the security policy modeling and analyzing of BeyonDB, which is a commercial database, find some security risks in the system design, and thereby verify its effectiveness.

Key words: formal modeling; database; theorem proving; security policy model

1 引言

数据库管理系统是多数信息系统的重要基础平台，其安全性是信息系统安全中的重要环节。近年来，伴随着国家信息化建设飞速发展，我国对数据库系统的安全也逐步重视，已有一批达到国标[1]第三级产品，在军用、政务、电力、通信、铁路等国家重要领域逐渐得到应用推广，取代国外同类产品，发挥骨干支撑作用。但是随着国家信息化程度的提高，现有第三级数据库产品已经逐渐不能满足安全需求。因此，迫切需要开展高安全等级数据库构建与评估技术的研究。而本文关注的安全策略模型建模与分析就是其核心技术之一。

安全策略模型是独立于软件实现的高层概念模型，它来源于需求规约，描述安全系统的安全保护需求与功能性质[2]。安全策略模型的形式化分析是评估第四级系统[1]的重要内容之一，在许多应用领域都扮演着重要的角色，如建立可信操作系统、可信数据库管理系统等高可信软件系统。目前，通过一系列形式化方法在工程中的应用[3~5]，对系统进行形式化描述和验证不仅可以发现系统设计上的问题、验证系统设计的正确性，而且还可以指导

收稿日期：2014-07-12；**修回日期**：2015-03-25
项目基金：国家自然科学基金资助项目（61232005, 61402456）
Foundation Item: The National Natural Science Foundation of China (61232005, 61402456)

开发、检验系统实现的正确性等。然而在数据库安全领域，现有的形式化方法还不能满足实际需求。此外形式化方法在很大程度上受到其支持工具的限制，一些语言和分析方法由于缺乏原生分析工具的支持，导致验证效率不能满足大型复杂系统的工程需求。如 Z 语言，在 20 世纪 80 年代末到 90 年代初国际上相继提出了很多 Z 语言的扩展，并进行了一些标准化工作，提出了与形式化方法和面向对象方法相结合的思想[6,7]。然而，虽然基于 Z 语言的形式化方法和相关工具得到了发展，但是仍然无法支持在数据库等大型复杂系统的工程应用。

针对这些问题，本文提出了一种高等级数据库形式化安全策略模型的建模方法，并将其应用于实际商用数据库 BeyonDB 中进行有效性验证。该数据库经相关测评机构测评已达到第三级，并具备部分第四级安全特性。将本文提出的建模与分析方法应用于 BeyonDB 安全功能设计与开发中，帮助发现并纠正了系统若干设计缺陷，取得了良好的实际效果。

2 相关工作

近年来，国内外已逐渐开展了形式化方法在各类模型及实际系统中的安全分析的研究，如操作系统、防火墙等。Qian 等[8]采用数学形式化方法，使用 OSOSM 对操作系统进行建模并采用 Isabelle 验证，保证了系统的高度安全性。Yang 等[9]从完整性模型的实用性角度出发，提出一种新型的动态完整性保护模型（DMIP, dynamic integrity protection model），并给出了形式化分析。还有部分工作提出了在形式化安全策略模型验证的基础上对目标系统进行测试的方法。数据库的安全策略模型形式化分析，由于其分析规模和主客体的复杂性，目前仍然缺少实际商用的安全策略模型的形式化建模与分析方法。

Bell 和 LaPadula 在 1975 年提出了 BLP 经典模型[10]，基于该模型提出了保护数据机密性的重要思想，为后来数据库管理系统各种安全策略模型的建立奠定了坚实的基础。SeaView 模型[11]基于 BLP 经典模型，详细提出了 16 条完整性约束，分析了包括自主访问控制（DAC, discretionary access control）和强制访问控制（MAC, mandatory access control）等安全模型，访问控制粒度实现元素级，为多级安全数据库的设计和分析提出了良好的基础。但是，SeaView 模型[11]采用的是 TCB 子集架构[12]，将数据库管理系统与 TCB 分开描述，不能满足当今大部分采用可信主体架构[12]的商用数据库的形式化分析需求。Li 等[13]提出了一种基于 Z 语言的策略模型形式化方法，并且以 BLP 模型为例进行了验证。Zhu 等[14]扩展了 SeaView 模型的客体结构，在丰富了客体类型的同时扩展了安全属性，并用 Gallina 语言对其进行了形式化描述，在 Coq 定理证明器帮助下对模型进行了证明。Sandhu[15]提出了基于角色进行访问控制的 RBAC（role-based access control）模型，该模型将权限与角色相关联，通过角色获取的权限进行访问控制，极大地简化了权限的管理。He 等[16]不仅提出了详细的安全属性，还将 BLP 模型和 RBAC 模型进行了逻辑结合，采用 Z 语言对该模型进行形式化描述，实现了多策略的数据库管理系统访问控制。

上述相关工作多数集中于对 DBMS 的通用模型部分进行研究，其访问控制粒度较粗，不满足高等级系统要求；或提出新的数据库安全策略模型，并进行形式化分析，没有考虑数据库系统完整性约束等固有属性，缺少从实际数据库系统建立安全策略模型的方法。更加缺少对实际商用数据库管理系统的安全策略模型进行描述和分析。

基于以上原因，本文提出了一种数据库形式化建模和安全性分析方法，该方法建立的模型能够提供更加完整的安全属性描述，更加灵活的可扩展性，保证了建模和验证的效率，并且应用于实际数据库系统，发现数据库安全功能设计和实现中的错误和缺陷，验证了其有效性。

