摘 要

在众多安全数据库的数据模型中，MLR 模型是非常成功的。MLR 模型成功地引入了“数据借用”的概念，解决了多实例等问题，但也带来了安全隐患。在该模型的数据借用操作中，低安全级的用户可以修改高安全级用户的视图。本文针对这个问题，提出了一种改进的安全模型。该模型扩展了 Smith-Winslett 模型，将修改后的“数据借用”操作增加到模型中。同时，该模型重新定义了多实例完整性和参照完整性，消除了多实例带来的语义模糊性、查询模糊性和更新引起的元组数据激增等问题。然后本文研究了推理控制问题。介绍了检测推理通道的方法，在此基础上描述了一种两阶段推理控制策略。这种策略包含静态提升安全级和动态推理控制两部分。静态提升安全级的算法能够保证在信息易用性损耗较小的情况下控制推理通道。在数据库运行阶段，利用动态推理控制算法可以在基本不影响访问速度的情况下切断剩余的推理通道。同时，本文还研究了隐通道控制问题。介绍了隐通道的分类、标识和处理措施。结合信息流法和隐蔽流树法的优点，提出了隐蔽流图法并给出了搜索步骤。根据上述的诸多理论，设计并开发了一个多级安全关系数据库管理系统的原型系统。实验结果表明文中提出的理论和各种算法是正确的。

关键词：安全模型，多级安全，隐通道，推理通道，动态推理控制

ABSTRACT

In many data models of secure database, the MLR model is very successful. It introduced the concept of data-borrow successfully and solved the polyinstantiation problem. However, it also brought a potential security problem. In the data-borrow operation of this model, a low-level user can modify the view of high-level user. This paper presents an improved security model to solve the problem. The model extends the Smith-Winslett model and introduces the amendatory data-borrow operation. At the same time, the model redefines polyinstantiation integrity and referential integrity. It eliminates the ambiguous semantics, fuzzy query and proliferation of tuples due to updates. Then this paper studies the problem of inference control. It presents methods of detecting the inference channels. Also, it describes a two-phase inference control strategy. This strategy includes static upgrade of the seurity level and dynamic inference control. Static upgrade of the security level algorithm can guarantee controlling inference channels with less information loss. When the database is running, dynamic inference control algorithm can cut off the inference channels well without efficiency decline. At the same time, the paper studies covert channel control. It introduces the taxonomy, identifications and resolvents of covert channel briefly. With the advantages of the information flow method and covert flow tree method, this paper proposes a covert flow diagram method and gives steps of searching the diagram. According to the various theories, a multilevel secure DBMS is designed and developed. The result of our experiment shows that the theory and algorithms are correct.

Key Words: Security Model, Multilevel Security, Covert Channel, Inference Channel, Dynamic Inference Control

图清单

表清单

注释表

MLS/DBMS:	Multilevel Secure Database Management System（**多级安全数据库管理系统**）
DAC:	Discretionary Access Control（**自主访问控制**）
MAC:	Mandatory Access Control（**强制访问控制**）
TCSEC:	Trusted Computer System Evaluation Criteria（**可信计算机系统评估标准**）
TDI:	Trusted Database Management System Interpretation（**可信数据库管理系统解释**）
ESW:	Extended Smith-Winslett（**改进的** Smith-Winslett **模型**）
FK:	Foreign Key（**外键**）
BLP:	Bell Lapadula（**贝尔－拉帕丢拉**）
SeaView:	Secure Data View（**安全数据视图**）
SRM:	Shared Resource Matrix（**共享资源矩阵**）
CFT:	Covert Flow Trees（**隐蔽流树**）
FD:	Functional Dependency（**函数依赖**）
RBAC:	Role-based Access Control（**基于角色的访问控制**）

承诺书

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

本人授权南京航空航天大学可以有权保留送交论文的复印件，允许论文被查阅和借阅，可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文。

作者签名：_____

日　　期：_____

第一章 绪论

1.1 引言

随着因特网的飞速发展和计算机技术的普及，数据库技术已经在社会的各个方面，包括经济、政治、军事、科技等领域，得到了广泛应用。目前，大多数公司和组织都把数据存储在数据库中，通过计算机网络进行传输和交换。在享受数据库技术带来的便利的同时，也承担了不小的风险。数据库中的机密信息、个人数据和商业情报，一旦被非法窃取或删改，将产生难以估量的损失。据统计，西方发达国家的政府机构和企业，每年通过计算机被窃取的资金高达数十亿元。因此，保护数据库中的数据已刻不容缓。研究发现，对数据库的威胁主要包括：对数据库内数据（包括敏感、非敏感的信息）的非授权访问；使授权用户不能正常得到数据库的数据服务。如何处理这些威胁已经成为信息安全非常重要的研究课题[1,2]，具有重大的理论意义和现实意义。

当前，商用的数据库系统大多采用自主访问控制或基于角色的访问控制来限制用户的非授权访问[3]。这两种访问控制策略的最大缺陷是不能有效抵御特洛伊木马攻击。为了解决这个问题，引入了强制访问控制机制。在这个机制中，高安全级的用户不允许对低安全级的对象进行写访问，因此木马的服务器程序无法将信息发送出去，从而阻止了敏感信息的泄露[4]。

实现了强制访问控制的数据库管理系统一般称为多级安全数据库管理系统（Multilevel Secure Database Management System，MLS/DBMS）[5]。在 MLS/DBMS 中，主体和客体均被赋予一个安全级别。安全级别由密级和类别集合组成。密级是有序的，类别集合是彼此独立、无序的。有时，类别集合不出现。为确定用户是否可以访问某个数据项，用户的安全级需要与数据的安全级进行比较[2]。目前流行的 MLS/DBMS 大多采用 BLP 模型作为其安全模型。在 BLP 模型中，遵循“向下读，向上写”的安全策略。这样虽然可以有效防止非授权用户直接获取敏感信息，但并不能有效阻止用户利用可获取的低安全级信息进行推理、利用隐通道来获取高安全级的信息[6]。推理控制和隐通道控

制对于保证数据的安全有着非常重要的意义。本文围绕安全数据模型、推理控制和隐通道控制等主题进行深入研究，为开发商用高安全级 MLS/DBMS 提供参考。

1.2 安全数据库的主要研究内容

尽管对象—关系数据库、面向对象数据库的研究如火如荼，多级安全数据库系统的研究目前仍主要基于传统的关系数据库系统（RDBS）[7]。这一方面是因为数据库安全技术的发展滞后于数据库技术的发展，更主要的原因是 RDBS 在理论上和技术上都已经比较成熟，并得到了广泛应用。特别是其非过程化的查询语言 SQL，易于扩充并保持其安全性。安全数据库的研究内容主要包括：多实例问题、多级安全数据模型、完整性问题、推理问题、隐通道的分析与处理、事务处理机制、特定环境下的安全研究等。

1. 多实例问题

多实例首先是在 SeaView 项目[8]中提出的，指的是在 MLS/DBMS 中同时存在具有相同名字的多个数据客体（元组、元素），它们的区别在于安全级别。在传统数据库系统中，实体完整性[9]要求主关键字唯一标识一个关系中的每一个元组，并且不能为空。这样的定义在一个典型的单级关系中易于理解，但不能简单扩展到多级关系中，因为主关键字的唯一性要求会产生信号通道，导致信息从高安全级流向低安全级。多实例的引入解决了保密性与完整性冲突的问题，然而同时也带来了多级关系的语义模糊性、查询模糊性和更新引起的元组数目激增等问题。为了控制多实例，各种安全模型都重新定义了实体完整性、参照完整性等规则，并引入了多实例完整性规则。然而，不同模型对于多实例元组或元素语义的理解还存在较大的分歧。

2. 多级安全数据模型

安全模型也称为策略表达模型，是一种不依赖于软件实现的、高层次上的概念模型。安全模型大致可以分为两类：自主型安全模型和强制型安全模型。自主访问控制通过明确授权来决定用户访问某数据对象，其典型代表是 Harrison、Ruzzo 和 Ullman 在 1976 年提出的一种基于访问控制矩阵的模型。强制访问控制通过无法回避的存取限制防止各种直接和间接的攻击，它的典型代表是 Bell LaPadula 模型[10]（简称 BLP 模型）。为保护数据库中信息的安全，

人们在 BLP 模型的基础上提出了各种安全模型，如：SeaView 模型，LDV 模型[11]，Jajodia-Sandhu 模型[12]，MLR 模型[13]等。

3. 完整性问题

完整性是指能够保障被传输、接收或存储的数据是完整的和未被篡改的。传统数据库的完整性一般包括实体完整性、参照完整性和用户自定义的完整性[14]。数据库的完整性约束条件与安全性需求往往是相冲突的。多实例的引入就是为了解决实体完整性与安全性冲突的。如何在数据库的完整性约束与安全性需求之间进行折衷，是一个值得研究的课题。

4. 推理问题

由于关系数据库中的数据之间存在各种内在关联，数据库用户有可能根据他所允许知道的信息，利用数据之间的内在逻辑联系，推断某些限制其访问的内容。这类问题称为数据库中的推理分析问题。推理通道与下面将提到的隐通道是不同的。推理通道只要有低安全级用户参与即可，因此推理通道是单方面的；而隐通道需要两个不同安全级的主体共同协作完成信息的传送。

5. 隐通道的分析与处理

目前被广泛使用的隐通道的定义是：一个非自主的安全模型 M，以及在一个系统中该模型的解释 I(M)，在任何两个主体 I(Sh)和 I(Sl)之间的信道是隐通道，当且仅当在模型 M 中，对应主体 Sh 和 Sl 之间的任何通信都是非法的[6]。隐通道存在的原因是系统中存在共享资源，这些资源可以被系统原语查看和修改。因此，隐通道分析的关键是搜索系统中的共享资源，以及查看和修改这些资源的系统原语。

6. 事务处理机制

事务处理是数据库系统的重要技术之一。安全数据库的事务处理包含了安全并发控制和多级事务的处理。如果两个不同安全级的事务希望访问同一个数据库元素，那么它们之间就有可能产生定时隐通道。这就需要安全并发控制技术来解决这个问题。多级事务处理是为了解决在一个事务内访问多个安全级上的数据库对象，又能保持事务的原子性问题。文献[15]中证明了要同时保持原子性和安全性是不可能的，因此必须在两者之间进行折衷。

7. 特定环境下的安全研究

近年来，安全数据库在各种特定环境中的研究已成为一个热门方向。空间安全数据库、移动环境下的安全数据库、XML 安全数据库等都是值得深入研

究的领域。

1.3 国内外研究现状

国外对计算机安全及数据库安全的研究起步较早，包括政府部门在内的各种研究机构及公司对此投入了大量的研究，研究成果和应用都较为丰富。

1967 年 10 月，在美国国防科学技术委员会的赞助下成立了特别工作组，致力于计算机安全保护、远程访问保护和远程共享计算机系统的保密信息。60 年代末、70 年代初期开始出现一系列有关计算机安全的论文。D. E. Bell 和 LaPadula 于 1973 年提出著名的 BLP 模型，这是最早的一个完全的、形式化的多级安全模型。此后又出现了格模型、无干扰模型、SeaView 模型等一系列的安全模型。这些模型的研究以及原型系统的开发大大地推进了数据库安全研究的理论和实践。

1985 年，美国国防部正式颁布了《DoD 可信计算机系统评估标准》[16]（Trusted Computer System Evaluation Criteria，简称 TCSEC 或 DoD85）。1991 年又颁布了 TDI[17]（Trusted Database Management System Interpretation），将 TCSEC 扩展到数据库管理系统。在 DoD85 中，按系统可靠或可信程度逐渐增高将系统划分为四组七个等级[16-17]：D；C（C1，C2）；B（B1，B2，B3）；A（A1），如表 1.1 所示。

在安全产品研究方面，欧美主流数据库公司都推出了通过一定等级评估的安全产品。1993 年，Oracle 公司发布了安全数据库 Trusted Oracle 7TM，满足 TCSEC 的 B1 级要求。目前 Oracle 公司的安全数据库产品 Oracle 9i 已经通过十四个独立的安全评估，达到美国可信计算机安全评价标准（TCSEC）的 A1 级并成功通过英国 ITSEC 和新的 ISO 标准。MS SQL Server 是购买 Sybase 公司 1987 年推出的 Sybase SQL Server。Microsoft SQL Server 2005 作为一个可升级的、可靠的并且易于使用的产品为企业用户所青睐，其在安全方面也有了很大的发展。Secure SQL Server 是 Sybase 公司的安全数据库，其安全产品通过 TCSEC 的 B1 级评估，也是美国军方使用的产品，曾在海湾战争中使用。目前最新产品 Sybase ASE 12.5.2 已经获得 CC 的 EAL4 级的安全认证，为通用软件达到的最高安全级别。

国内在这方面的研究和开发起步较晚，与技术先进国家有较大差距。我国

于 1999 年颁布了《计算机信息系统安全保护等级划分准则》[18]，在 2002 年又由公安部颁布了 GA/T-389-2002《计算机信息系统安全等级保护数据库管理系统技术要求》[19]。在产品方面，由北京大学牵头，人大、中软和华中合作研发的 COBASE 数据库管理系统，其 COBASE V2.0 的可信版本基本实现了 TCSEC B1 级的安全功能要求；华中理工大学（现华中科技大学）研制的 DM 系列数据库管理系统，其中 DM3 经公安部计算机信息系统安全产品质量监督检测中心评测达到国标 17859 的第三级（基本相当于 TCSEC B1 级）。国内的大部分数据库系统仍停留在原型系统上，真正商用的很少，而且也没有 B2 级以上标准的产品。

表 1.1 TCSEC/TDI 安全级别划分

安全级别	定　　义
A1	验证设计（Verified Design）
B3	安全域（Security Domains）
B2	结构化保护（Structural Protection）
B1	标记安全保护（Labeled Security Protection）
C2	受控的存取保护（Controlled Access Protection）
C1	自主安全保护（Discretionary Security Protection）
D	最小保护（Minimal Protection）

本文主要研究安全数据模型、推理控制和隐通道控制，因此下面分别介绍这三个方向国内外的研究进展。

1.3.1 安全数据模型

数据库技术的研究是从数据模型开始的。目前已有的数据模型根据安全策略可以分为两类：自主型数据模型和强制型数据模型。自主型数据模型遵守自主访问安全策略，使用访问控制矩阵来实现对数据的访问控制。这类数据模型容易因恶意代码等因素造成信息泄露，不能满足军队、国家等一些对信息保密要求较高的单位的应用需求。强制型数据模型遵守强制访问安全策略，用户对数据的访问控制基于用户和数据的安全级。强制型数据模型能够防止各种直接和间接的攻击。从历史的发展来看，主要有以下几种著名的数据模型：

1. 贝尔-拉帕丢拉模型

1973 年，Bell 和 LaPadula 提出了贝尔-拉帕丢拉模型（简称 BLP 模型），该模型的目的是在操作系统环境下提供对于信息的安全保护。BLP 模型也是

最著名的多级安全策略模型，其定义中包含了两条重要的安全规则：

（1）简单安全规则（ss 规则）：仅当一个主体的安全级别支配一个客体的安全级别时，主体才具有对客体“读”或“写”的存取权限。

（2）星（*）规则：如一个不可信主体具有对一客体添加权限，那么客体的安全级一定支配主体的当前安全级；如一个不可信主体具有对一客体的写权限，那么主体的当前安全级一定与客体的安全级相等；如一个不可信主体具有对某一客体的读权限，那么一定有主体的当前安全级支配客体的安全级。

这两个规则可以总结为“不上读，不下写”。满足它即控制了系统中信息的流动，确保了系统的安全性。所有采取强制访问控制策略的安全模型都采用这两条规则。

2. SeaView 模型

1987 年，Denning 等人在斯坦福研究所（SRI）开发了一个保护关系数据库系统的安全模型，称之为安全数据视图模型（Secure Data View）。为处理多级关系，SeaView 模型扩充了关系的概念，对关系增加了密级标识。密级可以赋予关系中的单个元素，即每个特定的属性值。

SeaView 模型定义了实体完整性、参照完整性和多实例完整性规则。其中，实体完整性要求所有主键属性的安全级相同，所有主键属性值不为空，其它属性的安全级支配主键属性的安全级；参照完整性要求所有外键属性的安全级相同，并且必须支配参照元组中主键属性的安全级；多实例完整性要求主键和主键属性的安全级与其余属性和这些属性的安全级之间满足多值依赖，并以此来控制多实例。但在对关系进行插入或更新操作时，SeaView 模型会产生大量的不合逻辑的多实例元组和多实例元素。

3. Jajodia-Sandhu 模型

为消除多值依赖，Jajodia 和 Sandhu 提出了另一个多级安全数据模型。这个模型定义了空值完整性、实例间完整性，并且重新定义了多实例完整性（规定用户指定的主键 K、K 的安全级属性 C_k 和属性 A_i 的安全级 C_i 函数决定 A_i 的属性值）。这个模型最大的问题是空值的二义性。

