第 6 章 数据库安全

6.1 数据库简介

数据库是由“数据”和一组“规则”组成的一个集合，这些规则通过指定数据之间一定的关系来组织数据。用户通过这些规则说明数据的逻辑格式，而数据实际上是存放在计算机硬盘(或其它存储部件)中，但用户不必关心它们的精确物理地址或物理格式。通常，由数据库管理员定义组织数据的规则，分配用户的权限，即每个用户可以访问哪些数据。而用户通过数据库管理系统或应用程序与数据库交换信息。

数据库文件(或表)由一些记录组成，每个记录包含一组相关的数据。表 6.1 给出了一个公司职员关系表，在这张表中，每个记录包括了职员的工作证号、磁卡号、姓名、性别、年龄和工资，它们分别称为字段或数据项。因此，一般情况下的一个数据库可以看作是一张二维表格，其中的一个记录占一行，一个属性占一列。

表 6.1 公司职员关系表

工作证号	磁卡号	姓 名	性 别	年 龄	工 资
921501	10083	张 明	男	38	1 500
921502	10089	吴晓莉	女	42	1 300
921503	10100	王新刚	男	26	900
…	…	…	…	…	…

大多数的数据库都是由多个表组成的，这是由于在实际应用中，事务之间存在着众多的关系，它们之间既有联系也有区别，因此难以将它们凑到同一张表格之中。而分成若干张表之后，可以更清晰地表示它们之间的关系。

数据库的逻辑结构称为模式，一个特定的用户可能只访问其中的一部分，称为子模式。表 6.2 就是表 6.1 的一个子模式。

表 6.2　子模式

姓　名	工　资
张　明	1 500
吴晓莉	1 300
王新刚	900
…	…

属性是数据库各列名字的统称，关系就是一组列，即由若干属性构成的。例如：利用表 6.1 的数据库，通过取姓名、工资这两列，形成了一个关系，即表 6.2。显然，一个数据库中存在着许多不同的关系。

用户通过读取、修改、增添或删除数据库的记录或字段的命令，与数据库管理程序交互信息。一个命令称为一个查询，数据库管理系统有关于查询的精确语法规则。多数数据库系统的查询语言都使用类英语的术语，便于用户操作。

执行一次查询的结果就是一个子模式。形成数据库的一个子模式的一种方式是通过选取满足某些条件的一些记录。例如：选取公司的男职员会产生表 6.3 的结果；选取工资超过 1 000 元且是男性的职员则会产生表 6.4 的结果。

表 6.3　子模式

工作证号	磁卡号	姓　名	性　别	年　龄	工　资
921501	10083	张　明	男	38	1 500
921503	10100	王新刚	男	26	900
…	…	…	…	…	…

表 6.4　子模式

工作证号	磁卡号	姓　名	性　别	年　龄	工　资
921501	10083	张　明	男	38	1 500
…	…	…	…	…	…

在通过查询选出了数据库中的一些记录之后，也可以将这些记录“投影”到一个或多个属性上。读取操作是从数据库中选出一些行，而投影操作是从数据库中抽取某些列。一次读取——投影操作的结果是得到所选记录的指定的一组属性值。例如：从表 6.1 中选取工资超过 1 000 元的记录，并把结果投影到工作证号与姓名的属性上，则得到表 6.5的结果。

表 6.5　子模式

工作证号	姓　　名
921501	张　明
921502	吴晓莉
…	…

在数据库中，还可以通过连接操作合并两张表。例如：两个子模式可以在一个公共属性上通过连接查询而合并，连接查询的结果仍然是一个子模式，它的记录是由两张表中具有共同属性值的记录合并而成。如表 6.2 和表 6.5 通过姓名属性连接后，得到一个新的子模式如表 6.6。

表 6.6　子模式

工　　资	工作证号
1 500	921501
1 300	921502
…	…

数据库中存放着大量的数据资料，可以提供查询检索服务，极大地方便了用户。一般说来，数据库具有下述优点：

(1)数据共享　众多用户都可以使用数据库中的原始数据。在网络上，异地用户通过网络也同样方便地访问数据库。

(2)数据的完整性　数据库中的数据是正确的和准确的。

(3)数据的一致性　对数据项的一个改变将不影响使用该数据的任何其他用户。

(4)最小冗余度　提高了数据的使用效率，减少了存储空间。

(5)访问控制　只有授权，用户才能访问或修改数据值。

6.2　数据库的安全需求

6.2.1　影响数据库安全性的因素

当前，一方面是数据库系统规模大，即存储的数据量多，数据处理的种类也多。另一方面是开放性不断增强，即一个数据库系统可以为许多不同类别的用户提供服务；同时，很多数据库系统还具有分布式的特点。这些都体现了数据库的复杂性，而复杂性又直接影响数据库的安全保密。