3 PVS 语言及定理证明器

本文采用了国际上较为流行的 PVS 语言和验证工具作为形式化方法的基础。该系统由斯坦福研究机构开发，包括 PVS 规约语言和 PVS 定理证明器 2 部分组成：前者基于高阶逻辑，具有丰富的数据类型系统，表达能力强；后者采用目标制导的方式工作，即一个证明的构造从待证明的定理开始，逐步推理，不断生成子目标，直至所有子目标均为真。PVS 定理证明器拥有强大灵活的命令集。高效的决策过程、高度交互与自动化的定理证明使 PVS 在当今形式化语言中较有优势。

PVS 语言不但拥有强有力的推理机制，而且有丰富的表达手段，目前已在很多领域得到应用。典

型应用包括：我国国内可信电子电力系统的建模与证明、轨交控制系统的SCADE开发和国际上NASA的对基于时间触发以太网的时钟同步协议的形式化验证等。

4　安全策略形式化描述方法

国标 GB17859-1999[1]中指出：为了满足结构化保护系统功能的安全性要求，安全策略模型的形式化应该至少包含自主访问控制和强制访问控制。因此，以 BeyonDB 系统对 BLP 模型的实现为例展示本文的建模方法，流程如图 1 所示。为使形式化模型与实际数据库保持高度的一致性，将形式化安全策略模型的描述和安全性分析分为 3 个部分。

1) 模型目标提取：抽取必要的实体与操作。

2) 系统状态、操作规则生成与描述：对数据库相关实体进行分析、描述、综合生成系统状态；利用系统状态中描述的实体等内容描述操作规则。

3) 描述安全定义并进行安全性分析：描述系统安全定义，对模型进行安全分析，交互式证明安全定理。通过安全分析发现系统安全缺陷，指导修改系统设计。重复以上步骤，不断完善形式化模型。

4.1　模型目标提取

对于形式化模型中应该具体描述的实体，抽取途径主要是通过系统中顶层的访问控制接口、安全需求以及数据库系统实际的系统表，采用基于操作接口的目标提取方法来获取描述目标，即对于一个指定的访问控制接口，统一以操作者、操作目标、操作成功判断条件的形式对每一个操作接口进行描述和分析。如表 1 所示，在 BeyonDB 数据库中，根据实际操作接口 Insert，结合实际存在的系统表 iiuser，将操作者抽象为模型中的主体：User；根据操作的对象实体表格，抽象出客体：Realtable；同时在实际的强制访问控制和自主访问控制中需要判断实关系表和操作者的标签支配关系，所以需要对标签（Label）和标签支配判断函数（dom?）进行描述并定义。按此方法对系统中所有接口进行描述，分类总结出系统的基本元素，为下一步的形式化语言描述提供必要基础。

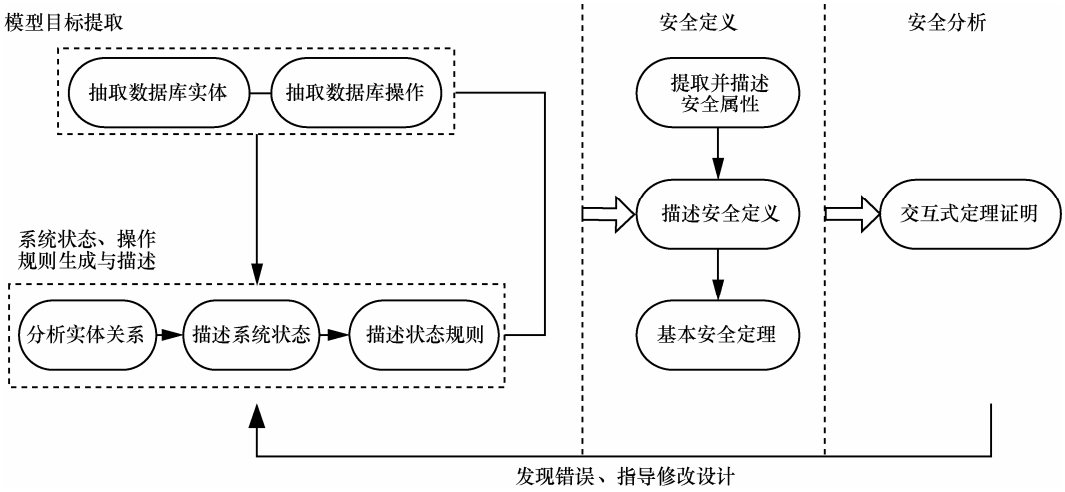

图 1　形式化建模与安全分析流程

表 1　　操作 **Insert** 的描述目标提取

描述对象	非形式化描述	抽象描述
操作	INSERT INTO realtable VALUES tuple	操作实体：Operation
操作者	数据库用户	主体：User
操作目标	实关系表	客体：Realtable (隐含指定 Database)
操作成功判断条件	1. 当前数据库在数据库系统中存在 2. 用户在当前数据库系统中存在 3. 实关系表在当前数据库系统中存在 4. 主体是客体的属主或者拥有 insert 客体权限 5. 用户的安全标签要支配实关系表格的安全标签	1. 数据库存在判断：DatabaseExist 2. 用户存在判断：UserExist 3. 实关系表存在判断：RealtableExist 4. 客体属主判断：Owner（或者 insert 权限判断） 5. 标签支配判断：dom? 6. 涉及客体：标签 Label，权限 Permission

4.2 系统状态生成与描述

完成了上述基于操作接口的目标提取后，使用 PVS 语言对提取出的元素进行描述，描述内容包括实体、与数据完整性相关的约束元素、访问控制相关元素、系统状态、操作规则。在描述中，为了使形式化模型具有较高的可读性和验证的效率，充分应用 PVS 语言中一些内置类型、类型构造器及函数。例如采用 PVS 语言中的集合类型来描述实体，在判断某一类型的实体时，利用 PVS 语言内置函数 member 能够方便判断某一元素是否存在于某一指定的集合当中。如系统状态，为了快速提取状态中某一内容，采用记录类型对状态进行描述，同时也提高了代码的可读性。