4. Smith-Winslett 模型

Smith 和 Winslett 提出一种基于信赖语义多级安全数据模型。在该模型中，每个安全级上的数据反映了那个安全级上的用户对现实世界的“信赖”。它是一种半元组级标记数据模型。Smith-Winslett 模型避免了语义模糊、查询模糊

和关系更新时多实例元组激增等问题。

5. MLR 模型

Sandhu 等人总结了上述几个模型的特点，提出了一种基于数据语义的多级关系（MultiLevel Relational，MLR）数据模型。该模型提出了数据借用的思想，即高安全级的用户可以向低安全级的用户借用数据。MLR 模型定义了数据借用完整性，并修改了数据操纵语句的语义。MLR 模型的缺点是允许低安全级用户修改高安全级用户的视图，因而在系统的实现过程中有可能产生隐患。

1.3.2 推理控制

推理问题在其他系统中也存在，但在数据库系统中更加突出。早在 1980 年，科研工作者们就已经开始对安全数据库的推理控制问题进行研究，其工作主要集中在问题的定义及框架的建立上。Jajodia 等证实了数据库自身约束是造成多级关系数据库中的大多数推理通道的原因[20]。为确保数据的安全性，检测和消除推理通道是必要的。下面介绍当前关于推理控制的研究工作。

1. 基于函数依赖和多值依赖的推理

Su 和 Ozsoyoglu 研究了在一个分别具有属性分级模式和记录分级模式的多级关系数据库中，由于函数依赖和多值依赖产生的推理问题[21]。对于函数依赖，他们给出了一个属性级别调整算法；对于多值依赖，他们提出了一个关系中元组级别调整算法，从而消除了由于函数依赖和多值依赖产生的危险。

2. 基于数据库约束和非敏感数据结合的推理

Brodsky 等人研究了当数据库约束与非敏感数据结合获得敏感信息时产生的推理通道问题[22]。他们提出了一个集成的安全机制，称为泄露监控器。泄露监控器使用一个泄露推理引擎扩展了标准的强制访问机制，确保数据的保密性。

3. 数据级推理检测系统

Yip 和 Levitt 注意到通过分析数据库中存储的数据，用户可以检测出更多的推理[23]。他们提出了六条推理规则，即包含、唯一特征、重叠、补充、函数依赖和分解查询。这些推理规则的存在表明仅使用函数依赖检测推理是不够的。

1.3.3 隐通道控制

早在 1973 年，Lampson 就认识到隐通道是对系统安全的一个潜在的威胁[6]。隐通道分析的关键步骤就是找出系统设计和实现中存在的隐通道，即隐通道的搜索方法。国外一些专家和学者已经提出的影响较大的方法有：信息流分析法、无干扰分析法、共享资源矩阵法、隐蔽流树法、基于源代码的搜索方法等。

找到隐通道后下一个步骤一般是评估它们的带宽。目前，计算带宽的方法主要是 Millen 提出的形式化方法和 Tsai 和 Gligor 提出的非形式化方法，他们分别使用于不同的场合。评估完带宽后，就可以根据安全策略来进行适当的处理。一般采用的方法是：消除隐通道、缩小带宽或对隐通道的使用进行审计。

1.4 本文的组织

本文主要研究多级安全数据库的数据模型、推理控制和隐通道控制等三方面内容，全文由六章组成，组织如下：

第一章：介绍课题的研究背景、多级安全数据库的研究内容和国内外研究现状。

第二章：一种改进的安全模型。分析了 MLR 模型的缺点，在 Smith-Winslett 模型的基础上提出了一种改进的数据模型，并修改了部分操作的语义，使之更加安全。同时，对几个安全模型的表达能力进行了比较，讨论了数据操纵处理规则，并且证明了模型的正确性、完备性和安全性。

第三章：推理控制。分析了推理通道的来源及常见的几种解决方法。提出了一种解决函数依赖的安全级提升方法，并能使数据可用性损失最小。同时，对于可用性损失过大的推理通道，提出了一种在数据库运行期间进行动态推理控制的方法，简单易用。在这两种方法的综合作用下，较好地解决了基于元数据的推理问题。

第四章：隐通道控制。主要介绍了隐通道的分类、标识和处理策略。重点介绍了共享资源矩阵法和其他几种常见的隐通道表示方法。对隐蔽流树法进行改进，提出隐蔽流图法，并且给出了在隐蔽流图中检索隐通道的策略。

第五章：原型系统的设计与实现。介绍了原型系统采用的体系结构和部分

实现过程，研究分析原型系统中三个安全模块的功能和数据结构。

第六章：总结全文，并对未来的工作进行展望。

第二章 一种改进的安全模型

确定多级安全模型的目的是提供一个不依赖于软件实现的、高层次上的多级模型，用以精确定义系统的多级安全策略。到目前为止，在大量的文献中已经提出了许多用以实现多级安全数据库系统的安全模型[13,26-31]，不同的模型有不同的特点。本章借鉴了 MLR 模型[13]数据借用的思想对 Smith-Winslett 模型[31]进行了扩展，同时对 Smith-Winslett 模型的操纵语义进行了部分改进，提出了 ESW（Extended Smith-Winslett）模型。

2.1 引言

安全模型也称为策略表达模型，是一种高层抽象、独立于软件实现的概念模型[6]。在包括数据库系统在内的各种安全系统中，安全模型是用于精确地描述该系统的安全需求和安全策略的有效方式。目前比较成熟的安全模型大部分是访问控制策略模型，用于描述与维护系统中数据的保密性和完整性。

根据访问控制策略类型的差异，目前主流的安全策略模型大致可以分为自主访问控制和强制访问控制两大类。自主访问控制策略是一种通用访问控制策略。在该策略下决定用户能否访问某数据对象的依据是系统中是否存在明确授权。授权一般有两种方法：一种是基于封闭世界的理论，用户要访问某个对象必须拥有该对象上的授权，否则拒绝。这种方式的缺点是如果一个集合中只有几个人不允许访问某对象，那么授权将有许多无谓的重复；另外一种是基于开放世界的理论。在这个理论下，用户可以访问任一对象，除非明确禁止用户访问该对象。这就很好地解决了封闭世界理论所遇到的问题。通常情况下是将两者结合起来使用。用户的授权可以传递、转让，也可以回收。自主访问控制模型普遍难以解决的一个问题是如何有效抵御特洛伊木马攻击。木马在对象的拥有者不知情的情况下，可以直接冒充属主给攻击者授权，或将信息表示成攻击者可以访问的某种方式，从而绕过访问控制机制，达到窃取或篡改、破坏的目的。强制访问控制通过用户无法回避的存取限制来防止这类攻击。与自主访问控制不同，强制访问控制策略把所有的权限都归于系统集中管理，保证信息的

流动始终处于系统的控制之下。这种策略给每个主体和客体分配一个安全级别，系统通过比较主体和客体的安全属性是否匹配来决定让主体访问指定的客体或者拒绝。典型的强制访问控制模型包括 BLP 模型，Dion 模型等。

对于安全性要求不高的系统，实现自主访问控制就能满足其安全性需要。但在国防、军事等应用中，由于对系统安全性要求很高，必须实现强制访问控制[24]。一个系统中如果只有强制访问控制，那么很多资源将无法正常使用。因此，安全性较高的系统大多采用强制访问控制为主、自主访问控制为辅的访问控制策略。

本章把强制访问控制作为重点。在所有强制访问控制模型中，MLR 模型是较为成功的安全模型，它集合了其它几个安全模型的优点，解决了其缺点；它首次提出了“借用”的概念。但是，由于它分级粒度很细，针对每个元素都分配了安全级，因此并没有在实际的产品中得到应用。同时，由于借用的语义使得低级用户的操作可以改变高级用户的视图，这就有潜在的安全问题；而 Smith-Winslett 模型中没有借用的概念，高级用户不能同时借用几个低级用户的数据。针对这些问题，本章对 Smith-Winslett 模型进行了扩展，使它既保持了原有的优点，又添加了改进的借用操作并修改了部分操作的语义。同时，ESW（Extended Smith-Winslett）模型继承并发展了多级实体的语义。一个多级实体可以跨越多个安全级别，并且能够模拟对属性值的多个安全理解。

2.2 ESW 模型

本节对 ESW 模型进行形式化定义，并对该新模型进行数据语义解释。然后在 2.3 和 2.4 节分别描述 ESW 模型的完整性性质和数据管理操作。

2.2.1 模型定义

一个多级关系是由下面两部分组成的：

定义 2.1 一个多级关系模式被表示为 $R(A_1, C_1, A_2, A_3, .., A_n, TC)$，其中 A_i 是定义在域 D_i 上的数据属性，C_1 是对应于 A_1 的分级属性，TC 是元组的分级属性。C_1 和 TC 的域由安全级格{L，H}指定，其中 L 表示系统最低安全级，H 表示系统最高安全级。

定义 2.2 一个关系实例被表示为 $r(A_1, C_1, A_2, A_3, .., A_n, TC)$，它

是一个形式为（a_1，c_1，a_2，a_3，..，a_n，tc）的不同元组的集合，其中 $a_i \in D_i$，c_1，$tc \in \{L, H\}$，$tc \geq c_1$。

上述的两个定义中，A_1 表示用户指定的外观主键，通常 A_1 由一个或多个属性组成，这些属性具有相同的安全级。每个关系在任何时刻都只能有一个实例，不同级别的用户可以看到这个实例不同的视图。为了描述方便，将访问安全级是 c 的主体称之为 c-主体，将元组级别是 c 的元组称之为 c-元组。下面的表就是一个多级关系实例。

表 2.1 一个多级关系实例

姓名	主键安全级	工资	部门	元组级别
王强	S	6000	信息	S
王强	U	2000	技术	U

定义 2.3 一个数据库是一个关系的集合。一个数据库状态是在某个特定的时刻，数据库中所有关系实例的一个集合[25]。

2.2.2 模型数据语义的解释

ESW 模型的思想与 MLR 模型的思想相似。对于所有的实例 r（A_1，C_1，A_2，A_3，..，A_n，TC）和所有的元组 $t \in r$，数据解释如下：

（1）A_1，C_1 可以唯一确定一个关系中的实体；C_1 表示这个实体的安全级别。

（2）$t[C_1]=c_1$ 表明这个实体由一个 c_1-主体创建，因而被称作是一个 c_1-实体。

（3）$t[TC]=tc$ 且 $t[C_1]=c_1$ 表示 t 是由一个 tc-主体加入的，t 中所有的数据都被 tc-主体认可。如果 t 不存在，则说明 c_1-实体不被 tc-主体认可；t 只能被级别 $c' \geq tc$ 的主体看到。t 只能被 tc 主体删除。

（4）如果 $t[TC]=t[C_1]$，那么 t 是实体的基本元组。所有元组 $t' \in r$ 满足 $t'[A_1,C_1]=t[A_1,C_1]$，同时 $t'[TC]>t[TC]$都是基于 t 的借用元组。

由上面的形式化描述可以得到如下有用信息：

（1）主体只可以更新（包括插入、删除、修改）同级别的数据，某一级别的数据只可以被同级别的主体更新。这条规则对应 BLP 模型中的“不下写”原则，但因其限定在同级别，所以控制比 BLP 模型更强。这也避免了 MLR 模

型中低级别主体可以修改高安全级主体的视图这一问题。

（2）一个 c-主体能够看到访问级被 c 支配的主体认可的数据。这条规则对应于 BLP 模型中的“不上读”原则。这里，一个 c-主体认可的数据包括两部分内容：c-主体自己创建的数据（通过插入操作）；c-主体通过借用操作得到的数据，它一般是由 c-主体向低级用户借用的数据和自己指定的数据两部分组成。

（3）存在两种类型的多实例：实体多实例和元组多实例。当一个关系包含多个元组具有相同的 A_1 值，如果这些元组的 C_1 值不同，则称其为实体多实例；如果这些元组的 C_1 值相同但 TC 值不同，则称其为元组多实例。

2.3 完整性性质

ESW 数据模型包含了 4 个完整性性质，其中实体完整性、外键完整性和参照完整性是传统关系数据库都必须满足的性质。而多实例完整性是多级安全数据模型所特有的。

2.3.1 实体完整性

ESW 的实体完整性是取自于原始的 SeaView 模型，也就是一个多级关系 R 的实例 r 满足实体完整性，当且仅当：对于所有的 $t \in r$，

（1）$t[A_1] \neq null$；

（2）$t[TC] \geq t[C_1]$。

第一个要求是指外观主键不能为空。第二个要求说明在任何元组中，元组的安全级别必须支配主键属性的安全级别。

2.3.2 外键完整性

设 FK 是参照关系 R 的外键。一个多级关系 R 的实例 r 满足外键完整性，当且仅当：对于所有的 $t \in r$，

（1）$\forall A_i \in FK, t[A_i] = null$； 或者

（2）$\forall A_i \in FK, t[A_i] \neq null$。

外键完整性要求在任何元组中，所有外键的属性值要么都为空要么都不为空。

2.3.3 参照完整性

设 FK_1 是参照关系 R_1 的外键，A_{11} 是 R_1 的外观主键，A_{21} 是被参照关系的外观主键。R_1 的实例 r_1 和 R_2 的实例 r_2 满足参照完整性，当且仅当：

对于所有的 $t_1 \in r_1$，如果 $t_1[FK_1] \neq null$，那么一定存在 $t_2 \in r_2$ 满足 $t_1[FK_1]=t_2[A_{21}] \wedge t_1[TC]=t_2[TC] \wedge t_1[C_{FK1}] \geq t_2[C_{21}]$，其中 C_{21} 是对应于 A_{21} 的安全级属性。

在上面的定义中，如果 $FK_1 \subseteq A_{11}$，那么 $t_1[C_{FK1}]=t_1[C_{11}]$，其中 C_{11} 是对应于 A_{11} 的安全级属性；否则 $t_1[C_{FK1}]=t_1[TC]$。参照完整性中要求 $t_1[C_{FK1}] \geq t_2[C_{21}]$是为了只允许向下参照；而 $t_1[TC]=t_2[TC]$说明对于任何级别 c，c-元组只能参照 c-元组。这是由于 c-元组包含了 c-主体认可的所有数据，一个 c-元组的缺少意味着 c-主体不认可这个实体的存在。这样，ESW 模型消除了参照模糊性。

2.3.4 多实例完整性

一个多级关系 R 的实例 r 满足多实例完整性，当且仅当：

（1）$A_1, TC \rightarrow C_1$ 并且

（2）对于所有 $1 \leq i \leq n$，$A_1, TC \rightarrow A_i$。

性质 1 是实体多实例完整性。直观解释是，在一个关系中可以有多个实体具有相同的主键值，但在任何安全级主体能够认可至多一个实体具有那个外观主键。例如，表 2.2 的多级关系是不允许的，因为在级别 S 存在两个实体具有相同的主键值“小李”，（小李，U）和（小李，S）。为了避免语义模糊性，S 可以认可两个中的任何一个，但不能同时认可两个。

表 2.2 实体多实例完整性冲突

姓名	主键安全级	部门	工资	元组级别
小张	U	部门 1	1000	U
小李	U	部门 1	1000	U
小李	U	部门 2	4000	C
小李	U	部门 2	1000	S
小李	S	部门 2	2000	S

在 ESW 模型中，跨越安全级的实体多实例是允许的，因此不存在插入元组时由于在较高级别存在实体多实例而被拒绝，这就避免了向下的信息泄露。

此外，在特定的安全级，主体认可的元组中不允许存在实体多实例，这就消除了模糊性。

2.4 模型的操作

ESW 数据模型一共包含五个数据操纵语句。其中四个是传统数据库的操作语句：INSERT，DELETE，SELECT 和 UPDATE，另外一个是 MLR 模型的 UPLEVEL 语句。UPLEVEL 语句主要是用来解决 Smith-Winslett 模型的操作不方便问题的。针对每个数据操纵语句，本节给出形式化的语法和语义。

2.4.1 INSERT 语句

1. 语法

一个 c-主体执行的 INSERT 语句具有如下的一般形式：

INSERT

INTO R[(A_{j1}[, A_{j2}] ..)]