4.2.1 实体描述

1) 主体存在

本文使用集合类型对各类实体进行描述，优点是可以直接利用 PVS 定理证明器内置的 member 和 subset 函数来判断某一个元素是否在一个集合中或一个集合是否是另外一个集合的子集。同时为了增加语言的可读性和使用的灵活性，采用记录类型描述所有主体集合（主体存在 Subject_Exist），每一类主体集合作为记录中的一个域，例如 UserExist 是主体的集合，RoleExist 是角色的集合。主体存在描述具体如图 2 所示。

```
Subject_Exist:type=[#userexist:UserExist,
roleexist:RoleExist,profileexist:ProfileExist
groupexist:GroupExist#]
```

图 2 主体存在描述

2) 主体映射

主体映射描述了数据库中相关联主体间的对应关系，如用户与组、用户与轮廓以及用户与角色等。各类对应关系均采用 PVS 语言中的函数类型来描述，如引入函数类型映射 UsertoType:TYPE=[USER->USER_TYPE]描述用户到用户类型，再将各类与主体相关的函数映射组织成记录类型 Subject_Map（如图 3 所示），Subject_Map 的生成为自主访问控制和强制访问控制提供了方便的依据。

```
Subject_Map:TYPE=[#usertotype：UsertoType,
profiletopermission:ProfiletoPermission,
roletopermission:RoletoPermission,
profiletotype:ProfiletoType,roletotype:RoletoType,
usertogroup:UsertoGroup,grouptouser:GrouptoUser,
usertoprofile:UsertoProfile,pofiletouser:ProfiletoUser,
usertorole:UsertoRole,roletouser:RoletoUser#]
```

图 3 主体映射描述

3) 客体存在

将除数据库客体以外的其他客体分为 2 类，一类是具有层次结构的客体，如实关系表、元组、视图；另一类是不具有层次关系的客体，如事件、序列、过程。模型中，将客体组织成如下树形结构，如图 4 所示。为了描述数据库中复杂的客体和繁多的完整性关系，使用记录格式 ObjectExist 描述模型中不同类型的客体，每一类客体集合作为记录一个域，分别包括数据库、实关系表、视图和元组。

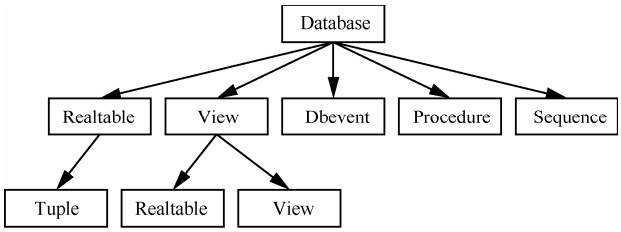

图 4 数据库客体层次结构

4) 客体映射

客体映射是各客体间的层次关系的抽象描述。通常采用的描述方法是，参照 BeyonDB 设计文档，依据系统中各类系统表，将相关系统表关联并抽象定义。例如，将设计文档中 iintegrity 与 iikey 这 2 张系统表进行关联并抽象，得到关系表与其引用表的映射 Realtable_Reftable。为方便描述数据库系统的引用完整性这一固有属性，要对必要的实体间映射进行描述，包括关系表与被引用表的映射 RefTable_RealTable、元组与主键的映射 Tuple_PKey、元组与外键的映射 Tuple_FKey。客体间的映射函数应至少包含上述内容，映射关系如图 5 所示。以上映射函数的描述为下文安全属性定义提供基础。

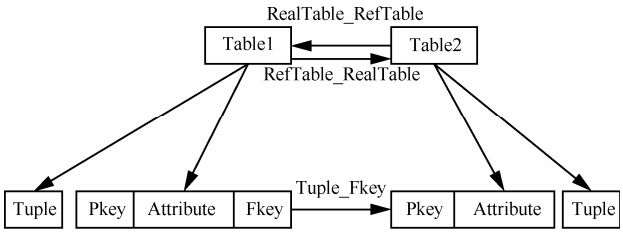

图 5 客体关系映射

4.2.2 策略相关元素描述

自主访问控制和强制访问控制的形式化证明是国家相关标准要求中的必要内容。以 BeyonDB 为例，自主访问控制通过以下步骤进行：首先获得该用户拥有的权限，其次判断指定权限是否在其中。按照系统设计文档对该步骤进行细化，分别定

义主体权限、客体权限、数据库权限。为了满足系统的可信主体权限最小化原则，将主体权限细分为系统权限、审计权限、安全权限，相应定义了用户到其拥有的不同类型权限的函数映射，并组织为记录类型 DAC_Table。强制访问控制的核心操作是标签支配判断，首先分别获得主客体标签，然后进行标签对比判断。由此定义 2 个函数：主体到主体标签的函数映射和客体到客体标签的函数映射。最后，根据断言 dom？如图 6 所示定义来进行标签支配判断（如下规约描述，返回一个 BOOL 类型）。

```
DAC_Table:TYPE=[#
Object_perms_index:[[Subject_ID,Object_ID]->setof[O
bjectPermission]],Subject_perms_index:[Subject_ID->setof[SubjectPermi
ssion]],
Database_perms_index:[Subject_ID->setof[DatabasePer
mission]],
Owner:[Object_ID->User_ID],
Grant_perms_index:[[Subject_ID,Object_ID]->setof[Ob
jectPermsFlag]]#]
dom?(label_a:Label_ID,label_b:Label_ID):
bool=(label_a`level>=label_b`level)AND
(subset?(label_b`category,label_a`category))
```

图 6　dom?规约描述

同时为了增加系统的灵活性，能够在会话当中动态调节主客体的标签，根据系统中描述的最低、最高安全等级，最小、最大范畴，定义模型中安全等级范围和范畴范围，如图 7 所示，均采用记录类型描述，这样便于对极限值的获取。另外，为了记录登录用户和数据库的相关信息，定义一个会话记录，通过会话记录（下面表达式中的 session）可以方便提取与登录用户和数据库相关的内容。在安全约束属性描述中，为了给安全定义中的状态属性的证明提供依据，对当前用户的合法操作进行记录，定义一个记录类型的动作记录 Action_ID，再根据访问控制不同分别用 Cur_Perms_DAC 和 Cur_Perm_MAC 表示记录的集合。

```
LevelRange:TYPE=[#defaultlow:Level_ID,
defaulthigh:Level_ID#]
CategoryRange:TYPE=[defaultnull:Category_ID,
defaultall:Category_ID,
database_id:Database_ID#]
Action_ID:TYPE=[Operation_ID,Object_ID]
Cur_Perms_DAC:TYPE=setof[Action_ID]
Cur_Perms_MAC:TYPE=setof[Action_ID]
```

图 7　级别、范畴规约描述

4.2.3　系统状态描述

一个数据库的系统状态应该包含所有评估者关心的数据，仅对模型各单位元素进行描述是不充分的。本文使用记录类型描述系统状态 system_state，并将不同类型的实体、关系映射、访问控制等内容组织成系统状态中的一个域。这样可以根据操作的需要从系统状态中提取出相关的信息。system_state 通过具体操作、安全属性集合及实体关系集合进行获取生成，具体算法如下。

系统状态获取算法

算法 GetSystemState

输入：操作集合 OPERATIONS ，安全属性集合 INVARIANTS，实体或关系集合 VARS

输出：系统状态 SYSTEM_STATE

1) BEGIN
2) FORALL *var* in VARS{
3) FORALL *op* in OPERATIONS{
4) IF(*var* in *op*'s condition){
5) add(*var*,SYSTEM_STATE);
//若执行操作 *op* 的判断条件包括 *var*，则添加 *var* 到系统状态
6) break;}
7) IF(*op* changes *var*){
8) add(*var*,SYSTEM_STATE);
//若执行操作 *op* 会改变 *var*，则添加 *var* 到系统状态
9) break;}
10) }
11) FORALL*inv* in INVARIANTS{
12) IF(*inv* contains *var*){
13) add(*var*,SYSTEM_STATE);
//若安全属性 *inv* 中有 *var* 描述，则添加 *var* 到系统状态
14) break;}}}
15) Return SYSTEM_STATE
16) END

4.2.4　操作规则

操作规则（operation rules）描述了模型是如何具体实施安全策略的，揭示了 2 个连续状态的状态转换过程中状态变量之间的关系。针对不同操作，操作规则包括操作发生条件与操作执行影响 2 部分。操作发生条件即操作成功判断条件，如强制访问控制的标签支配判断、自主访问控制授权检查。操作执行影响主要描述的是操作执行后系统状态变量的改变。操作规则的产生是将操作判断条件和操作执行后系统状态变量的改变进行逻辑组合，具体算法如下所示。

操作规则生成算法

算法 Operation Rules Generation

输入：系统状态 SS0，基于操作接口的目标提取方法中的各项操作成功判断条件的逻辑与集合 CONS，初始操作规则序列 Rules

输出：操作规则序列 Rules

1) Begin
2) SS1=SS0
3) Rules=NULL
4) Rules = Rules AND CONS
5) FORALL *var* IN SS0{
6) IF(*op* changes *var*)
{
7) Rules=(Rules AND SS1=SS1 WITH *op*(*var*));

//如果操作 *op* 改变了系统变量 *var*，则操作规则序列添加改变后的状态 SS1

8) break;
9) }
10) Return Rules
11) END

下面以 Update 为例展示由该算法生成 Do_update 操作规则。

1) 输入：rt_id_update：欲修改表格
2) t_id_update：欲修改元组
3) pkey_id_update：新元组主键
4) fkey_id_update：新元组外键
5) SS0，SS1：操作前后系统状态

//操作成功判断条件逻辑与集合 CONS

6) exist(rt_id_update)

//表格存在判断，数据库、元组同样方法判断，主键必须非空篇幅有限不全部描述

7) AND if pkey_id_update=newid thenpkey_id_update /=other tuple fkey

//如果修改元组主键则元组的主键在其他表中不能作为外键

8) AND if fkey_id_update=newid then fkey_id_update=one tuple pkey

//若修改元组外键则修改后的外键值必须为某个已经存在被引用表元组主键

9) AND have_object_permission(update) or user=rt_id_update.owner

//DAC 自主访问控制检查，需要有 UPDATE 权限或者为该表格属主

10) ANDdom?(LabelofUser(SS0.uesr_id), LabelofRealtable(rt_id_update))

//主体标签支配客体标签等级

11) ANDLabelofUser(SS0.user_id) =LabelofTuple(t_id_update)

//用户标签等于元组标签等级

//操作执行后状态改变：不变的系统状态变量有 object_exist(SS1),mac_table(SS1), dac_table(SS1), subject_exist(SS1), subject_map (SS1), cur_session_info(SS1), usertolrange(SS1),usertocrange(SS1)；