VALUES (a_{j1}[, a_{j2}] ..);

其中 R 代表一个关系；A_{j1}，A_{j2}，..是属性名，a_{j1}，a_{j2}，..是属性 A_{j1}，A_{j2}，…上的数值。在本文中，“[]”代表可选，“…”代表重复。属性上的值必须在它的域范围内，如 $a_{j1}\in D_{j1}$，$a_{j2}\in D_{j2}$ 等。

2. 语义

每个 INSERT 语句至多能够向关系 R 中插入一个元组。被插入的元组 t 构造如下：对于 $1 \le i \le n$，

（1）如果 A_i 在 INTO 子句的属性列表中，则 $t[A_i]=a_i$；

（2）如果 A_i 不在 INTO 子句的属性列表中，则 $t[A_i]$=null，同时 $t[C_1]=t[TC]=c$。

插入操作被允许，当且仅当：

（1）不存在 $t'\in r$ 满足 $t'[A_1]=t[A_1] \wedge t'[TC]=c$；并且

（2）产生的数据库状态满足实体完整性、参照完整性和外键完整性。

如果上述两个条件能够满足，那么元组 t 被插入到 r 中；否则，操作被拒绝，原始的数据库状态保持不变。

2.4.2 DELETE 语句

1. **语法**

一个 c-**主体执行的** DELETE **语句具有如下的一般形式：**

DELETE

FROM R

[WHERE p];

其中 p **是一个谓词表达式，除了通常情况下的数据属性外还可以包含安全级属性。其余符号与上节相似。**

2. **语义**

只有 t∈r **且** t[TC]=c **的元组才会在** p **的计算中被考虑。换句话说，** p **在执行前被修改为** p∧t[TC]=c**。对于满足条件的元组** t∈r **将被删除。如果在级别** c **产生的数据库状态满足参照完整性，也就是说，元组级别等于** c **的所有元组都满足参照完整性，那么** DELETE **语句执行成功；否则，删除被拒绝，并且原始的数据状态保持不变。**

2.4.3 SELECT 语句

1. **语法**

一个 c-**主体执行的** SELECT **语句具有如下的一般形式：**

SELECT B_1[, B_2] …

FROM R_1[, R_2] …

[WHERE p]

[BELIEVED BY c_1, c_2, ..];

其中 R_1，R_2，..**是关系名；** B_1，B_2，..**是** R_1，R_2，..**中的属性名，每个** B_i **是一个数据属性或分级属性或元组级别属性；** p **是一个谓词表达式，除了通常情况下的数据属性外，** p **还可以包含有关分级属性的条件。** c_1，c_2，..**是安全级的值（可以使用通配符*表示所有受** c **支配的级别），它们均小于** c**。**

2. **语义**

只有那些 t∈r_1，r_2，..**且** t[TC]**出现在** BELIEVED BY **子句中的元组将在** p **的计算中考虑（如果没有** BELIEVED BY **子句，则要求** t[TC]=c**）。如果** FROM **子句中包含多个关系，则谓词** p **被隐含地替换为** p∧（R_1.TC=R_2.TC= ..**）。对于**

满足条件 p 的元组，其对应于 SELECT 子句里提到的属性的数据将被返回。SELECT 语句总是执行成功的，尽管返回的元组集可能是一个空集。

3. 说明

一个关系中的 c-元组只能和其他关系中的 c-元组进行连接。因为一个 c-元组包含了 c-主体认可的所有数据，因此它只应当与其它的 c-元组连接，否则很难解释返回结果。允许不同安全级的元组进行连接不仅会带来连接模糊性问题，而且还可能导致隐通道的产生。由于在 ESW 模型中任何安全级上不允许存在元组多实例（如表 2.3），因此 SELECT 语句的连接可能性是唯一的。

表 2.3 c 安全级上元组多实例

姓名	主键安全级	工资	部门	元组级别
王强	C	6000	信息	C
王强	C	2000	技术	C

2.4.4 UPDATE 语句

1. 语法

一个 c-主体执行的 UPDATE 语句具有如下的一般形式：

UPDATE R

SET $A_{j1}=s_{j1}$[, $A_{j2}=s_{j2}$] …

[WHERE p];

其中各个符号的名称与上面的说明一致。

2. 语义

只有 t[TC]=c 并且 $t \in r$ 的元组才会在 p 的计算中被考虑。对于满足条件 p 的元组 t，r 将按如下方式被更新：

（1）如果 A_1 中没有属性在 SET 子句中，对于 $2 \le i \le n$，如果 A_i 在 SET 子句中，则 $t[A_i]=|s_i|$（| |代表计算后的结果）。

（2）如果 A_1 中某个属性在 SET 子句中，则

（a）如果 $t[C_1]=c$，对于 $1 \le i \le n$，如果 A_i 在 SET 子句中，$t[A_i]=|s_i|$；

（b）如果 $t[C_1]<c$，$t(A_1,C_1)=(|s_1|, c)$；对于 $2 \le i \le n$，如果 A_i 在 SET 子句中，则 $t[A_i]=|s_i|$。

如果产生的数据库状态满足 EI，FKI，PI 和 RI，那么 UPDATE 语句执行

成功；否则，更新语句被拒绝，并且原始的数据库状态保持不变。

2.4.5 UPLEVEL 语句

1. 语法

一个 c-主体执行的 UPLEVEL 语句具有如下的一般形式：

UPLEVEL R

GET A_{j1} FROM c_{j1}[, A_{j2} FROM c_{j2}] …

[WHERE p];

其中 R 是关系名；A_{j1}，A_{j2}，..是属性名，$2 \le i \le n$。这里假定关系 R 有 n 个属性，A_1 是外观主键。c_{j1}，c_{j2}，..是属性的安全级。p 是一个谓词表达式，除了通常情况下的数据属性外，p 还可以包含有关分级属性的条件。

2. 语义

只有 $t[TC] \le c$ 并且 $t \in r$ 的元组才会在 p 的计算中被考虑。对每一个实体，至少有一个元组 $t' \in r$ 满足条件 p，一个 c-元组将按下面的方式被构造：

（1）$t[A_1,C_1]= t'[A_1,C_1]$；

（2）for $2 \le i \le n$

（a）如果 A_i 在 GET 子句中

（i）如果存在一个元组 t''，$t''[A_1,C_1]=t[A_1,C_1] \wedge t''[TC]=c_i$，那么 $t[A_i]= t''[A_i]$；

（ii）如果不存在元组 t''，$t''[A_1,C_1]=t[A_1,C_1] \wedge t''[TC]=c_i$，那么 $t[A_i]=null$；

（b）如果 A_i 不在 GET 子句中，那么 $t[A_i]=null$；

此时，

（1）如果存在任何元组 t''，$t''[A_1,C_1]=t[A_1,C_1] \wedge t''[TC]=c$，用元组 t 代替元组 t''；然后

（2）如果不存在任何元组 t''，$t''[A_1,C_1]=t[A_1,C_1] \wedge t''[TC]=c$，把元组 t 加入到关系实例 r 中。如果加入元组后的数据库状态满足多实例完整性、外键完整性和参照完整性，则语句 UPLEVEL 操作成功；否则，借用语句被拒绝并且原始的数据库状态保持不变。

3. 说明

UPLEVEL 语句允许一个 c-主体从原来的 c-元组获得数据，这跟从低安全级借用数据是一样的。通过从 c-元组获得数据的新元组将替换旧的 c-元组。

2.5 操作语句示例

下面给出一组完整的例子来说明模型操作语句的使用。

多级关系 EMPLOYEE 如表 2.4 所示。

表 2.4 EMPLOYEE 多级关系表

姓名	主键安全级	部门	工资	元组级别
小张	U	部门 1	1000	U
小李	U	部门 1	1000	U
小李	U	部门 2	2000	C
小丁	S	部门 2	2000	S

（1）插入操作

一个 S-主体执行以下操作：

```
INSERT INTO EMPLOYEE
VALUES (‘小李’, ‘部门 2’, 3000);
```

EMPLOYEE 表变化如下：

表 2.5 插入操作后的 EMPLOYEE 多级关系表

姓名	主键安全级	部门	工资	元组级别
小张	U	部门 1	1000	U
小李	U	部门 1	1000	U
小李	U	部门 2	2000	C
小李	S	部门 2	3000	S
小丁	S	部门 2	2000	S

在这个表中小李（U）和小李（S）是两个不同的实体。

（2）删除操作

一个 U-主体对表 2.4 执行如下删除操作：

```
DELETE
FROM     EMPLOYEE
WHERE    姓名=‘小李’;
```

删除后，关系变化如表 2.6 所示

（3）查询操作

一个 c-主体对表 2.4 执行以下操作：

SELECT *
FROM EMPLOYEE
WHERE NAME=‘小李’
BELIEVED BY ANYONE;

此操作的返回结果为：

<小李（U），部门 1，1000 >
<小李（U），部门 2，2000 >

表 2.6 删除操作后的 EMPLOYEE 多级关系表

姓名	主键安全级	部门	工资	元组级别
小张	U	部门 1	1000	U
小李	U	部门 2	2000	C
小丁	S	部门 2	2000	S

（4）更新操作

一个 c-主体对表 2.4 执行如下操作：

UPDATE EMPLOYEE
SET 工资=4000
WHERE 工资=2000;

更新后，关系变化如表 2.7 所示：

表 2.7 更新操作后的 EMPLOYEE 多级关系表

姓名	主键安全级	部门	工资	元组级别
小张	U	部门 1	1000	U
小李	U	部门 1	1000	U
小李	U	部门 2	4000	C
小丁	S	部门 2	2000	S

（5）借用操作

一个 s-主体对表 2.7 执行如下操作：

UPLEVEL EMPLOYEE

GET 部门 FROM C, 工资 FROM U

WHERE 姓名=‘小李’;

操作完成后，关系变化如表 2.8 所示：

表 2.8 借用操作后的 EMPLOYEE 多级关系表

姓名	主键安全级	部门	工资	元组级别
小张	U	部门 1	1000	U
小李	U	部门 1	1000	U
小李	U	部门 2	4000	C
小李	U	部门 2	1000	S
小丁	S	部门 2	2000	S

2.6 模型比较

1.表达能力比较

ESW 模型是在 Smith-Winslett 模型上进行扩展的，仍然属于半元组级标记的模型。半元组级标记的数据模型都具有如下的模式：

A_1	C_1	A_2	A_3	…	A_n	TC

图 2.1 半元组级标记模型的模式表达形式

ESW 模型的表达能力如图 2.2 所示，它同时支持元组多实例和实体多实例，它可以看作是 MLR 模型的一种特例（C_i=TC，$2 \le i \le n$）。MLR 模型的表达能力[13]如图 2.3。从两图可以看出，ESW 模型与 MLR 模型在表达能力上是一致的。

2.操作及安全性比较

ESW 模型与 MLR 模型的操作语句数目相同，语义略有区别。在 MLR 模型中，高级主体被允许借用低级主体拥有的某个数据，而且这种借用是永久性的。因此，当低级主体拥有的数据被更新或删除时，相应的变化会向上传播到高级主体借用的数据中。由此可以看出，低级主体的操作可以修改高级主体的视图。本文认为这不仅不符合常规，也是有安全隐患的。低级用户可以频繁更新自己的数据或者故意输入错误数据，这会导致高级用户无所适从。ESW 模

型符合改进的 BLP 模型，即“不上读，同级写”。同时，ESW 模型认为：在高级主体向低级主体借用数据时，他认可的是低级主体当前所拥有的数据。也就是说，数据借用操作是一次性的而不是永久性的。在高级主体执行借用操作后，被借用的数据就成了高级主体自己的数据，不再受到低级元组数据变化的影响。

ESW 模型与 Smith-Winslett 模型相比，主要的区别是修改了更新操作，增加了借用操作。Smith-Winslett 模型把更新和借用都通过 UPDATE 语句实现，这样的缺点是某个安全级的主体不能在一个语句中同时借用几个安全级的数据。ESW 模型将更新和借用严格区分，这样更新操作就和传统的数据库一样，不再跨越多个安全级；而新增的借用操作可以保证用户同时借用几个安全级的数据。通过这些修改，ESW 模型的操作更加方便和实用。

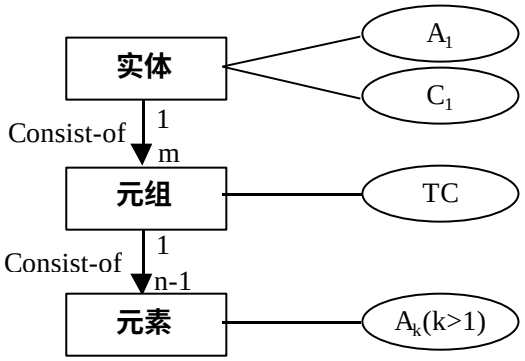

图 2.2 半元组级标记数据模型的表达能力

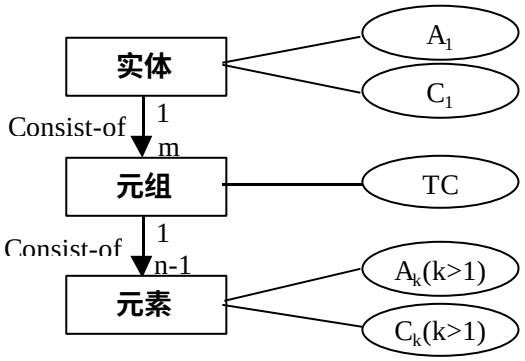

图2.3 元素级标记数据模型的表达能力

2.7 模型的正确性、完备性和安全性

2.7.1 正确性

本节将证明 ESW 模型是一个正确的数据模型。为了阐述正确性的真正含义，先给出两个定义：

定义 2.4 在 ESW 数据模型中，一个合法的数据库状态是指所有的关系实例都满足四个完整性性质[13]。

定义 2.5 一个正确的数据模型是指在该模型中，任何规定的操作语句序列都会将任一合法的数据库状态转换为另一个合法的数据库状态[13]。

从这两个定义可以看出完整性性质和操作语义对于模型的正确性证明起着至关重要的作用。事实上，传统数据库也有实体完整性和参照完整性，但是和 ESW 模型相比，它们很简单。传统数据库的操作语义同样简单，因此它的正确性证明不难。对于 ESW 模型来说，由于增加了安全级和操作语义，其正确性证明就复杂很多。

定理 2.1 ESW 模型是正确的。

证明：由定义 2.5 可知，要证明 ESW 模型的正确性必须要证明插入、删除、更新、借用操作能够将一个合法的数据库状态转换到另一个合法的数据库状态。

由于查询操作不会改变数据库状态，所以只需证明四种情况：

（1）由于 INSERT 操作语义已要求满足实体完整性、外键完整性和参照完整性，所以只需证明产生的关系实例满足多实例完整性即可。

因为插入操作允许，所以不存在元组 $t' \in r$， $t'[A1]=t[A1] \wedge t'[TC]=c$，这样在 c 安全级上不可能存在元组 t 的多实例，因此多实例完整性满足。

（2）对于 DELETE 操作，其本身已经强制要求满足参照完整性，所以只需要证明操作后的数据库状态满足实体完整性、外键完整性和多实例完整性。

① 由于在原始关系 r 中没有任何元组 t'在操作后会发生变化，所以实体完整性满足。

② 外键完整性被满足是因为在原始关系 r 中所有的元组满足外键完整性，执行删除操作后既没有元组被修改也没有元组被添加。

③ 多实例完整性被满足是因为原始关系 r 中所有的元组满足多实例完整

性，执行删除操作后既没有元组被修改也没有元组被添加。

（3）UPDATE 操作语义已要求满足四个完整性性质。

（4）由于 UPLEVEL 操作语义已要求满足多实例完整性，外键完整性和参照完整性，所以只需证明其满足实体完整性。

对于任何插入的元组 t，一定可以在原始关系 r 中找到一个元组 t'，$t'[A_1,C_1]=t[A_1,C_1]$。因为元组 t'满足实体完整性，所以操作后的数据库状态也满足实体完整性。

由上面的证明可知，所有的五个操作都可以将任何合法的数据库状态转换为一个合法的数据库状态，并且这些操作语句的任何序列也都将任何合法的数据库状态转换为一个合法的数据库状态。因此，ESW 模型是正确的。

2.7.2 完备性

本节将证明 ESW 模型的完备性。首先给出完备性定义：

定义 2.6 一个完备的数据模型是指在该模型中任何合法的数据库状态都可以通过一系列的数据操作语句转换到任何其它合法的数据库状态[13]。

定理 2.2 ESW 模型是完备的。

为了证明 ESW 模型的完备性，首先证明以下两个引理：

引理 2.1 任何合法的数据库状态都可以通过一系列规定的数据操作语句转换到一个空数据库状态[13]。

证明：根据 DELETE 操作的语义，可以执行如下操作：

（1）删除关系中所有的实体。一个实体在关系中可能对应于多个元组，依次删除该实体的每个元组。这可以通过认可这个元组的主体执行 DELETE 语句实现（WHERE 子句中指定该实体的标识符）。

（2）如果元组之间存在参照关系，则首先删除那些拥有参照元组的实体，然后删除那些拥有被参照元组的实体。这样的目的是保证参照完整性总是被满足并且 DELETE 操作不会被拒绝。