变化的系统状态变量:添加 update 到 cur_perms_DAC 和 cur_perms_MAC 中

12) AND cur_perms_MAC(SS1)=add((update,t_id_update), cur_perms_MAC(SS0))
13) AND cur_perms_DAC(SS1)=add((update,rt_id_update),cur_perms_DAC(SS0))
14) AND cur_op(SS1)=update
15) AND{

//新元组主键改变，则增加新元组主键并移除旧主键和元组的对应关系；新元组外键改变并且为新外键，则增加新元组外键并且移除旧外键与元组的对应关系

16) if (fkey_id_upadate=newid Andfkey_id_update/=other_tupe_fkey) THEN change(object_map,object_exist)
If (pkey_id_update=newid THEN change(object_map,object_exist)
17) END

5 形式化策略模型安全分析

5.1 安全定义

在国标[1]对第四级的信息系统明确要求了自主访问控制和强制访问控制，此外针对不同的信息系统，还有其他特别的安全需求。对具体的数据库系统来说，安全属性的定义应该从 2 个方面进行该考虑，首先是为满足国家标准的安全属性内容，主要是以经典安全策略模型作为基础（自主访问控制、强制访问控制）；其次是针对不同系统的具体安全需求。例如，数据库需要添加的固有的一些安全属

性（如实体完整性、引用完整性、用户自定义完整性）。通过将要求的安全需求内容与实际系统安全需求内容结合定义出多条安全属性。在安全属性基础之上，根据形式化验证目标将不同安全属性通过 PVS 中的逻辑运算符“and”进行连接来定义状态安全，如图 8 所示。具体地，针对 BeyonDB，本文将数据库安全属性分为 6 条，包括主体标签支配会话标签，客体间标签支配（关系标签支配数据库标签，元组标签支配关系标签），自主访问控制，简单安全，*安全以及引用安全属性。

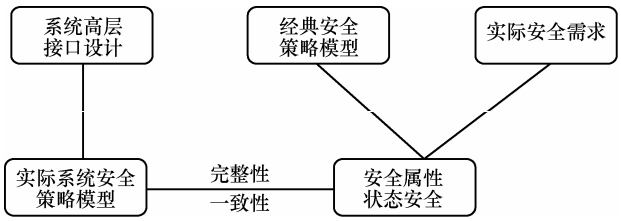

图 8　安全定义与安全性分析

安全属性 1（主体标签支配）：会话的发起者即当前用户必须支配当前会话的安全标签。

```
Subject_Label_Invariant?(ss:system_state):
bool=dom?(LabelofUser(mac_table(ss))(user_id),
LabelofSession(mac_table(ss))(cur_serssion_info(ss)))
```

图 9　主体标签支配描述

安全属性 2（客体间标签支配）：为了达到更细粒度的标签控制，需要满足以下条件：关系的安全等级支配其所属数据库的安全等级；元组的安全等级支配其所属的实关系表的安全等级（如图 10 所示）。

```
Object_Label_Invariant?(ss:system_state):
bool=forall(relation:Relation_ID,database:Database_I
D):
member(relation,relationexist(object_exist(ss))
(database_id(cur_session_info(ss))))
imples
dom?(LabclofRclation(mac_tablc(ss))(rclation),
LabelofDatabase(mac_table(ss))(database_id(cur_sess
ion_info(ss))))
ANDforall(tuple:Tuple_ID,relation:Relation_ID):
member(tuple,tupleexist(object_exist(ss))(relation))
impliesdom?(LabelofTuple(mac_table(ss))(tuple),
LabeofRelation(mac_table(ss))(relation)
```

图 10　客体标签支配描述

安全属性 3（自主安全）：根据 cur_perms_DAC，如果当前用户对某些客体进行过操作合法，那么该用户必然是当前客体的属主或者对该客体具有相应的客体权限（如图 11 所示）。

```
DAC_Security_Invariant?(ss:system_state):
bool=forall(action:Action_ID):
member(action,cur_perms_DAC(ss))implies
(Owner(dac_table(ss))(object_id(action))=user_id(cu
r_session_info)OR
member(OperationtoPerms(operation_id(action)))=
Object_perms_index(dac_table(ss))
(user_id(cur_session_info(ss)),object_id(action))))
```

图 11　自主安全描述

安全属性 4（引用完整性）：数据完整性对数据库来说至关重要，数据库中的数据是否满足完整性关系到其能否真实反映出现实世界，能否在数据库中保护数据的正确性、有效性和相容性。尤其对于引用完整性，BeyonDB 系统中采用了 3 种不同的策略，级联删除、受限删除、置空删除。可以根据需要采用不同的默认策略，这里限于篇幅原因描述受限删除时的系统引用完整性如图 12 所示，要求某一元组的外键要么全空，要么全部非空（此种情况必定存在相应的一个被引用的表的元组，使其主键等于该元组的外键并且安全等级应该支配被引用表的安全等级）。

```
Reference_Invariant?(ss:system_state):
bool=forall(rt_id:Realtable_ID,t_id:Tuple_ID):
if
member(rt_id,RealtableExist(database_id(cur_session_inf
o(ss))))
ANDmember(t_id,Tuple_Exist(rt_id))implies
Tuple_Fkey(t_id)=nullID
ORexist(rt_id2:Realtable_ID,t_id2:Tuple_ID)
:member(rt_id2,RealtableExist
(database_id(cur_session_info(ss)))) AND
member(t_id2,TupleExist(rt_id2))
AND Tuple_Pkey(t_id2)=Tuple_Fkey(t_id)
AND dom?(LabelofTuple(t_id),LabelofTuple(t_id2))
```