以上两步操作后，关系实例中的所有实体被删除，得到空数据库状态。

引理 2.2 一个空数据库状态通过一系列规定的操作后可以转换到任一合法数据库状态[13]。

证明：任一合法的数据库状态都可以按照如下方式构造出来：

（1）先加入拥有被参照元组的实体，然后加入拥有参照元组的实体。

（2）一个多级实体可以按下列方式加入：

① 一个实体的所有元组按其安全级从低到高的顺序加入。

② 每个元组被同级别的主体以如下方式加入：

- 基本元组 t_1 是由一个 INSERT 语句加入的。其中，所有满足 $t_1[A_i]\neq null$ 的 A_i 都出现在 INTO 子句中，同时 $t_1[A_i]$ 出现在 VALUES 子句中。
- 其他元组 t_m 都是由 UPLEVEL 语句和 UPDATE 语句合作完成的。首先把 A_1，C_1 和它们的值放在 WHERE 子句中，所有 A_i 和 $t_m[A_i]$ 满足 $t_m[A_i]=t_1[A_i]$ 都被包含在 GET 子句中。然后把 A_1，C_1 和它们的值放在 UPDATE 的 WHERE 子句中，利用 UPDATE 语句把 $t_m[A_i]\neq t_1[A_i]$ 的属性值更新。

这个过程将会被成功执行，因为：

（1）增加一个实体不会改变其他的任何实体。因为 A_1，C_1 和它们的值都被包含在 UPDATE 和 UPLEVEL 的 WHERE 子句中。

（2）实体完整性、外键完整性和多实例完整被满足，因为构造出来的数据库状态是合法的。

（3）参照完整性被满足，有两点原因：一是构造的数据库状态是一个合法的状态；二是因为拥有被参照元组的实体加入在前，拥有参照元组的实体加入在后。

根据 INSERT、UPDATE 和 UPLEVEL 的语义，可以证明新增的元组就是 t_1 和 t_m。

从引理 2.1 和 2.2 可以看出，任何合法的数据库状态都可以通过一系列规定的数据操作语句转换到任何其他合法的数据库状态。因此，ESW 模型是完备的。

2.7.3 安全性

安全性是 ESW 模型的根基，也是区别于其他数据模型的最大部分。因此，安全性的证明将直接关系到 ESW 数据模型是否满足无“向下”信息流的要求。本节的证明主要基于无干扰的概念。

本节将用到以下符号：

S：所有不同访问安全级的主体；

T：在一个数据库状态中具有不同元组安全级的所有元组。

对于任何访问安全级 c，

SV(c)：访问安全级小于等于 c 的主体集合；

SH(c)：S- SV(c)；

TV(c)：元组安全级小于等于 c 的元组集合；

TH(c)：T-TV(c)。

对于任一访问安全级 c，S= SV(c)∪SH(c)且 SV(c)∩SH(c)=∅；T=TV(c)∪TH(c)且 TV(c)∩TH(c)=∅。现在定义安全性需求如下：

定义 2.7 一个安全的数据模型是一个无干扰的模型，举例来说，对于任何访问级 c，删除来自于主体 $s_1 \in SH(c)$的任何输入不会影响到任何主体 $s_2 \in SV(c)$的输出[13]。

本节针对“输出”有两点假设：一是不考虑响应时间，本节的安全性证明不处理时序隐通道；二是每次 SELECT 语句返回的元组都是一个确定性的排序。这两点假设将使证明的焦点集中到数据模型的信号通道上。

ESW 数据模型的输入是一个来自于不同访问安全级主体的操作序列，这些操作包括 INSERT、DELETE、SELECT、UPDATE 和 UPLEVEL。输出是返回给主体的结果，包括：

（1）任何 SELECT 语句返回元组的集合；

（2）任何 INSERT、DELETE、UPLEVEL 和 UPDATE 语句成功或失败的信息。

定理 2.3 ESW 模型是安全的。

首先证明下面的两个引理：

引理 2.3 对于任何访问安全级 c，改变 TH(c)不影响任何主体 $s \in SV(c)$的输出。

证明：根据 SELECT 操作的语义，在处理一个 c'-主体 $s \in SV(c)$（$c' \le c$）发出的 SELECT 语句时，TH(c')中没有元组将在谓词 p 的计算中被考虑。因为 $c' \le c$ 意味着 $TH(c') \supseteq TH(c)$，所以改变 TH(c)不会影响到输出给主体 $s \in SV(c)$的返回元组集合。

对于任何 c'-主体 $s \in SV(c)$（$c' \le c$），根据 INSERT、DELETE、UPDATE 和 UPLEVEL 操作的语义：

（1）主体 s 发出的任何 INSERT 语句被拒绝，当且仅当

① 存在一个元组 $t' \in r$ 满足 $t'[A_1]=a_1 \wedge t'[TC]=c'$；或者

② 插入的元组 t 违反了实体完整性或外键完整性；或者

③ t 参照了某个 c'-元组 t'，而 t'不存在。

（2）主体 s 发出的任何 DELETE 语句被拒绝，当且仅当被删除的元组被某个 c'-元组 t'参照。

（3）主体 s 发出的任何 UPDATE 语句被拒绝，当且仅当

① A_1 中某个属性出现在 SET 子句中并且存在一个元组 $t' \in r$ 满足 $t'[A_1]=t[A_1] \wedge t'[TC]=c'$，其中 t 是更新以后的元组；或者

② t 违反了实体完整性或者外键完整性；或者

③ 当 A_1 中某个属性出现在 SET 子句中时，原始 t 被某个 c'-元组 t'参照；或者

④ t 参照了某个 c'-元组 t'，而 t'不存在。

（4）主体 s 发出的任何 UPLEVEL 语句被拒绝，当且仅当

① 存在元组 $t' \in r$ 满足 $t'[A_1]=t[A_1] \wedge t'[C_1] \neq t[C_1] \wedge t'[TC]=c'$，其中 t 是构造的元组；或者

② t 违反了外键完整性；或者

③ t 参照了某个 c'-元组 t'，而 t'不存在。

在所有的情况下，只有元组 t 或者 c'-元组 t'需要考虑。因为元组 t，$t' \notin TH(c') \supseteq TH(c)$，所以改变 TH(c)不会影响到主体 $s \in SV(c)$的输出。

证明结束。

引理 2.4 对于任一访问安全级 c，删除主体 $s \in SH(c)$的任何输入不会改变 TV(c)。

证明：一个主体只能通过 INSERT、DELETE、UPDATE 和 UPLEVEL 操作改变数据库状态：

（1）一个 c'-主体 $s \in SH(c)$（$c' > c$）进行的一次 INSERT 操作只能生成一个 c'-元组 t'。因为 $c' > c$，所以 $t' \notin TV(c)$；

（2）一个 c'-主体 $s \in SH(c)$（$c' > c$）进行的一次 DELETE 操作只能删除一个 c'-元组 t'。因为 $c' > c$，所以 $t' \notin TV(c)$；

（3）一个 c'-主体 $s \in SH(c)$（$c' > c$）进行的一次 UPDATE 操作只能修改一个原始的 c'-元组 t'。因为 $c' > c$，所以 $t' \notin TV(c)$；

（4）一个 c'-主体 $s \in SH(c)$（$c' > c$）进行的一次 UPLEVEL 操作只能修改一个原始的 c'-元组 t'或者增加一个新的 c'-元组 t'。因为 $c' > c$，所以 $t' \notin TV(c)$；

由上证明可知，删除主体 $s \in SH(c)$的任何输入不会改变 TV(c)。

证明结束。

上面两个引理可以用图 2.4 **来表示。**

现在证明定理：从引理 2.3 **和引理** 2.4 **可知，由于** $S=SV(c)\cup SH(c)$，$SV(c)\cap SH(c)=\varnothing$，$T=TV(c)\cup TH(c)$，$TV(c)\cap TH(c)=\varnothing$，**删除任何主体** $s_1\in SH(c)$ **的输入不会影响到** $s_2\in SV(c)$**的输出。**

证明结束。

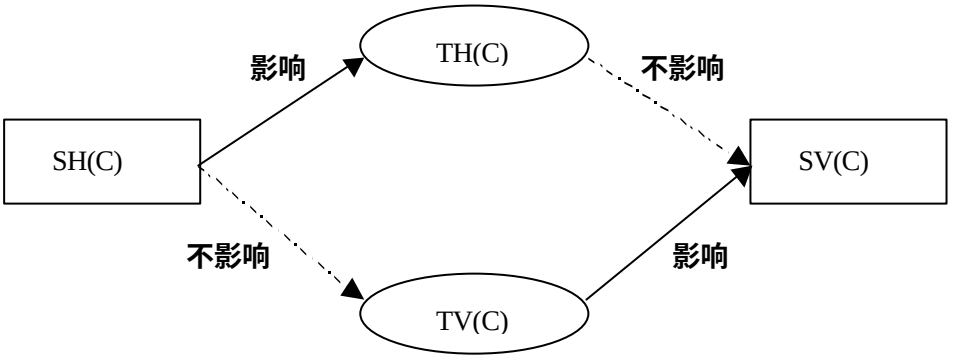

图 2.4 **高低安全级之间的影响**

2.8 本章小结

本章完整定义了 ESW **模型，并且证明这个模型是正确的、完备的和安全的。**ESW **模型扩充了** Smith-Winslett **模型，成功引入了** MLR **模型的数据借用操作。针对** MLR **模型数据借用所导致的低安全级用户修改高安全级用户视图的缺陷，本章重新定义了数据借用的概念，增强** BLP **模型所定义的“不下写”规则为“同级写”。**

在 ESW **模型中，对于任意给定的安全级，外观主键唯一标识一个实体，这避免了参照二义性。同时，从全局来看，一个外观主键可以标识多个不同的实体，允许了跨安全级的多实例，这样能够避免由于拒绝插入而导致的信息泄露。多实例完整性和参照完整性的重新定义消除了多实例带来的语义模糊性、查询模糊性和更新引起的元组数据激增等问题。**

第三章 推理控制

在多级安全数据库系统中，用户往往可以利用低安全级的数据获取他不应该知道的高安全级的数据，这就产生了推理问题。数据挖掘技术的发展，导致这种威胁日益严重[32]。因此，推理控制对于保证多级安全数据库中数据的安全具有非常重要的意义。高安全级别的 DBMS 在其强制访问控制措施中都包含检测和消除推理通道的功能。本章介绍了检测推理通道的方法，在此基础上描述了一种两阶段推理控制策略。这种策略包含静态提升安全级和动态推理控制两部分。静态提升安全级的算法能够保证在信息易用性损耗较小的情况下控制推理通道。而在数据库运行阶段，利用动态推理控制算法可以在基本不影响访问速度的情况下切断剩余的推理通道。

3.1 引言

早在 80 年代初期，数据库研究者们就已开始对安全数据库的推理控制问题进行研究，研究发现安全级较低的用户通过不同形式的推理可以访问高安全级的数据。最常见的表现形式是从一系列的查询结果中进行判断、推理。举例来说，支持某项目的公司是敏感信息，然而用户可以通过查询项目的会议名、参加会议的人员和与会人员的工作单位推断出项目名和支持它的公司。还有一种比较复杂的情况是大量的用户进行合谋（分工合作）推理。Jajodia 等证实了数据库自身约束是造成多级安全数据库中的大多数推理通道的原因。目前有关推理通道的问题不仅仅局限于多级数据库，而且已拓展到其他有关统计数据库、数据挖掘以及基于 WEB 的推理问题等领域。

推理控制问题实质上就是推理通道的检测与控制。其技术大致可分为两类：静态控制和动态控制。静态控制是指在设计阶段发现推理通道，通过修改设计提高某些敏感数据的安全级[33,34]。这种方法的缺点是可能导致给属性或原始数据分配过高的安全级，削弱数据的可用性。动态控制是指在查询期间消除推理通道，如果检测到推理通道就拒绝查询或修改查询，以保证敏感信息安全[35]。这种基于查询的推理控制代价高，且仍会导致敏感信息的泄漏（从拒绝的

查询来推理敏感信息[36])。本章针对当前细粒度访问控制趋势，介绍了推理通道的检测方法，在此基础上提出了两阶段推理控制策略并给出了算法。通过这一系列步骤可以很好地保护敏感信息、解决推理问题，同时又能保证高访问效率。

3.2 术语的定义及相关理论

本章主要讨论推理问题中的函数依赖。在数据库推理安全研究中，如果安全级定义到关系或元组级别，那么讨论函数依赖造成的推理通道问题是没有意义的[37]。在第二章中提到的安全模型是半元组级标记数据模型，所以，有必要重新定义多级关系模式及多级关系实例。

定义 3.1 多级关系模式是用 $R(A_1, C_1, \ldots, A_n, C_n, TC)$ 的形式来表示，其中 A_i 表示实体的属性，C_i 是与属性 A_i 相关联的安全级，C_i 定义在 $[L_i, H_i]$ 上，$H_i \geq L_i$，TC 为 $\cup(L_i, H_i)$，表示元组的安全级，H 为最高安全级。A_1 表示外观主键，这里的主键与传统关系数据库中的主键不同，允许键值重复存储，并假设已解决多实例问题。

定义 3.2 多级关系实例用 $r(A_1, C_1, \ldots, A_n, C_n, TC)$ 表示，是不同元组 $(a_1, c_1, \ldots, a_n, c_n, tc)$ 的集合，其中 a_i D_i，D_i 为属性域，C_i $[L_i, H_i]$，或 a_i=null，c_i $[L_i, H_i] \cup$null，$tc \geq lub\{c_i \mid c_i \neq null\}$ $(i=1, \ldots, n)$。lub 表示最小上界。

定义 3.3 推理通道（Inference Channel）：InfCh(H)={X Uá | ë(X)< ë(H) ÎNHFER(X ∃Z Uá[ë(Z)<ë(H) INFER((Z∪{X}),H))>(INFER(Z å),H)+ 。其中 Uá 表示数据集合；ÎNHFER(X 为推理函数。推理通道包含两部分内容："或"的前半部分表示由低安全级的 X 可以推出较高安全级的 H；"或"的后半部分表示低安全级的 Z 自身并不能推出 H，但与 X 结合后，能推出较高安全级的 H。

函数依赖从语义上确切地表示了关系模式属性之间的对应关系，是泄露敏感信息的重要推理通道。但用户仅仅知道函数依赖 FD：X Y ，而不知道属性 X 和属性 Y 之间值的映射关系是不足以造成敏感信息泄露的，这里假设用户知道 FD：X Y 的同时，也知道其属性值之间的相关映射。

定义 3.4 安全的函数依赖推理：对于函数依赖 X Y ，如果属性 X 的安全

级不低于 Y 的安全级，即$\lambda(X) \geq \lambda(Y)$，能访问属性 X 的主体一定能访问属性 Y，称 X Y 是安全的函数依赖推理。对于左部与右部有多个属性的函数依赖 B_1，B_2，..，B_n X，A_1，A_2，..，A_m Y，则要求 $lub(\lambda(B_1), \lambda(B_2), .., \lambda(B_n)) \geq lub(\lambda(A_1), \lambda(A_2), .., \lambda(A_m))$。安全的函数依赖集 F 推理产生的全部函数依赖 F^+集是安全的。

本章把存贮在数据库中的有信息的单元或者相互间的关系看作一个对象。就本章引言中的例子来说，“工程名”，“会议名”，“人员信息”和“公司名”都是对象。一个推理通道是指完成一个推理得到敏感信息所需的最少对象的集合。推理通道的长度是指集合中对象的个数。例如“工程名”，“会议名”，“人员信息”和“公司名”组成了一个长度为 4 的推理通道。

对于给定的敏感信息，对其进行推理控制的目的在于检测出所有的推理通道，然后根据推理控制策略进行相应处理。这一过程必须符合强制访问控制的要求。保护敏感信息不被直接访问是多级安全系统的核心，推理控制只是系统的一个有效补充。所以，应该避免因破坏强制访问控制策略而导致敏感信息被直接泄露。

3.3 推理通道的检测

推理通道的检测可以在两个层面上进行：一个是在数据库设计过程中，利用语义数据建模技术来检测推理通道；另一个是在事务执行阶段，通过分析事务的操作（“读”或者“写”操作）来判定是否会导致非法推理。

利用语义数据建模技术最早的例子是 HINKE 的 ASD 视图工程，在数据库设计中通过构造语义关系图来表示可能的推理通道。其中，数据条目用图的结点来表示，数据之间是否存在推理通道用边来表示。如果在结点 A 和结点 B 之间存在两条路径，一条路径上包含了图中所有的边，而另一条路径上不包含图上所有的边，那么结点 A 和结点 B 之间就有可能存在推理通道。然后进一步分析确认是不是真正的推理通道。

Buczkowski 提出了类似于 Morgenstern 的系统。该系统用 PINFER 函数表示数据之间的关系。例如 PINFER(X Y)表示从 X 推导出 Y 的概率，并用模糊逻辑来计算概率，它是根据各种事实之间的依赖关系程度计算出来的。

上面两种方法都是建立在数据库的设计和分类层面上。另外有一些研究者

力图在事务执行层面，通过分析查询的方法来解决推理通道问题。Mazumder等学者使用定理证明器来分析事务的安全，进而决定是否从数据库的完整性约束、事务的前条件、输入事务的数据类型等可以推导出预先定义的秘密。目前还有种方法是检测用户的查询历史，通过判断准则确定是否有高安全级的信息可以推导出来。如果有，则拒绝查询。这种方法所消耗的存储空间较大，而且会影响查询速度。

3.4 两阶段推理控制

推理控制应该遵循两大原则：

（1）最大可用原则：在保证数据库安全的前提下，尽可能使数据库中的数据最大可用。

（2）短通道优先原则：推理过程用到的关联关系越多，推理通道就越长。相对于长通道而言，短通道带来的危害更迅速更直接，应该优先消除。

本节将属性安全级调整和动态推理控制相结合，描述了一种两阶段推理控制的方案。这种方案充分整合了静态推理控制和动态推理控制的优点，能够很好地控制函数依赖所引起的推理通道。

3.4.1 属性安全级调整

属性安全级调整是最常见的推理控制的方法，它的优点是能够在数据库设计时切断推理通道，不会造成丝毫的信息泄露。缺点是使数据的可用性降低，也就是违反了上面提到的第一条原则。由于调整函数依赖的属性安全级的方案常常不止一个，因此如何使安全级提升后信息损失最小是属性安全级调整的最重要的问题。本节主要描述了一个消除函数依赖推理危害的最小信息丢失安全级调整算法。

3.4.1.1 最小信息丢失安全级调整

为了反映实际的信息丢失，下面用公式表示安全级调整问题。每个属性在每个可允许的级别 m 都被赋予一个权值 w_{im}，w_{im} 为一正数，表示属性 A_i 信息对应的有效可用性。在这种方案下，如果 m>n，有 $w_{im}<w_{in}$。这也符合实际情况，安全级越高，数据的可用性越小。这样，每次提高 A_i 的安全级，就得到

新的减少的权值。一个函数依赖中属性总权值在调整前和调整后的差值就是信息丢失。

下面给出了一个递归的属性安全级调整算法，使得对于给定的函数依赖集，经调整后，不存在函数依赖危害。

算法 3.1：SCLA（Static Classification Level Adjustment）**算法**。

输入：

（1）**函数依赖集** F；

（2）**属性全集的一个划分** A_IN，**每一个划分块包含一个属性和该属性的密级**；

（3）MLS/DBMS **中属性的权值方案。权值方案的设定原则就是：针对同一个属性而言，安全级与权值成反比。安全级越高，属性的权值越小。**

输出：

属性全集的一个划分 A_OUT，**每一个划分块包含具有相同密级的属性和这些属性的密级。** A_OUT **表示新的密级分配，它不仅消除了函数依赖危害，而且具有最小的信息丢失。**

步骤：

（1）**初始化。令** MIN=+∞（MIN **表示最小的权值丢失**）；

（2）**逐一检查每个函数依赖**：$B_1B_2..B_n \rightarrow A_1A_2..A_n$，**如果** $lub(B_1, B_2, .., B_n) \geq lub(A_1, A_2, .., A_n)$，**则删除这个函数依赖。检查完毕后得到的函数依赖集称为** F_r；

（3）**右分解。**F_r **中每个右边包含多个属性的函数依赖按如下方式分解：令**$(X \rightarrow A_1A_2..A_n) \in F_r$ **分解为**$\{X \rightarrow A_1, X \rightarrow A_2, .., X \rightarrow A_n\}$。**所有函数依赖右分解后再删除重复的，然后得到的函数依赖集称为** F_t；

（4）**左分解。对** F_t **中的每个依赖按下列方式分解：令**$(B_{i1}B_{i2}..B_{il} \rightarrow A_i) \in F_t$ **分解为** $F_i=\{(B_{i1}, A_i), (B_{i2}, A_i), .., (B_{il}, A_i)\}(1 \leq i \leq m)$**这里**$|F_t|=m$；

（5）**从每个**$F_i(1 \leq i \leq m)$**中取出一个二元组，形成二元组的集合。令** OPEN **表示所有二元组集合的集合**；

（6）**对于** OPEN **中的每个二元组集合，若其包含的二元组中存在左边属性相同的二元组，则只保留右边属性安全级最大的二元组，得到新二元组集合的集合称为** N_OPEN；

（7）**循环。从** N_OPEN **中选择一个二元组集合** C，N_OPEN:= N_OPEN-

{C}，调用函数 FD_ADJUST(C)，直到 N_OPEN=∅；

函数 FD_ADJUST(C)

```
{
    CNT=0;                                  //CNT 表示当前权值丢失
    U=A_IN;
    while(C 不为空)
    {
        从 C 中取出一个二元组 c，且令 c 的左边属性为 M，右边属性为 P;
        C=C-c;
        if(属性 M 的安全级<属性 P 的安全级)
        {
            Bp=Bp∪BM;                       // Bp 是 U 中包含属性 P 的划分块
            U=U-BM;
            /*W(M, L(M))为属性 M 在级别 L(M)的权值)*/
            CNT=CNT+(W(M, L(M))-W(M, L(P)));
            if(CNT≥MIN)    return;
        }
    }
    MIN=CNT;
    A_OUT=U;
}
```

在算法 SCLA 中，调用了一个实际进行调整工作的函数，称为 FD_ADJUST。为了提高算法的效率，在步骤（1）和（2）中，首先删除了那些不需要调整的函数依赖。在步骤（6）中，删除了不需要调整的二元组。同时，当某个属性被调整后，比较当前权值丢失的总值 CNT 和当前权值丢失的那个最小值 MIN。如果前者大于后者，函数直接退出，进行其他情况调整。

3.4.1.2 特殊情况下的算法改进

在某些特殊情况下，这个算法可以改进，从而使得算法的效率进一步提高。假设所有的属性在同一安全级下所对应的信息量是一样的，那么在同一安全级下，所有属性对应的权值也是相同的。也就是说：如果 L(P)=L(M)，那么 W(M,

L(M))=W(P, L(P))。这种情况下，算法改进如下：

算法 3.2：BSCLA（Better Static Classification Level Adjustment）**算法。**

输入：

（1）**函数依赖集** F；

（2）**属性全集的一个划分** A_IN，**每一个划分块包含一个属性和该属性的密级；**

（3）MLS/DBMS **中属性的权值方案。权值方案的设定原则就是：针对同一个属性而言，安全级与权值成反比。安全级越高，属性的权值越小。**

输出：

属性全集的一个划分 A_OUT，**每一个划分块包含具有相同密级的属性和这些属性的密级。** A_OUT **表示新的密级分配，它不仅消除了函数依赖危害，而且具有最小的信息丢失。**

步骤：

（1）**初始化。令** MIN=+∞（MIN **表示最小的权值丢失**）；

（2）**逐一检查每个函数依赖：** $B_1B_2..B_n \rightarrow A_1A_2..A_n$，**如果** $lub(B_1, B_2, .., B_n) \geq lub(A_1, A_2, .., A_n)$，**则删除这个函数依赖。检查完毕后得到的函数依赖集称为** F_r；

（3）**右分解。** F_r **中每个右边包含多个属性的函数依赖按如下方式分解：令**（$X \rightarrow A_1A_2..A_n$）$\in F_r$ **分解为** $\{X \rightarrow A_1, X \rightarrow A_2, .., X \rightarrow A_n\}$。**所有函数依赖右分解后再删除重复的，然后得到的函数依赖集称为** F_t；

（4）**左分解。对** F_t **中的每个依赖按下列方式分解：令**（$B_{i1}B_{i2}..B_{i1} \rightarrow A_i$）$\in F_t$ **分解为** $F_i=\{(B_{i1}, A_i), (B_{i1}, A_i), .., (B_{i1}, A_i)\}(1 \leq i \leq m)$**这里**$|F_t|=m$；

（5）**从每个** $F_i(1 \leq i \leq m)$**中取出一个左边属性安全级最大的二元组，形成一个二元组的集合。令** S_OPEN **表示这个二元组集合；**

（6）**对于** S_OPEN **中的每个二元组集合，若其包含的二元组中存在左边属性相同的二元组，则只保留右边属性安全级最大的二元组，得到新二元组集合称为** P_OPEN；

（7）FD_ADJUST(P_OPEN) 。

可以看出算法改进了步骤（5）**开始的所有步骤，其主要原因是既然所有属性在同一安全级上的权值相等，那么只需调整函数依赖左侧安全级最高的属性就可以保证信息丢失最少。**

3.4.1.3 算法的性能

在函数 FD_ADJUST( )中，while 循环的执行对时间复杂度影响较大，最坏情况下需要 O(m)时间，m 为 C 中二元组的个数。在算法 CLA 中，步骤（5）对复杂度的影响最大，它产生了$\pi|F_i|$个组合，对于 F 中不同的函数依赖，$|F_i|$各不相同。在最坏的情况下，假设$|F_i|=|A_IN|$。这样，就得到了最坏情况下总的时间复杂度为 $O(m|A\_IN|^m)$。特殊情况下的改进算法中，总的时间复杂度为 O(m+m|AIN|)。

3.4.2 动态推理控制

静态安全级调整总是会影响信息的可用性的。调整的过程中，如果发现调整后数据的可用性不满足要求，那就对这个函数依赖进行标记，对它进行动态推理控制。动态推理控制主要针对两种情况：一种是密级调整信息可用性损耗过大；另一种就是在数据库运行期间发现的推理通道。

动态推理控制的思路来源于安全数据库分级，利用动态安全级来控制推理通道。方法可分为两个阶段：第一阶段是初始化阶段。假设有一个推理通道的长度为 n，那么需要有 n 个密钥。也就是说密钥的个数与推理通道的长度在数值上相等。密钥的生成非常的简单，不需要特殊的算法，因而可以保障较高的初始化速率。在初始化阶段，推理通道中的所有对象都被绑定一组密钥。这一阶段的动作只需执行一次。第二阶段就是在用户查询过程中的处理。用户不需要保存密钥，这杜绝了暴力破解密钥的可能性。处理思想是这样的：当一个密钥被用来访问某个对象后，其他的针对此对象的查询都使用同一个密钥。这通过删除与之联系的其他密钥来实现。

密钥是采用这样一种方法生成：假设一个长度为 n 的推理通道中所有对象的最低安全级为 t，同时假设用户的访问等级为 f（此处假定安全级是自然数，如果不是，则将安全级映射到自然数集），则生成的密钥前 n-1 个是 t，最后一个大于 f 即可。

下面分别描述动态安全级方法在三种不同情况下是如何解决推理控制问题的。

3.4.2.1 **单推理通道**

首先考虑数据库中只有一个推理通道的情况。 m **表示推理通道的长度，推理通道中的对象分别为** O_1，..，O_m。U **代表数据库的某个使用者。每个** O_i **绑定的密钥集都一样，用** K(O)**来表示。在这个简单模式下，密钥的总数为** m。**假设推理通道中对象的最低安全级为** t，**用户的访问等级为** f，**则密钥集可以为**{t，t，..，t，f+1}，**其中** t **的个数为** m-1。

当用户尝试访问某个对象时，一个较小的密钥被选中。如果该密钥大于用户的访问等级则提示无权访问。否则，本对象其余的密钥全部删除，其余对象中均删除一个与被访问对象密钥相同的密钥。最终，有一个对象的密钥高于所有用户的访问等级，没有人可以访问它，这个对象被称为“保留对象”[38]。**一旦“保留对象”被确定，不论有多少用户进行合谋推理都是没有意义的。图** 3.1 **就是一个简单的例子。** O_1，O_2，O_3 **是一个推理通道中的三个对象。初始的时候每个对象都有三个密钥。当** O_1 **被访问后，每个对象拥有的密钥变成了图** 3.1 **的下半部分所示。在这里我们假设访问的用户的安全级均为** C，**这样当用户访问了三个对象中的任意两个后，剩下的一个必定是访问不了的。**

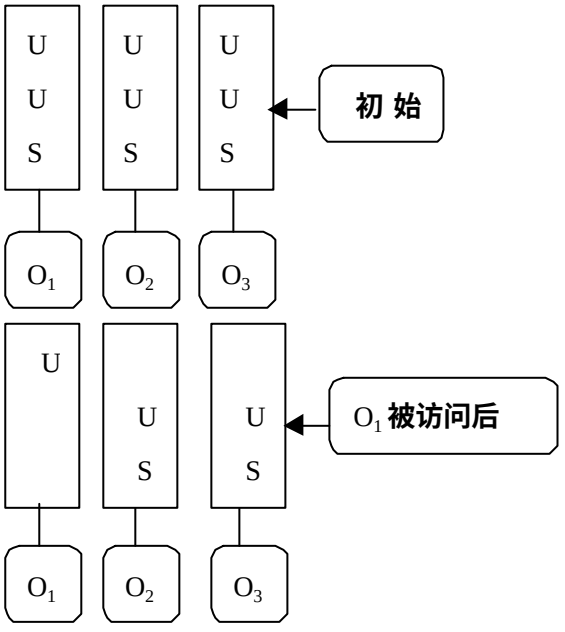

图 3.1 **动态推理控制的例子**

下面用算法来描述该模式。在此算法中，K **代表** m **个密钥的一个集合。**

算法 3.3：**一个用户访问对象** O_i **的单推理通道控制**

Key initialization:

$K(O_i)=K$; //i=1, …, m.

Query processing:

Input: i;

Select a smaller key k_j from K;

if $k_j>f$ then

Output “access denied”;

else

if $|K(O_i)|!=1$

$K(O_i)=\{k_j\}$;

$K(O_s)=K(O_s)\backslash\{k_j\}$ for all s and $|K(O_s)|>1$;

Deliver O_i to the user.

3.4.2.2 **不相交的多推理通道**

现在考虑无相交对象的多推理通道问题。不相交是指两个推理通道之间没有相同的对象。解决的方法和 3.4.2.1 **是类似的。为方便起见，用** C **代表数据库的一个推理通道：**C_1，…，C_p，**推理通道** C_j **的长度用** m_j **表示，**m_{max} **表示最长的推理通道的长度。密钥集合** K **包含了** m_{max}**个密钥。对于任一推理通道** C_j，K_j **是包含了** m_j-1 **个** t，1 **个** f+1 **密钥的集合。**

算法 3.4：**一个用户访问对象** O_i **的不相交的多推理通道控制**

Key initialization:

$K_j=\{t, …, t, f+1\}$; //j=1, 2, …, p.

if $O_s \in C_j$ then

$K(O_s)=K_j$, for all possible s.

Query processing:

Input: i;

if find j such that $O_i \in C_j$ then

{

select the smaller key k_p from $K(O_i)$;

if $k_p>f$ then

output “access denied” and quit;

else

{

if |K(O_i)|!=1

K(O_i)=k_p;

K(O_s)=K(O_s)\{k_p} for all O_s C_j, si and | K(O $_s$)|>1;

Deliver O_i to the user;

}

}

else

output “information not found”.

3.4.2.3 相交的多推理通道

最后，考虑情况比较复杂的相交的多推理通道。相交的多推理通道是指一些对象同时存在于两个或两个以上的推理通道中。相交的多推理通道又可以分为两种情况。一种情况是相交的对象不是任一通道的保留对象。在这种情况下，用户访问该相交对象和访问其他对象一样。另一种情况是相交的对象是一个保留对象，这样，用户的访问请求会被拒绝。

算法在下面描述。密钥集的初始化和 3.4.2.2 **小节的一样。对于相交的对象，密钥集的数目等于该对象在所有推理通道中出现的频率。我们用** $K_j(O_i)$**代表对象** O_i **在通道** C_j **中的密钥集。**

算法 3.5：**一个用户访问对象** O_i **的相交的多推理通道控制**

Key initialization:

K_j = {t, …, t, f+1}; //m_j-1 **个** t, j=1, …, p.

for All possible s do

{

if O_s C_j then $K_j(O_s)$ = K_j; // j=1, …, p.