图 12　引用完整性描述

安全属性 5（简单安全性）：根据具体的数据库的要求将 BLP 模型调整为：如果一个用户对某客体进行了读的操作，那么当前用户的安全标签应该支配该客体的安全标签从而达到不上读的要求（如图 13 所示）。

```
Simple_Security_Invariant?(ss:system_state)
:bool=forall(action:Action_ID)
:member(action,cur_perms_MAC(ss))
AND(PermstoType(OperationPerm(proj_1(action)))
=(rcadonlyorrcadwritc))implics
Dom?(LabelofUser(mac_table(ss))
(user_id(cur_session_info(ss)))
LabelofRelation(mac_table(ss))
(mac_table(ss)(proj_2(action))))
```

图 13　简单安全性描述

安全属性 6（*安全性）：如果一个用户对某客体进行了写操作，那么该客体的安全标签一定支配当前用户的安全标签以达到不下写的安全要求，如图 14 所示。

```
Star_Security_Invariant?(ss:system_state)
:bool=forall(action:Action_ID)
:member(action,cur_perms_MAC(ss))
AND(PermtoType(OperationtoPerm(proj_1(action)))
=(writeonlyORreadwrite))
impliesdom?(LabelofTuple(mac_table(ss))
(proj_2(action)),LabelofUser(mac_table(ss))(user_id(
cur_session_info(ss))))
```

图 14　*安全

5.2　安全性分析

仅仅定义出模型的各种元素是不充分的，必须对模型进行形式化的分析，即验证模型描述的系统的安全性，系统处于安全的状态。

因此系统状态安全定义应该是上述安全属性、数据库特有属性约束和类型约束的组合。形式化定义如图 15 所示。

```
State_Safe?(ss:System_Sate):bool=Subject_Label_Invariant?(ss)
and Object_Label_Invariant?(ss) and DAC_security_Invariant?(ss)
and Reference_Invariant?(ss) and  Simple_Security_Invariant?(ss)
and Star_Security_Invariant?(ss)
```

图 15　形式化定义

该安全定义是以 PVS 断言的 and 连接生成，可以灵活地添加各类数据库属性作为安全状态的描述。由于 PVS 能够对已证明的片段进行保存，并可将已存的证明片段用于整体的证明过程。故以该方式定义的状态安全在保证不影响已分析过的内容的同时，增加新的安全断言，从而使建模和分析的效率更高。

本文提出的安全性分析的思路是：首先使用 do_init 函数对系统状态进行初始化，得到初始状态 init_ss0，然后根据状态安全定义 State_Safe? 分析初始状态 init_ss0 是否为安全状态；然后，根据状态安全的定义判断任意状态 SS0 是否安全，如果 SS0 安全则通过状态转移规则函数对 SS0 进行操作得到状态 SS1，则 SS1 也是安全的；重复上述操作直至所有可达状态判断完毕，如果所有可达状态都是安全的那么数据库系统就是安全的。由此，需要的安全定理包括：初始状态安全定理和状态转换安全定理。

初始状态安全定理　系统进行初始化操作后得到的状态满足安全状态的定义。即要证明的基本安全定理如图 16 所示。

```
InitialSafe:THEOREM do_init(init_ss0)
impliesSTATE_SAFE?(init_ss0)
```

图 16　初始状态安全定理

状态转换安全定理　针对每一个操作，都应保证对于一个安全状态进行操作 OP 之后得可达状态仍然是安全的。即操作运行之前处于安全状态操作运行之后仍然处与安全状态。

表达式为：状态 SS0 安全 and 操作 OP 则可达状态 SS1 也安全。即需要证明的状态转换安全定理如图 17 所示。

```
SecurityTheorem:THEOREM
(STATE_SAFE?(SS0)ANDop(SS0,SS1))
impliesSTATE_SAFE?(SS1)
```

图 17　安全定理

若能够证明 InitialSafe 和任意操作后的 SecurityTheorem 则系统就是安全的。本文采用 PVS 交互式证明对如上 2 个定理进行证明。并且发现了 BeyonDB 的一些缺陷。

在安全证明中，通常情况下很难一次证明成功，下面结合一些例子，举出 3 类典型的错误原因。

原因一：模型与设计不统一，如 BeyonDB 中要求一张引用表只能应用到一张被引用表，则在形式化描述模型中引用表到被引用的映射关系（单向）则应该是一对一的关系。如果遇到此类问题，需要检查描述的模型，将模型与系统的实际设计严格对应，防止不一致的问题发生。

原因二：模型未消除设计中的二义性，模型内部存在矛盾和不一致性。如模型变量需要常量化时，需要保证该常量符与其他位置同一变量的常量符一致。这种问题的发生是因为对 PVS 证明机制没有清楚了解，所以针对证明过程中使用的命令，需要明确命令的证明过程，尤其涉及到变量常量化的命令。

原因三：模型与设计相符，且描述方面没有问题，仍然无法证明，则发现了系统设计中的确存在缺陷。

5.3　系统存在缺陷

将本文提出的方法对实际的商用数据库 BeyonDB 进行分析，发现其一些典型的问题，并给而解决方案（如表 2 所示），此时需要在系统中进行试验确定系统漏洞，从而发现实际系统的安全缺陷，指导修改系统设计。

发生问题的主要类型有：系统设计本身不完整；操作执行时不满足安全属性。

1) 策略设计不完整：制定策略时，忽略一些细

节，尤其像对 procedure 这种实体进行操作时容易出现问题。因为 procedure 实体涉及到若干个不同安全等级的客体，如果仅仅考虑用户主体安全标签与 procedure 安全标签的标签支配是不充分的。例如一个低等级用户的安全标签等级支配 procedure 的安全标签等级，系统中要求执行者具有 exec 的权限。不关注用户和所有涉及其他客体等级支配关系。当 procedure 中包含高于用户等级的客体时，用户执行该 procedure，则低安全等级用户间接访问了高安全等级的客体，在操作执行后会对操作所涉及的所有客体进行记录，记录（exec,object1），（exec,object2）等。在进行安全证明时，如果低等级用户“读”访问了高安全等级的客体 object1，则证明无法通过简单安全属性，返回系统，修改系统策略，要求在执行 procedure 的过程中，用户的标签安全等级需要支配 procedure 所包含的所有客体的安全等级。