}

Query processing:

Input i:

if O_i C_j then

```
{
    select the smaller key k_p from K_j(O_i);
    if k_p>f then
        output "access denied" and quit;
    else
    {
        if |K_j(O_i)|=1
            Deliver O_i to the user;
            return;
        K_j(O_i)= k_p;
        for every C_j such that O_i  C_j do
        {
            select the smaller key k_p from K_j(O_i);
            while O_s  C_j and |K_j(O_s)|>1 do
            {
                K_j(O_s)=K_j(O_s)\{k_p};
                If K_j(O_s)=={f+1} then
                    K_s(O_s)={f+1} for all r such that O_s  C_r;
            }
        }
        Deliver O_i to the user;
    }
}
else
    Output  "information not found".
```

3.4.2.4 **算法的性能**

要解决推理问题，必须要考虑三大因素：安全性，保证没有用户可以利用这条推理通道得到敏感数据；数据的可用性，在保证安全性的基础上使数据的可用性损失尽可能小；响应时间，解决推理通道的同时必须要保证查询响应能够满足要求。

利用上面的算法，所有的用户都可以访问长度为 m 的推理通道中相同的 m-1 个属性。因此，不管是单独推理或者联合推理都不可能得逞。同时，数据的可用性达到了最大，不管推理通道的长度是多少，都只有一个属性不能访问，其余均可以访问。在响应时间上，由于密钥的数量只取决于数据库中最长的那个推理通道，密钥集所占用的空间可以忽略不计，其响应速度也与没有采用动态推理控制算法的数据库的响应速度近似相同。

3.5 小结

本章介绍了检测推理通道的方法，同时描述了一种两阶段推理控制方案。此方案将静态提升安全级和动态推理控制相结合。针对静态提升安全级，描述了一个最优的安全级调整方案并且给出了特殊情况下的算法改进。对于调整后信息损失过大或者在数据库运行阶段发现的推理通道，利用动态推理控制来解决。由于动态推理控制算法中保留对象是不固定的，因此与静态提升安全级相比更加灵活。两者的结合，极大地消除了由函数依赖所引起的推理通道。

第四章 隐通道控制

在设计一个多级安全数据库系统时，系统安全策略规定一个高安全级的用户不能把高级数据写到低级别数据对象中，也就是“不下写”策略。然而，在系统实际使用时，攻击者有可能构造一组表面上合法的操作序列，将高级数据传送给低级用户。这种隐蔽的非法信息通道称为隐通道。比如某个资源的读写数目是有限的，若某个高级用户要传递给低级用户一些信息，他就可以通过耗尽这个客体的资源这种形式来传递。这种资源耗尽型的隐通道是无法避免的[39]。本章介绍了隐通道的分类，隐通道的标识，隐通道的处理策略，提出了隐蔽流图法。这种方法集成信息流法和隐蔽流树法的优点，具有较高的应用价值。

4.1 隐通道的分类

隐通道的定义在第一章已经介绍过。在现实的系统实践中，可以按照不同的特性对隐通道进行分类。下面根据隐通道是否使用了存储变量、信息在传递时是否失真，以及是否有多个信道共享同步信号等特征对其进行分类。

（1）双向同步和单向同步隐通道

在利用隐通道传递信息之前，信息发送者和接收者必须规定好同步方式。同步的目的是让一个进程通知另一个进程，它已经完成了对一个数据的读或者写。根据发送者与接收者之间的同步关系可以将隐通道分为双向同步隐通道与单向同步隐通道。若一个隐通道除了包含一个用户数据传输的变量外，还包括两个同步变量，其中一个同步变量用于发送者到接收者的同步，另一个用于接收者到发送者的同步。这种隐通道称为双向同步隐通道[6]。在多级强制安全系统中，所有高级用户可以直接读低级别的信息。但在这类系统中，从发送者到接收者的同步依然不可缺少。这种只需要一个同步变量的信道被称为单向同步信道。

（2）存储隐通道和时序隐通道

存储隐通道与时序隐通道的区分最初见于文[40, 41]等一些著作，而到后来的文[42]通过分析磁盘臂隐通道，指出在本质上，这两种隐通道是没有区别

的，或者说这两个种类的隐通道之间的界限是模糊的。但是在实际分析隐通道时，人们习惯会加以区分。两者的主要差异在于，存储隐通道的发送者直接或间接地修改一个存储变量，接收者读取同一个存储变量。而时序隐通道的发送者通过对使用资源的时间（比如 CPU 时间）的影响来发送信息，接收者通过观察响应时间的改变来接收信息。

由于一个信道要用到数据传输和同步信号两种变量，因此，人们把一个信道使用的同步变量和数据变量都是存储变量的隐通道称为存储隐通道；如果其中至少有一个是时序变量就称为时序隐通道。存储隐通道和时序隐通道是最常用的隐通道分类方法。

（3）无噪声隐通道和有噪声隐通道

根据信号传送的质量，隐通道可分为无噪声隐通道和有噪声隐通道。如果接收者收到的符号与发送者发送的符号完全一样的可能性为 1 的信道称为无噪声信道。对隐通道来说，每个符号是一个比特。这就是说，不管系统中其他用户的行为如何，接收者可以保证每次都能正确的接收到一个比特。

反之，如果接收者收到的符号与发送者发送的符号完全一样的可能性小于 1，这样的隐通道称为有噪声隐通道。在隐通道的控制策略中，有时杜绝隐通道是不可能的，那就可以在确定的隐通道中引入噪声，达到控制它的目的。

（4）聚焦隐通道和非聚焦隐通道

一个同步变量可以用于同一对发送者和接收者之间的多个数据变量来同时传递信息。这些不同的数据变量共同使用同一个同步变量会降低通信的同步对资源的消耗。这种信道称为聚焦隐通道。

根据发送者和接收者读写数据变量的方式，聚焦隐通道可以分为串行、并行与混合隐通道。如果所有的数据变量的读写是顺序进行的，那么这个隐通道是串行隐通道；如果对所有的数据变量的读写是由一组不同的发送进程和接收进程并行进行的，就称之为并行隐通道；两者兼备的称为混合隐通道。

4.2 隐通道的标识

20 世纪 80 年代中期，由于 TCSEC 对 B2 级别以上安全系统隐通道分析的要求，隐通道分析技术的研究也随之升温。彻底搜索隐通道是隐通道分析中最为困难的一环。学者们先后设计了多种方案，较有代表性的有信息流法

（Information Flow）、**无干扰法**（Noninterference）、**共享资源矩阵法**（SRM）、**语义信息流法**（Semantic Information Flow Analysis）、**隐蔽流树法**（CFT）、**改进的** SRM **法**（Extended SRM）**等。这些方法从本质上看都是信息流法，其差异只是着眼点不同。现在的分析隐通道的主流观点是：只有采用信息流分析技术，才能发现隐通道。因此，对隐通道的标识本质上就是分析系统中的非法信息流**[43]。

4.2.1 无干扰法

无干扰模型（Noninterference）**是安全领域的一个经典模型，它首先由** Goguen **和** Meseguer **在文**[44]**中提出。无干扰模型从本质上形式化了这样一个想法：在一个安全系统中，一个用户不能意识到任何不由它所支配的用户（对于多级安全系统，就是安全级别比他高的用户）的任何操作。它将一个安全系统的** TCB **抽象成一个状态机**（state machine），**假设** A **和** B **是这样一个抽象** TCB **的两个用户，如果** w **是对这个状态机从初始状态的输入，并且这个输入的最后一个操作是来自** B **的，那么定义** B(w)**为** B **由最后一个输入获得的输出。同时** w/A **定义为将** w **中来自** A **的输入删除以后得到的子串。这样，定义** A **非相干于** B **当且仅当对于所有可能的以** B **的输入结尾的输入串** w，B(w)=B(w/A)。**如果一个多级系统中不存在隐通道，那么任何一个用户都应该非相干于级别比它低的用户。**

在实际系统分析中，这种方法非常繁琐。为此，Goguen **使用一个简化了的定理来体现非相干性，这就是所谓的“展开定理”**（unwinding theorem）。**这个定理使得实际检查一个系统的非相干性成为可能。这个定理指出：系统的状态可以按照某个用户** B **分成等价类，一个** B **等价类内的所有成员满足：对于任何** B **的输入，**B **所获得的输出是相同的。对于任何输入的下一个状态也是** B **等价的。多级系统中的安全级可以被用来作为等价状态的标签。**

4.2.2 共享资源矩阵法

共享资源矩阵法(Shared Resource Matrix)**首先由** Kemmerer **在文**[45]**中提出，尽管有一定的局限性，但仍然是当前最实用的隐通道发现方法。它不仅可以用于代码分析，还可以用于规范分析甚至模型和机器代码分析。它的基本思想是：既然产生隐通道是因为系统中存在共享资源，那么如果找出所有用于读**

写的系统资源和操作，就能找到所有隐通道。采用 SRM 方法进行隐通道分析步骤如下：

（1）列举一个主体可以访问或修改的所有共享资源，检查每个资源，确定它是否可能被用于在主体间隐蔽地传递信息；

（2）构造矩阵并且根据规范或代码确定操作原语对该变量的读写关系；

（3）生成该矩阵的传递闭包，从而得到所有的间接读写操作；

（4）分析该矩阵，得到形如<Pa，Var，Pv>的三元组，其中 Pa 和 Pv 是执行修改和访问动作的原语，Var 是二者共享的全局变量。

这种方法也有两个缺点：一是它不能证明单个的 TCB 原语是否安全，因此，如果增加了一个新的 TCB 原语，则需要重新分析；二是共享资源矩阵法会发现伪隐通道，解决的方法是结合信息流公式法来自动消除这些伪隐通道。

4.2.3 信息流法

D. Dening 于 1976 年提出了信息流法（Information Flow）[46]。在他的文章中详细阐述了信息流的概念，并提出了信息流的格模型。在源代码中，信息流描述了两个变量之间的因果关系。在任一变量 b 或变量 a 的函数中，如果能通过观察新状态下 b 的值推断出旧状态下 a 值的信息，就存在从 a 到 b 的信息流。信息流分析过程包括找出信息流并检验它是否违反信息流规则。分析的过程是：每次分析一个函数，直到分析完函数中的每个表达式，并把每一对变量之间的信息流写成一个流语句（例如：若有语句“d:=e”，则信息从变量 e 流向变量 d，记为 d←e）。这样，就从一个给定的函数产生很多流语句；接着根据信息流格模型规则“若信息从 b 流向 a，则 a 的安全级必然支配 b 的安全级”对这些流加以检验，找出非法流。如果非法流是真实存在的，就标记为隐通道。

信息流分析法从理论上来说可以找出系统中所有的信息流安全缺陷，而不会漏过一条非法流，能彻底地搜索出所有的隐通道。但是，由于很多系统资源不属于安全策略模型中明确规定的，往往没有指定安全级别，因此利用信息流法有时无法证明一个信息流是非法流，需要大量的人工排查。

4.2.4 隐蔽流树法

隐蔽流树法（Covert Flow Trees）首先由 Porras 和 Kemmerer 在文[47]中提出，它的中心思想和共享资源矩阵法类似，也是考虑 TCB 原语与 TCB 变量之

间的改写和（间接）读取关系。所不同的是，它采用了形象的直观树形的表示方法，并且支持图形化的分析工具。隐蔽流树分析的第一步是根据 TCB 的说明或者源代码对每个 TCB 原语构造 3 个列表：引用列表、改写列表和返回列表，分别对应了这个原语所读取的变量、所能改写的变量以及返回结果给用户的变量。在具体构造一个 CFT 的时候，选择一个 TCB 的变量作为这个 CFT 分析的焦点，以之作为根节点。然后，根节点分作两个分支，左边的分支代表发送者对这个变量的改写动作可能使用的 TCB 原语，右边分支代表接收者对这个变量的改变的认知所使用的 TCB 原语。紧接着的工作是继续扩展右边的认知分支。扩展的原则是：如果这个变量出现在某个原语的返回列表中，那么将那个原语添加称为一个“直接认知”的分支，如果它不出现在任何一个原语的返回列表中，那么添加“失败”的叶节点作为“直接认知”。如果这个变量出现在一个原语的引用列表中，并且这个原语的改写列表非空，那么这个原语作为一个“间接认知”的分支被添加。然后，添加这个原语中改写列表中的每一个变量作为一个新的认知右边分支，这个原语作为左分支。对于每个新的这样的认知分支，不断递归地进行以上的操作，直到所有的叶节点都是“直接认知”或者“失败”。通过穿越构造完成的 CFT 可以发现潜在的隐通道。穿越的操作将 CFT 中所有原语的节点分成两个列表，一个包含了改写这个变量的 TCB 原语，另一个包含了直接或间接认知这个变量的原语。这分别来自两个列表中的任两个原语的组合构成潜在的隐通道。

4.3 隐通道处理措施

处理隐通道的常见技术包括消除法，带宽限制法和审计法。在大多数安全数据库中，这三种方法被结合起来使用。

4.3.1 消除法

消除法是最彻底的消除隐通道的方法，同时也是最复杂的方法。为了消除隐通道，往往需要更改系统设计或者实现的方式。改变的内容包括：消除潜在隐蔽通信参与者的共享资源；消除导致隐通道的接口和机制；增加访问控制策略等。举个例子，为了消除由于共享内存引起的资源耗尽型隐通道，可以为每个安全级别的进程预先分配一定大小的内存，从而消除不同级别的进程由于共

享内存产生的隐通道。

4.3.2 带宽限制法

当消除隐通道的方法不适合时，可以考虑限制隐通道的带宽。带宽限制的主要思想是设法降低通道的最大或者平均带宽，使之降低到一个事先预定的可接受的程度。具体的方法有：增加每次传递一个符号的时间；增加信道噪音；限制每次可以传递的信息量等。

4.3.3 审计法

为了防止对隐通道的滥用，可以对隐通道的使用进行审计，以便及时地发现非法的使用和进行必要的处置。审计要求包括：能够确定哪一个（或者一类）隐通道被利用；使用隐通道的发送者和接收者；审计机制不能被绕过；很低的误报概率。然而，审计方法不太可能发现隐通道传输的具体信息，因为发送者可以使用加密的方法传输。

4.4 隐蔽流图法

隐蔽流树法能够发现其他方法检测不到的隐通道，也比较的直观。但是它的缺点是树生长得非常快，一个简单的文件系统所对应的隐蔽流树已经超过了10^4个结点。构造和遍历这样一棵庞大的隐蔽流树，不但需要消耗大量的 CPU 和内存资源，而且也不利于隐通道的图形化表示。隐蔽流树法的另外一个缺点是必须为每个待分析的属性构造一个有向图。如果需要分析的属性很多，那就必须构造不同的隐蔽流树。事实上，属性之间往往是相互关联的，不同属性构造出来的隐蔽流树也有很多的重复之处。针对这些问题，本节提出了隐蔽流图法。隐蔽流图法采用的是图式结构，整个系统共用一张隐蔽流图，在构造方法上比隐蔽流树法更为简单直观。

4.4.1 隐蔽流图的构造

隐蔽流图的构造所需的信息与隐蔽流树的构造所需的信息是一样的，每个操作都需要创建三个列表：引用列表、修改列表和返回列表。引用列表是指在操作执行的过程中所有值被引用的属性；修改列表是指在操作执行的过程中所

有值被修改的属性；返回列表是指所有值被操作返回的属性。这里提到的属性均指函数间共享的变量，因为分析一个函数内部变量之间的隐蔽流是没有意义的，它不会被用于真正的隐通道。一旦把系统中所有原语操作的引用、修改和返回列表准备完成后，就可以开始构造系统的隐蔽流图。隐蔽流图法不仅可以在系统设计的时候使用，在开发、测试的时候同样可以使用。这是由于三个列表中包含的信息在软件工程生命周期的所有阶段中都是可用的。

下面通过一个简单的示例来说明隐蔽流图的构建方法。首先给出原语操作及其它们的三个列表：

操作 1. 引用列表：A；　修改列表：C，D；　返回列表：NULL；

操作 2. 引用列表：D；　修改列表：C；　返回列表：NULL；

操作 3. 引用列表：C；　修改列表：NULL；　返回列表：C；

操作 4. 引用列表：D；　修改列表：B，C；　返回列表：D；

由以上这些信息构建的隐蔽流图如下：

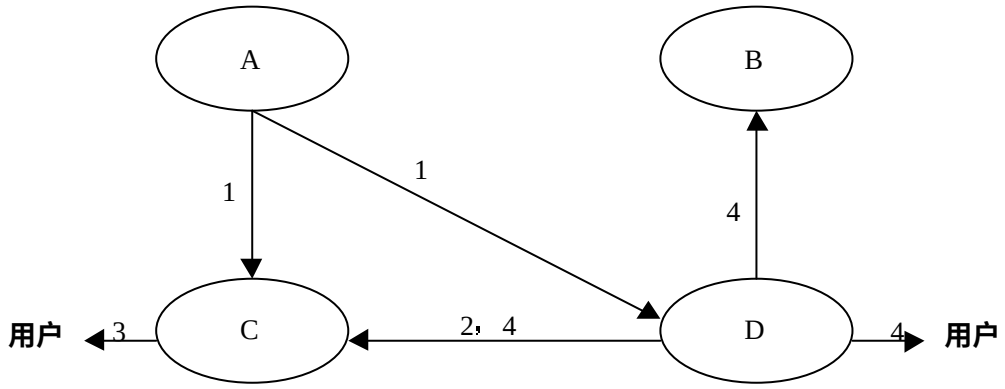

图 4.1 根据条件构造的隐蔽流图

图中的顶点表示被引用、修改或返回给用户的属性，有向边表示属性之间的隐蔽流，有向边上的标识符表示所有可能引起这个隐蔽流的操作。如果一个操作引用了属性 A 并且修改的属性 B，那么它将引起从属性 A 到属性 B 的隐蔽流。此外，如果某个操作在它的返回列表中包含了属性 A，那么它将引起从属性 A 到用户的隐蔽流。从上图可以看出，隐蔽流图能够清楚地描述由于原语操作引起的属性之间的信息流。由图可以得知：属性 C 和 D 可以被直接识别，属性 A 能够被间接识别，属性 B 无法识别。

由于一个属性可以被用作隐通道当且仅当可以被引用和修改，因此图 4.1 中只有那些入度和出度都不为 0 的顶点所对应的属性可以被用作隐通道。换句话说，属性 A 和 B 都不能被用作隐通道。删除它们所对应的顶点和指向这些顶点的有向边，保留从这些顶点发出的有向边。简化后的隐蔽流图将被用于搜索隐通道的实际应用场景。

4.4.2 隐蔽流图的搜索

隐蔽流图的搜索的思想就是为每个可能被用作隐通道的属性创建两个列表：第一个列表是由可以修改当前属性的操作组成，称为属性的修改列表；第二个列表是由支持对当前属性直接或间接识别的操作序列组成，称为属性的识别列表。然后通过将两个列表中的操作或操作序列组合，用户就可以构造出一个潜在的隐通道。

属性的修改列表可以通过搜索所有指向该属性所对应顶点的有向边来构建。搜集这些有向边上的标识符，并合并其中的重复操作。例如，图 4.1 中的属性 C 的修改列表为{1，2，4}。为了创建属性的识别列表，从该属性所对应的顶点出发，搜索所有可能到达用户的路径。具体的搜索方法可以采用深度优先搜索或者广度优先搜索。由于有向图中可能存在环，为了避免搜索路径无穷尽地循环下去，可以约定每个属性在搜索路径中重复的最大次数。

4.5 小结

本章主要介绍了隐通道在不同标准下的分类，隐通道的各种标识方法以及标识隐通道后的处理策略。提出了隐蔽流图法，并且简要描述了隐蔽流图的搜索策略。隐蔽流图法集合了信息流法和隐蔽流树法的优点，可以在软件工程的各个阶段使用，具有良好的应用前景。

第五章 原型系统的设计与实现

为了获得数据库系统的安全性，一种方法是在数据库的前端编写附加的软件进行安全保护。它适合于某些特殊应用的需求，但其通用性和安全性都比较差。因此，为了满足对信息系统安全性的要求，就必须在 DBMS 内部实现各种安全机制，也即实现安全数据库系统。

前面对 ESW 安全模型的完整性规则、读写规则进行了研究，证明了它的正确性、完备性和安全性，并研究了推理通道和隐通道的控制问题。为了检验这些理论的正确性，作者设计了一个安全数据库原型系统并实现了部分模块。下面主要介绍与安全性紧密相关的访问控制模块、推理控制模块和隐通道控制模块的设计和实现过程，其它模块将在以后进一步的研究。

5.1 原型系统的体系结构

作者设计的原型系统采用可信主体 DBMS 体系结构，如图 5.1 所示。采用这种体系结构的原因是它不受操作系统进程数的限制，并且比较容易实现。SYBASE 就是采用这种体系结构实现了一个在安全 UNIX 系统上的 B1 级别的安全数据库管理系统。

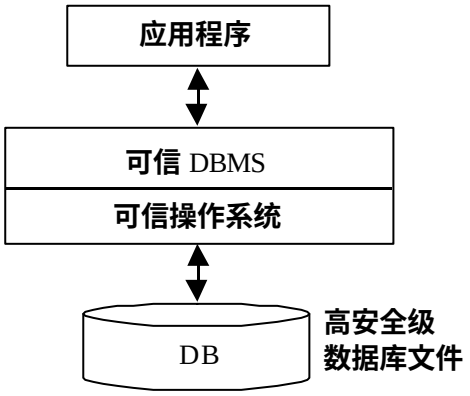

图 5.1 原型系统体系结构

5.2 原型系统的安全功能模块划分

DBMS 的实现涉及技术众多，如查询优化、多重索引、备份、审计等。但该原型系统主要是检测前面所提出的理论的正确性，因此在分析和设计时把重点放在与安全性相关的模块上。

当用户登录的时候，需要对用户的身份进行认证。登录成功以后，用户可以发出访问请求。来自用户的访问请求，经过 SQL 解析后传递给自主访问控制模块。经自主访问控制模块检查，确定其是否有访问权限。如果有，记录审计信息后进入强制访问控制模块。强制访问控制模块进行检查，检查通过后记录审计信息，然后进行推理通道模块的分析和隐通道模块的分析，确保不存在高安全级信息向未授权的低安全级主体的流动和未授权的低安全级主体对高安全级信息的推理。检查完毕，记录审计信息。这时候用户的访问请求才被允许执行，对数据库进行操作。不管对数据库的操作结果如何，审计模块一如既往地进行审计记录。原型系统功能模块如图 5.2。

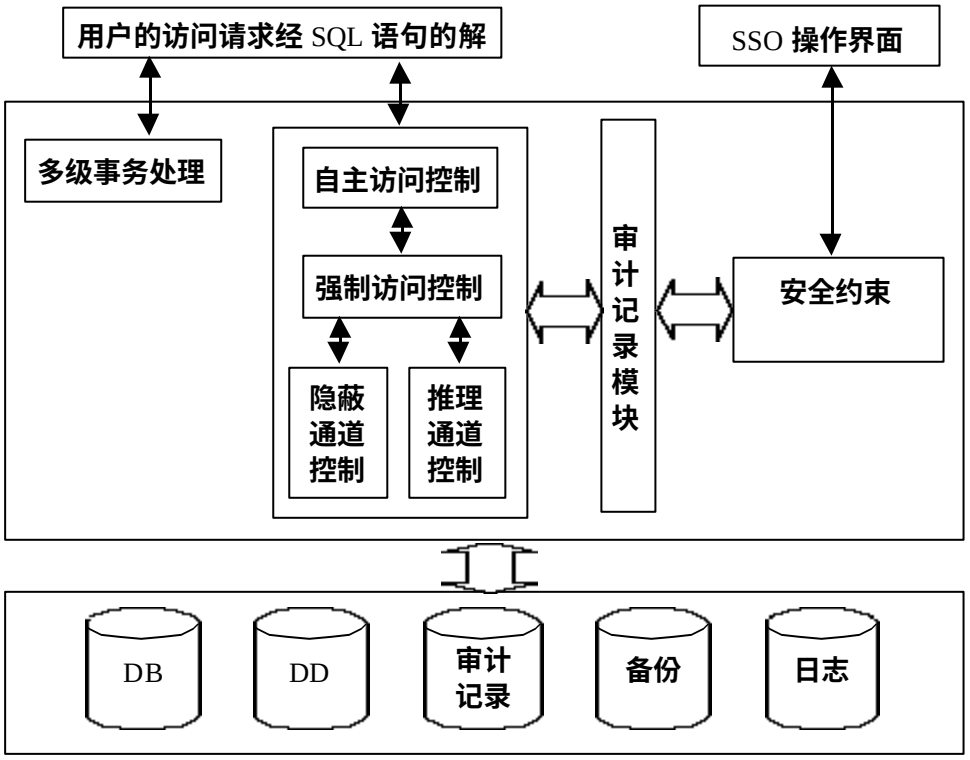

图 5.2 原型系统功能模块

5.3 访问控制模块

访问控制分自主访问控制（DAC）和强制访问控制（MAC）。现在多数安全数据库管理系统在支持强制访问控制的同时，也支持自主访问控制。两者各有优缺点，将两者结合正好可以达到优势互补的效果。

5.3.1 自主访问控制模块

自主访问控制模块实现的是检查用户是否对表拥有相应的操作权限（如插入、删除记录）。如果有，则通过自主访问控制检查，用户的访问请求被传递给强制访问控制模块作进一步检查；如果没有，访问请求被拒绝。自主访问控制由系统管理员（DBA）负责，通过 GRANT 和 REVOKE 两个命令给用户授权和回收权限。本模块采用的是“封闭世界”理论，即只有有明确的授权，用户才可以访问相应的对象。

用户表一级的操作有：CREATE 和 DROP；记录一级的操作包括：SELECT、INSERT、DELETE、UPLEVEL 和 UPDATE 等。当用户提出访问请求的时候，自主访问模块查看用户权限表来判断用户是否具有对指定表的操作权限。DBMS 启动的时候自动加载用户权限表，该表中每条记录表示一个用户对一个表具有的权限，每条记录定义如下：

```
STRUCT User_DAC
{
    INT Num;                    //记录号
    CHAR User_ID[64];           //用户名，最大长度为 64
    CHAR Tab_Name[64];          //表名，最大长度为 64
    INT Permission;             //用户权限,每位分别表示用户是否
                                //具有相应的权限
};
```

自主访问控制模块又分为几个小模块，其结构和机制如图 5.3。

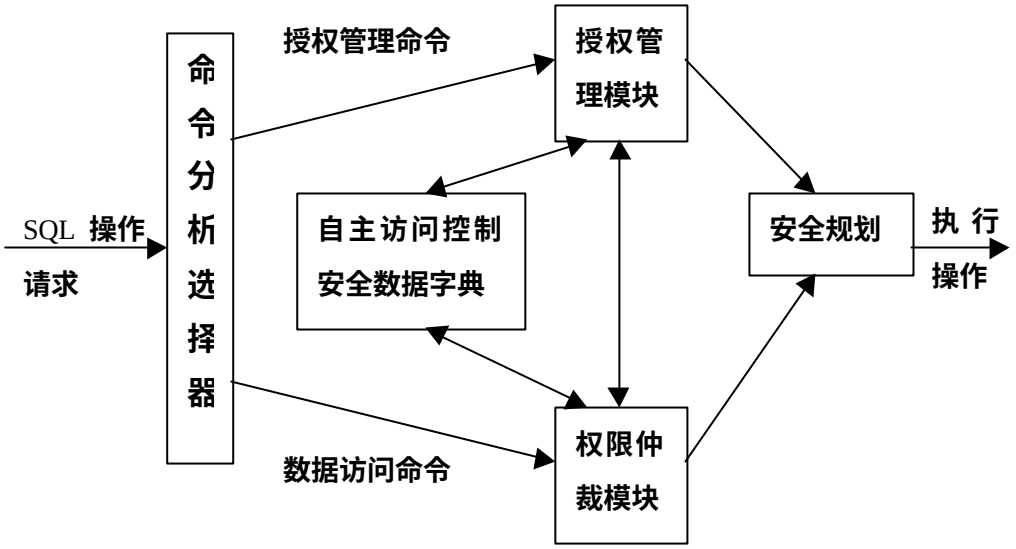

图 5.3 自主访问控制模块机制图

5.3.2 强制访问控制模块

强制访问控制是通过用户无法回避的存取请求来防止各种直接和间接的攻击。强制访问控制策略为每个主体和客体都分配一个分层的密级和不分层的范围，通过比较密级和范围来确定主体对客体是否具有相应的操作权限，详细叙述见第一章。本文定义的安全模型 ESW 是半元组级模型，其数据结构定义如下：

```
STRUCT Subject_type{                    //主体的数据结构
  CHAR  DBName[128];                    //数据库名
   CHAR UserName[64];                   //用户名
   SHORT INT Level;                     //主体的安全密级
};
STRUCT Attr_type {                      //表中属性的数据结构
   CHAR TableName[64];                  //表名
   CHAR AttrName[128];                  //属性名
  e_num  Datatype  Attrtype;            //属性的数据类型
   STRUCT Attr_type *next;              //指向下一个属性的指针
};
STRUCT TupleAttr_type1 {                //元组中属性值的结构定义
```

```
    CHAR TableName[64];              //表名
    CHAR AttrName[128];              //属性名
   e_num  Datatype  Attrtype;        //属性的数据类型
    void *pValue;                    //指向属性值的指针，根据属性的
                                     //数据类型分配大小
    STRUCT TupleAttr_type *next;     //指向下一个属性的指针
};
STRUCT  Table_type  {                //表的数据结构
    CHAR TableName[64];              //表名
    INT AttrNumber;                  //表中的属性的个数
    STRUCT Attr_type *first;
};
STRUCT Tuple_type  {
    CHAR TableName[64];              //表名
    SHORT  INT    Tuple_level;       //前一个字节是元组的安全密级，
                                     //后一个为关键字的安全密级
    TupleAttr_type *first;           //指向元组第一个属性值
};
STRUCT  Level_type{                  //安全密级的数据结构
    INT Level;                       //不大于 15
    CHAR LevelName[128];             //安全密级名
};
STRUCT Category_type{                //非分层的范围的数据结构
    INT Category;                    //不大于 100
    CHAR CategName[128];             //非分层的范围名
};
```

当通过了自主访问控制模块的操作请求送到强制访问控制模块后，对该操作进行访问控制检查，如果通过，送到推理通道控制模块和隐通道控制模块进行进一步检查，同时审计模块进行审计记录，然后才能对数据库进行操作。如果通不过，请求被拒绝。

这里需要说明三点，一是主体安全级中的范围，一般在设计阶段确定后就

不再变动，所以讨论的时候，一般用密级代替安全级。二是这里讨论的所有主体，均是不可信主体，对于可信主体，可以进行特权操作，这里对于可信主体不作进一步讨论。三是关键字的密级和元组的密级都定义在元组结构的一个变量中。

5.4 推理通道控制模块

用户的操作请求经过自主访问控制模块和强制访问控制模块检查之后，被送到隐通道控制模块和推理通道控制模块。关于推理模块，一般又分为推理检测模块和推理控制模块，如图 5.4 所示。首先在数据库设计阶段由推理通道检测模块找出推理通道。在数据库运行阶段，则检测用户所访问的对象所持有的密钥是否高于用户的安全级。如果比用户的安全级高，则拒绝访问；否则就提交用户的操作请求并把执行结果返回给用户。

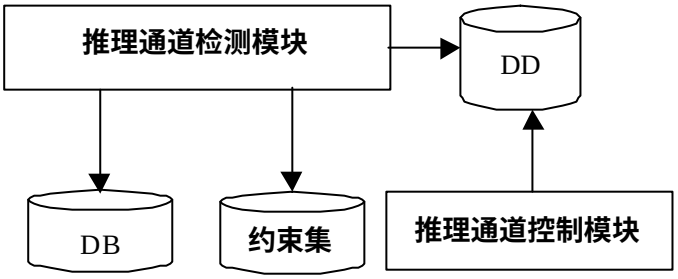

图 5.4 推理控制模块内部结构图

推理通道检测模块工作在数据库的设计阶段和启动阶段。在设计数据库表的时候，检测出表中的函数依赖和多值依赖关系，对于这两种依赖关系，处理方法不同，这里主要对函数依赖进行了讨论。在原型系统中函数依赖关系表示如下：

```
STRUCT Function_Constraints
{
    INT Num;                        //依赖关系编号
    LeftType left;                  //“ ”的左边的变量
    RightType right;                //右边的变量或者是常量
    INT selectionFlag;              //是否有选择条件标志
    CHAR selection_control[128];    //选择条件，该函数依赖适用的范
```

围

};

对于这种函数依赖，我们采用的方法是在数据库设计阶段尽量消除这种推理通道。这种方法虽然降低了数据的可用性，但大大地增强系统的安全性。如果发现提升安全级的做法已经不满足数据可用性的需要了，那么加以标记，留作运行时动态推理控制。两个模块的工作过程介绍如下：

（1）推理检测模块的工作过程

在数据库的启动阶段，推理检测模块检查数据库中是否有新增加的表，如果有则检查表中的复合推理通道，并把它们记录到数据字典中。

下面的这段代码就是检测复合推理通道并把它们记录到数据字典中。

```
INT k=0;
for(INT i=0;i<100;i++)
 {
  Path[k].num=1;
  Path[k]→head.Tablename= Ference_Cons[i].Ferencing_tablename;
  Path[k]   head.Attrname= Ference_Cons[i].Ferencing_attrname;
  Path[k]   head   next=NULL;
  for(INT j=i+1;j<100;j++)
    {
      INT  m=i;
      /*把该节点外键所指向的节点找出来*/
      if(Ference_Cons[m].Ferenced_tablename==Ference_Cons[j].Ferencin
      g_tablename
      &&
      Ference_Cons[m].Ferenced_attrname==Ference_Cons[j].Ferencing_at
      trname )
        {
          m=j;
          Path[k].num++;
          /*在该路径中增加一个节点*/
          Ference_Pathnodes *p=new Ference_Pathnodes;
```

```
            p.tablename= Ference_Cons[j].Ferencing_tablename;
            p.attrname= Ference_Cons[j].Ferencing_attrname;
            p    next=NULL;
            /*并把该节点附加到路径的后面*/
            Path[k]    head    next=p;
           }
  }
 /*检查路径上的第一个节点和最后一个节点是否也是外键依赖关系，
   如果不是就不是复合通道，必须清除*/
 for( INT i=0;i<100;i++)
  {
   if (Ference_Cons[i].Ferencing_tablename==Path[k]    head.Tablename
            &&
            Ference_Cons[i].Ferencing_attrname==Path[k]    head.Attrname
            &&
            Ference_Cons[i].Ferenced_tablename==p.tablename
            &&
            Ference_Cons[i].Ferenced_attrname==p.attrname)
            k++;
   else
            Path[k]    head=NULL;
       }
    }
```