2) 实体设计不完整：设计系统实体时，存在安全隐患。当一个表格在没有设置主键的情况下，允许插入内容完全相同的记录，且没有序列号区分，例如安全等级为 L1 的用户创建一个等级为 L1 的表格 T1，插入一条等级为 L1 的元组 R1，并且对该元组执行了 select 操作，系统会记录该操作（select, R1）。然后提高了用户自身的等级为 L2，并且向 L1 表中插入等级为 L2 的相同内容元组 R1'，此时当系统进行证明时，当遇到相同内容不同等级的 R1 和 R1'时，无法进行判断，不满足简单安全属性。为了更加符合数据库的完整性要求，所以在没有主键的数据库表格中系统要默认插入一列 ID，不允许相同内容的元组存在。

3) 策略执行不满足安全属性：策略设计本身没有问题。但是需要一定的前提条件，必须在满足前提条件的情况下才满足安全属性。

a) 表级、行级开关：在 BeyonDB 中为了增加系统的灵活性，所以设置了表级还有行级的不同粒度的强制访问控制开关。如果只开启表级的强制访问控制开关未开启行级访问控制开关，如果一位支配表级安全标签等级的主体用户就可以访问该表的所有记录，包含行级安全等级高于用户的记录。给出的解决方法是：表级、行级强制访问控制开关必须同时开启。

b) 在会话中，允许用户修改会话标签，用户安全标签修改后，则操作记录中的部分记录会违背安全属性。比如用户对客体 O 进行了读操作（用户安全标签等级支配 O），若该用户提高了自身的安全标签等级且支配 O，则不满足“简单安全属性”和“星”安全属性。给出的解决方法是：在会话过程中不允许用户修改安全标签。

表 2　　错误类型以及系统缺陷

类型	错误	系统设计缺陷
策略设计不完整	安全策略的设计考虑不充分，部分内容会与安全属性发生冲突	Procedure 执行过程涉及到多个不同安全等级的客体
实体设计不完整	实体设计过程中考虑不充分，在安全性证明是发生冲突，无法继续证明	无主键表格插入记录
策略执行约束不满足	1. 策略执行的前提条件不充分满足，与安全属性发生冲突； 2. 策略执行过程中，某些操作会与安全属性发生冲突	表级、行级开关会话过程中，主体标签修改

5.4　方法对比

本文提出的方法与以往方法比较，描述的粒度是记录级别，并且给出了数据库固有的完整性方面的约束，并给出证明；没有采用 TCB 子集架构，这样更符合商用数据库可信系统的要求。将本文方法与其他方法在实际系统相符性、安全属性完整性、可扩展性、建模与验证效率等几个方面进行比较，如表 3 所示。

表 3　　方法对比

指标	SEPOSTG[16]	理论模型分析方法[7]	本文方法
实际系统相符性	中	未讨论	高
安全属性完备性	较差	未讨论	较完备
可扩展性	中	高	高
建模与验证效率	中	低	高

与实际系统设计相符性：提供了通过系统的变量以及系统表等内容获得实体与关系的方法，不仅仅局限于访问控制的理论模型，而是将实际系统的设计细节反映在模型中，如针对客体定义 Owner 获取客体的属主，定义 ProcedureToObject 描述过程与客体的映射，定义 Tuple_Fkey 描述客体元组到外键的映射。针对主体，定义 UserToRole，UserToGroup 来描述用户到角色，用户到群组的映射。考虑实际操作系统，定义出更加详细的内容。因此实用性更强，这是传统理论模型分析方法中[7]并没有涉及到

的内容。文献[7]中只将客体简单描述为单一客体 O_Single 并未详细的考虑客体间、主体间的详细内容，仅仅针对 DBLP 涉及的内容进行描述，局限性较强，并不适合实际的系统。文献[16]中只对粗粒度的 Database 和 Table 进行描述，并未对细粒度的 Tuple 等进行讨论。

安全属性更加完备：安全属性除了需要满足安全测评相关标准对访问控制的需求外，还需结合实际数据库系统，将实体完整性等考虑在内，而文献[16]和文献[7]并没有考虑完整性。文献[7]关注 BLP 模型文献[16]关注 RBAC 和 BLP 模型，仅考虑系统中数据的机密性，忽略数据库完整性。

可扩展性：作为结构化建模方法的保证，采用了可扩展性的构架，在系统状态、安全定义、操作描述等方面采用 PVS 中的逻辑“与”描述使结构相对于其他方法更加易于扩展，文献[16]描述了不同的不变式，但是受到定理证明器的限制无法将不变式进行逻辑组合，导致部分不变式需要手工证明。

建模与验证效率：迭代式的建模与证明过程使模型修改容易，迭代代价较小，并且 PVS 原生工具的支持使本文方法在实际工程中有更高的效率。其他文献中，早期是采用手工证明，或者采用 Z 语言描述，但相应证明器不能提供有效的推理支持，所以在大量需要展开的模式时效率非常低，这就要求在模型描述的时候进行优化，但 Z 语言描述的模型在修改方面非常困难，故证明的工作量也会增加许多。文献[7]中指出，当展开大量的模式时，定理的证明速度明显放慢。而文献[16]中针对部分的不变式无法采用机器证明，故采用人工证明，大大降低了证明的效率。