（2）**推理控制模块工作过程**

推理控制模块主要是进行动态推理控制，其原理是为每个函数依赖中的属性对应一个密钥集，当用户访问的属性对应的密钥集大于用户的安全级时就允许访问，否则拒绝。

密钥集的结构定义如下：

```
STRUCT KeySet
   CHAR TableName[64];                    //表名
   CHAR AttrName[128];                    //属性名
```

```
    INT    Key[128];                    //密钥，默认推理通道的最大长度
                                        //为 128
    STRUCT KeySet *next;                //指向下一个属性的密钥集
};
```

动态控制的函数如下：

```
int DynamicControl( char AttrName[])
{
    KeySet  *p=Khead;                   //指向密钥集链表的首指针
    while(p!=NULL)
    {     if(strcmp(AttrName, p    AttrName)==0)
          {       for(int i=0;i<128;i++)
                      if(p    Key[i]!=0)        //找出最小的一个密钥
                              break;
                  if(userSec>=Key[i])
                          return  1;            //1 代表用户可以访问
                  else
                          return  0;            //0 代表用户不能访问
          }
          p=pext;
    }
    return 1;
}
```

5.5 隐通道控制模块

用户的请求通过推理通道控制模块审核后，由隐通道控制模块根据隐通道工作原理，分析用户提交操作，查询隐蔽流图，检测是否存在隐通道。如果检测出存在隐通道，那么操作请求被拒绝；否则，接收此操作请求并审计，然后访问数据库。隐通道的工作原理如图 5.5。

对于已经构建好的隐蔽流图，我们采用深度优先搜索算法来寻找潜在的隐通道。算法描述如下：

算法 5.1：**深度优先搜索算法**

输入：基于系统规范或者源代码构造的隐蔽流图（采用邻接矩阵存储表示）

输出：从某个顶点出发到达用户的所有路径

算法：DFSearch(int Start, int End, String Path) /*Start **表示待分析的属性所对应的顶点编号，** End **表示用户所对应的顶点编号，** Path **表示当前的搜索路径***/

（1）for(i=1; i<=NodeCount; i++) //NodeCount **表示隐蔽流图中顶点**
//**的数目**

（2）if NoEdgeBetween(Start, i) //**如果从** Start **到** i **没有边，则考虑**
then continue; //**下一个顶点**

（3）if Repeat(Path, i)>=REPEAT //**如果当前顶点在搜索路径中重复**
then continue; //**的次数大于等于** REPEAT，**则终止** //**该路径**

（4）Path = Path+Edge(Start, i)+Node(i);//**产生新的搜索路径**

（5）if i=End then //**到达用户所对应的顶点，即用户**
//**已经接收到发送者传过来的信息**

（6） show(Path); //**显示当前的隐通道属性识别路径**

（7）else

（8） DFSearch(i, End, Path); //**从当前顶点出发，继续搜索到达**
//**用户的路径**

使用算法 5.1 **对图** 4.1 **中的隐通道进行搜索。对于属性** D，**其识别列表为**{(D)4(User)}，(D)2(C)3(User)，(D)4(C)3(User)}；**对于属性** C，**其识别列表为**{(C)3(User)}。**其中，括号内的字母表示属性，括号间的数字表示操作。**

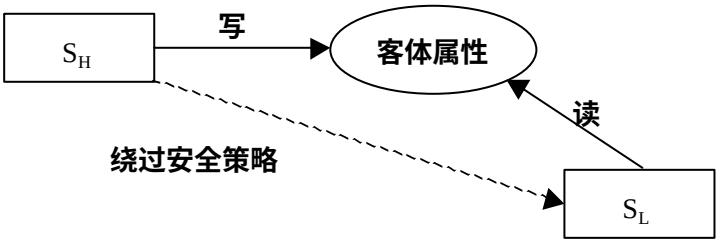

图 5.5 **隐通道工作原理**

5.6 小结

本章讨论了原型系统的体系结构和功能模块，基于 ESW 模型描述了访问控制模块；对于推理通道，主要在分析函数依赖引起的推理通道基础上提出了推理控制模块；在分析隐通道的改进分析方法隐蔽流图法的基础上描述隐通道控制模块。本文主要对这三个模块进行了研究，至于审计等其它模块，将在今后作进一步讨论。

实验证明，前面提出的安全模型、两阶段推理控制策略和隐蔽流图的算法和思想都是可实现的、可行的和正确的。

第六章 总结与展望

6.1 总结

安全数据库是数据库领域的一个重要的研究方向。随着数据库技术的广泛应用，数据库安全的研究已经成为信息安全非常重要的研究课题，其目标是保护数据以防止非法的使用造成数据的泄漏、更改和破坏。本文所做的主要工作有：

（1）针对 MLR 模型借用操作的缺陷以及 Smith-Winslett 模型操作不方便的缺点，将两者的优点进行融合，提出了在 Smith-Winslett 模型上改进的 ESW 模型。提出了该模型的四种完整性规则；并且修改和扩展了原模型的读写操作规则。该模型不仅消除了语义模糊性，同时也被证明是正确、完备和安全的。

（2）介绍了多级安全数据库系统的推理问题和检测推理通道的方法，在此基础上描述了一种两阶段推理控制策略。这种策略包含静态提升安全级和动态推理控制两部分。静态提升安全级的算法能够保证在信息易用性损耗较小的情况下控制推理通道。而在数据库运行阶段，利用动态推理控制算法可以在基本不影响访问速度的情况下切断剩余的推理通道。

（3）介绍了隐通道的分类、隐通道的标识和隐通道的处理策略等。重点介绍了隐通道的几种分析方法，对隐蔽流树法进行了改进，提出了隐蔽流图法，使检测出来的冗余隐通道大为减少。这种方法集成信息流法和隐蔽流树法的优点，具有较高的应用价值。

（4）讨论了原型系统的体系结构和功能模块，基于 ESW 模型描述了强制访问控制模块；对于推理通道，主要在分析函数依赖引起的推理通道基础上提出了推理控制模块；在分析隐通道的改进分析方法隐蔽流图法的基础上描述隐通道控制模块。

限于作者水平，文中疏忽错漏之处在所难免，欢迎各位专家批评指正。

6.2 展望

随着技术的发展，数据库安全的研究也表现出若干新的研究方向。比如基于角色的访问控制（Role-based Access Control，RBAC）、面向对象的数据库管理系统（OODBMS）、基于移动网络的安全数据库等。

随着网络技术的飞速发展，数据库安全变得越来越复杂，其可生存性也是一个尚未开发的领域。如何在受到信息攻击的时候依然保证数据库的正常运行将是研究的一个热点。

多级事务的研究目前尚未达到可应用于产品的程度，特别是移动环境下多级事务的处理研究仍处在初级阶段，有待于进一步的探索。

数据库加密技术是数据库的最后一道防线，如何保证加密后数据库的易用性、完整性等问题永远是值得深入研究的领域。

随着人工智能技术和数据挖掘技术的发展，控制推理通道也变得越来越困难。基于用户历史的查询将随着历史记录的不断增加以及用户数的上升变得不可行。前面提到的动态推理控制也存在一些弱点。因此，推理问题必将是未来研究工作的一个热点。

参考文献

[1] S. Jajodia, Database Security and Privacy, ACM Computing Surveys, 50th anniversary commemorative issue, 1996, 28(1):129~131

[2] 刘启原，刘怡，数据库与信息系统的安全，北京，科学出版社, 2000

[3] K. F. Brewster, Discretionary Access Control Issues in High Assurance Secure Database Management Systems, National Computer Security Center, NCSC Technical Report-005.5(5), 1996

[4] D. G. Marks, A. Motro, S. Jajodia, Enhancing the Controlled Disclosure of Sensitive Information, Proceedings of the European Symposium on Computer Security, Rome, Italy, 1996:290~303

[5] National Academy Press, Multilevel Data Management Security, Washington D.C., National Academy Press, 1983:143~150

[6] 张敏，徐震，冯登国，数据库安全，北京，科学出版社, 2005

[7] 朱虹，多级安全数据库管理系统研究, [博士学位论文]，武汉，华中科技大学, 2001

[8] T. F. Lunt, The SeaView Security Model, IEEE Transactions on Software Engineering, 1990, 16(6):121~145

[9] C. J. Date, An Introduction to Database Systems, Massachusetts, Addison-Wesley Longman Publishing Co., 1986

[10] D. E. Bell, L. J. LaPadula, Secure Computer System: Unified Exposition and Multics Interpretation, Bedford, The MITRE Corp, 1976

[11] P. D. Stachour, B. Thuraisingham, Design of LDV: A Multilevel Secure Relational Database Management System, IEEE Transactions on Knowledge and Data Engineering, 1990, 2(2):190~209

[12] S. Jajodia, R. Sanddu, Toward a Multilevel Secure Relational Data Model, International Conference on Management of Data, New York, ACM, 1991:50~59

[13] R. Sandhu, F. Chen, The Multilevel Relational(MLR) Data Model, ACM Transactions on Information and Systems Security, 1998,1(1):93~132

[14] 萨师煊，王珊，数据库系统概论，北京，高等教育出版社, 2000:52~55

[15] K. P. Smith, B. T. Blaustein, S. Jajodia, et al., Correctness Criteria for Multilevel Secure Transactions, IEEE Transactions on Knowledge and Data Engineering, 1996, 8(1):32~45

[16] National Computer Security Center, DoD 5200.28-STD, Trusted Computer Security Evaluation Criteria(TCSEC), Washington D.C., US Department of Defense, 1985

[17] National Computer Security Center, NCSC-TG021, Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria, Washington D.C., US Department of Defense, 1991

[18] **中华人民共和国国家标准**, GB17859-1999, **计算机信息系统安全保护等级划分准则**, **北京**, **中国标准出版社**, 1999

[19] **中华人民共和国公安部**, GA/T-389-2002, **计算机信息系统安全等级保护数据库管理系统技术要求**, **北京**, **中国标准出版社**, 2002

[20] S. Jajodia, C. Meadows, Inference Problems in Multilevel Secure Database Management Systems, Calif., IEEE Computer Society Press, 1995:570~584

[21] T. Su, G. Ozsoyoglu, Controlling FD and MVD Inferences in Multilevel Relational Database System, IEEE Transactions on Knowledge and Data Engineering, 1991,3(4):474~485

[22] A. Brodsky, C. Farkas, S. Jajodia, Secure Databases: Constraints, Inference Channels and Monitoring Disclosures, IEEE Trans on Knowledge and Data Engingeering, 2000,12(6):900~919

[23] R. W. Yip, K. N. Levitt, Data Level Inference Detection in Database System, Proceeding of 11th IEEE Computer Security Foundation Workshop, USA, IEEE Computer Society Press, 1998:179~189

[24] **王杰**, **高安全级数据库管理系统保护技术的研究与实现**, [**硕士学位论文**], **南京**, **南京航空航天大学**, 2005

[25] **崔宾阁**, **安全数据库系统隐通道分析及相关技术研究**, [**博士学位论文**], **哈尔滨**, **哈尔滨工程大学**, 2006

[26] **武立福**, **毛宇光**, **一种改进的多级安全关系数据模型**, **计算机应用**, 2003, 23(7):103~105

[27] C. E. Landwehr, Formal Models for Computer Security, ACM Computing Surveys, 1981, 13(3): 247~278

[28] D. E. Denning, Cryptography and Data Security, Boston, Addison-Wesley Longman

Publishing Co., 1982

[29] D. E. Denning, A Lattice Mode of Secure Information Flow, Communication of the ACM, 1976, 19(5):236~243

[30] 袁晓东，冯颖，B1 级数据库管理系统强制存取控制模型研究，计算机学报，2000, 23(10):1096~1101

[31] K. Smith, M. Winslett, Entity Modeling in the MLS Relational Model, Proceeding of the 18th VLDB Conference, Canada, VLDB, 1992:199~210

[32] 李立新，安全数据库及其应用系统研究, [博士学位论文]，重庆，重庆大学, 2001

[33] TZong, G. Ozsoyoglu，Controlling FD and MVD Inferences in Multilevel Relational Database System, IEEE Trans，On Knowledge and Data Engineering, 1991, 3(4):474~485

[34] M. Morgenstern, Controlling Logical Inference in Multilevel Database Systems, Proc. of the IEEE Symp. On Security and Privacy, USA, ACM, 1998:245~255

[35] 严和平，汪卫，施伯乐，安全数据库的推理控制，软件学报, 2006, 17(4):750~758

[36] K. Kenthapadi, N. Mishra, K. Nissim, Simulatable Auditing, In: ACM Symp. On Principles of Database Systems, USA, ACM, 2005:118~127

[37] 翟志刚，多级安全关系数据库管理系统的研究与实现，[硕士学位论文]，南京，南京航空航天大学, 2006

[38] X. Chen, R. Wei, A Dynamic Method for Handling the Inference Problem in Multilevel Secure Databases，International Conference on Information Technology: Coding and Computing, Canada, 2005:751~756

[39] National Computer Security Center，NCSC-TG-030, A Guide to Understanding Covert Channel Analysis of Trusted System, Washington D.C., US Department of Defense, 1993

[40] S. B. Lipner, A Comment on the Confinement Problem, Operating Systems Review, 1975, 9(5):192~196

[41] M. D. Schroeder, D. D. Clark, J. H. Saltzer, The Multics Kernel Design Project, In: Proceedings of the 6th ACM Symposium on Operating Systems Principles, USA, ACM, 1997:43~56

[42] J. C. Wray, An Analysis of Covert Timing Channels, In: Proceedings of the IEEE Symposimum on Research in Security and Privacy, California, 1991:2~7

[43] 刘文清，陈喆，韩乃平，隐蔽通道标识与处理，计算机工程, 2006, 32(8):1~3

[44] J. A. Goguen, J. Meseguer, Security Policies and Security Models, In:Proceedings of the

IEEE Symposium on Security and Privacy, California, 1982:11~20

[45] R. A. Kemmerer, Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels, ACM Transactions on Computer Systems, 1981, 1(3):256~277

[46] D. Denning, A Lattice Model of Secure Information Flow, Communication of the ACM, 1976, 19(5):236~250

[47] P. A. Porras, R. A. Kemmerer, Covert Flow Trees: A Technique for Identifying and Analyzing Covert Storage Channels, In:1991 IEEE Computer Society Symposium on Research is Security and Privacy, California, 1991:36~51

致　谢

首先衷心感谢导师毛宇光副教授在这两年多来在学习、工作和生活上给予我的悉心指导、热情关心和无私帮助。毛老师教学的认真、治学的严谨给我留下了深刻印象，所学到的治学态度将使我终身受益。在这里向我的导师致以最诚挚的敬意和最衷心的感谢。

感谢王建东教授、朱朝晖教授、曹子宁副教授、周勇博士等各位老师在学习上和生活上的关怀。各位老师的学识和勤勉的工作作风是我学习的榜样。

感谢吕丽、吴再达、张浩庭等同学，感谢室友王辉、周杰、王文超，作为我的同学，他们在学习和生活上都给予了我很大的帮助，在与他们的讨论当中，我学习到了很多专业和非专业的知识。另外感谢我的师兄翟志刚，他在学术上给了我极大指导，帮助了我的成长。

感谢我的父母和所有的亲人，他们对我的支持是我不断前进的动力。

本文参考了论文后开列的大量文献，受到很大教益，谨此对文献的作者深表谢忱。

最后，向所有给予我关心和帮助的师长、同学和朋友们致以最真诚的谢意！

在学期间的研究成果及发表的学术论文

1. 在学期间参加的研究项目

(1) 2006 年 3 月-2007 年 9 月：数据库中的不完全信息与不一致信息研究，计算机软件新技术国家重点实验室（南京大学）基金课题；

(2) 2007 年 7 月-2007 年 12 月：数据挖掘的逻辑基础问题研究，计算机软件新技术国家重点实验室（南京大学）基金课题。

2. 在学期间发表的学术论文

[1] 顾海星，毛宇光，动态推理控制，计算机技术与发展，2008, 18(1):30~32

[2] 顾海星，毛宇光，安全数据库的推理控制，南京航空航天大学第九届研究生学术会议论文集（光盘），南京，2007