综上，本文方法对于大型复杂系统在实际设计开发中应用，能给与足够的支持，帮助发现系统漏洞，指导系统开发并且在系统安全性建模和验证效率方面都拥有更大优势。

6　结束语

本文提出了一种数据库形式化安全策略模型建模与分析方法，为当前第四级安全数据库管理系统的安全策略模型建模与分析提供参考。该方法基于设计文档分析，抽象提取出安全定义和操作规则：前者体现系统关键安全需求，定义模型中状态与状态转移间的约束；后者体现实施安全需求的方式。该方法在 BeyonDB 数据库安全策略模型分析中获得了很好的效果，显示出该模型安全属性上的可扩展性和灵活性。采用的 PVS 语言以及其原生的定理证明器使该方法效率得到保障，证明该方法具有良好的可行性。

参考文献：

[1] 国家质量监督检验检疫总局.GB17859-1999 计算机信息系统安全保护等级划分准则[S].北京:中国标准出版社,1999.
General Administration of Quality Supervision, Inspection and Quarantine of P.R.C. GB17859-1999 Classified Criteria for Security Protection of Computer Information System[S]. Beijing:Standards Press of China, 1999.

[2] 张敏, 冯登国, 陈驰.基于安全策略模型的安全功能测试用例生成方法[J].计算机研究与发展,2009,46(10):1686-1692.
ZHANG M,FENG D G, CHEN C. A security function test suite generation method based on security policy model[J].Journal of Computer Research and Development, 2009, 46(10):1686-1692.

[3] 官尚元,伍卫国,董小社.自动信任协商的形式化描述与验证研究[J].通信学报,2011,32(3):86-99.
GUAN S Y, WU W G. DONG X S. Research on formal description and verification of automated trust negotiation[J]. Journal on Communications, 2011, 32(3):86-99.

[4] LUO X Y, TAN Z, SU K L.A verification approach for web service compositions based on epistemic model checking[J].Chinese Journal of Computers, 2011,34(6):1041-1061.

[5] 肖芳雄, 黄志球, 曹子宁. Web 服务组合功能与 QoS 的形式化统一建模和分析[J]. 软件学报, 2011, 22(11): 2698-2715.
XIAO F X, HUANG Z Q, CAO Z N. Unified formal modeling and analyzing both functionality and QoS of Web services composition[J]. Journal of Software, 2011, 22(11): 2698-2715.

[6] 陈小峰. 可信平台模块的形式化分析和测试[J]. 计算机学报, 2009, 32(4): 646-653.
CHEN X F. The formal analysis and testing of trusted platform module[J]. Chinese Journal of Computers, 2009, 32(4): 646-653.

[7] 何建波, 卿斯汉, 王超. 对一类多级安全模型安全性的形式化分析[J]. 计算机学报, 2006, 29(8): 1468-1479.
HE J B, QING S H, WANG C. Formal safety analysis of a class of multilevel security models[J]. Chinese Journal of Computers, 2006, 29(8): 1468-1479.

[8] 钱振江, 黄皓, 宋方敏. 操作系统形式化设计与安全需求的一致性验证研究[J]. 计算机学报, 2014, 37(5): 1082-1099.
QIAN Z J, HUANG H, SONG F M. Research on consistency verification of formal design and security requirements for operating system[J]. Chinese Journal of Computers, 2014, 37(5):1082-1099.

[9] 杨涛, 王永刚, 唐礼勇. 一种实用动态完整性保护模型的形式化分析[J]. 计算机研究与发展, 2013, 50(10): 2082-2091.
YANG T, WANG Y G, TANG L Y. A practical dynamic integrity protection model[J]. Journal of Computer Research and Development, 2013, 50(10): 2082-2091.

[10] BELL D E, LA PADULA L J. Secure computer system: Unified exposition and multics interpretation[R]. MITRE CORP BEDFORD MA, 1976.

[11] LUNT T F, DENNING D E, SCHELL R R, *et al*. The sea view security model[J]. Software Engineering, IEEE Transactions, 1990,

16(6): 593-607.
[12] 张敏, 徐震, 冯登国. 数据库安全[M].北京:科学出版社, 2005.
ZHANG M, XU Z, FENG D G. Database Security[M]. Beijing: Science Press, 2005.
[13] 李丽萍, 卿斯汉,周洲仪. 安全策略模型规范及其形式分析技术研究[J]. 通信学报, 2006, 27(6): 94-101.
LI L P, QING S H, ZHOU Z Y. Research on formal security policy model specification and its formal analysis[J]. Journal on communications, 2006, 27(6): 94-101.
[14] HONG Z, YI Z, L C Y, *et al*. Formal specification and verification of an extended security policy model for database systems[A]. Trusted Infrastructure Technologies Conference[C]. 2008. 132-141.
[15] SANDHU R S, COYNE E J, FEINSTEIN H L, *et al*. Role-based access control models[J]. Computer, 1996, 29(2): 38-47.
[16] HE Y Z, HAN Z, FU H, *et al*. The formal model of DBMS enforcing multiple security polices[J]. Journal of Software, 2010, 5(5): 514-521.

作者简介：

王榕（1986-），女，辽宁大连人，中国科学院软件研究所博士生，主要研究方向为数据安全、隐私保护。

张敏（1975-），女，安徽萧县人，中国科学院软件研究所副研究员，主要研究方向为数据库安全与隐私保护。

冯登国（1965-），男，陕西靖边人，中国科学院软件研究所研究员，主要研究方向为密码学与信息安全。

李昊（1983-），男，河南固始人，中国科学院软件研究所副研究员，主要研究方向为网络与系统安全。