1. 影响数据库安全性的主要因素

(1)数据库数据量大，数据敏感度不同而且粒度很细　由于数据的性质不同，使得

各自的安全性要求即敏感程度也不同，从而造成在实施安全性保护时必须针对数据各自的敏感度而采取不同的保护措施，根据这种敏感度划分出来的数据区域又可能是很细碎的。例如：一个公司职员数据库由若干条职员记录构成，职员记录中姓名、年龄、住址、电话、工作部门等信息可以是分开的，而工资项因为敏感度高，不能公开，因而需要进行适当的保护；此外，对一些特殊职员，他们的住址、电话也需要保护，而工资还需要得到更高一级的保护……。这个例子说明了在数据库安全保密中的"细粒度"问题。

(2)用户成分复杂　数据库安全保密主要是针对那些较开放的数据库，这类数据库为各种各样的众多用户提供服务，尽管用户可能有不同的使用目的。应该根据用户的性质确定他们的使用权限，保证某些数据限定只有某些用户才能访问。这便造成了数据库中针对用户的数据隔离和数据共享问题。

(3)数据操作的种类多，安全性要求也不同　一般说来，对数据库中的数据可进行读出、写入、添加、删除、转移、拷贝等多种操作，而在具有安全性要求的数据库中，用户对同一数据的操作权限可能就不同，比如可读但不可写，可添加但不可删除等。这种问题一般采用授权管理来解决。

(4)推导控制问题　在数据库中，统计工作经常是不可避免的，往往统计结果的敏感度较低，但是其中一些原始数据的敏感度却很高。人们能够利用统计结果推导出这些高敏感度的原始数据，这就危及了信息的安全。因此，需要解决防止从统计信息推导出敏感信息的问题。

(5)分布式数据库带来更多的安全性问题　例如：数据一致性问题、并发控制问题、用户登录以及信息传输问题，等等。随着计算机网络日益普及和发展，分布式数据库得到了越来越广泛的应用，它的安全问题也愈发得到关注。

(6)信息存储和处理的高效率是数据库的主要优点　我们在考虑安全性时，要尽可能少地影响原数据库系统的效率。

以上这些特点，决定了数据库安全问题不能用以往的硬件物理安全或操作系统安全简单地代替解决。因为，它们最多只能提供对整个系统或整个数据文件的保护，而不能解决数据库中多层次、多方面的安全性问题。

2. 数据库自身安全要求

由数据库自身的特点，决定了数据库在安全保密方面有其自身的要求。

(1)物理完整性与逻辑保密性　任何主体(包括人和程序等)在非授权情况下均不能直接访问数据库中的数据，从而保证整个系统结构的完整。

(2)推导控制　防止任何主体通过执行某些数据操作或运算，如多点间的信息交换或多次查询结果的交叠等，推断得到非授权信息。

(3)灵活性　在各种要求层次上，合法的授权用户均能很方便地获得相应的操作特权。

(4)高效性　数据库安全机制要尽可能少地降低数据库基本操作的效率，同时应满足用户的需求。

(5)合理性　用户可存取的数据不能因安全机制的引入而减少，安全机制在时间和空间上的开销要合理。

由上面可以看出，理想的安全保密数据库的实现需要综合考虑各方面的因素。

6.2.2 安全数据库系统模型

基于数据库的安全需求准则，可以建立一个简单的安全数据库系统模型(见图6.1)，模型共分四层：系统层、函数层、接口层和应用层。

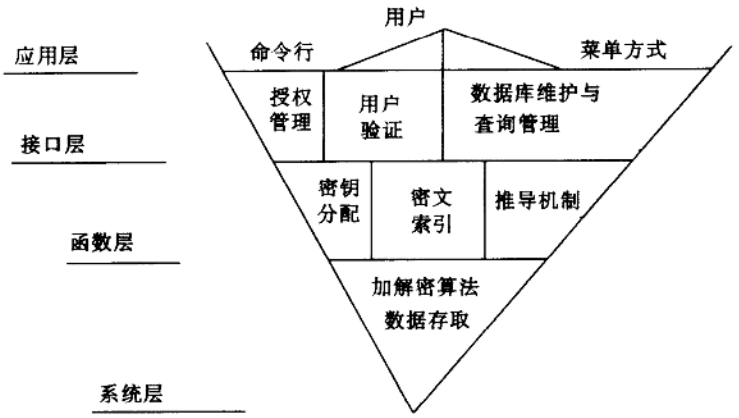

图6.1 安全数据库系统模型

(1)系统层 系统层包括数据存取和加密、解密算法。在系统层，数据库的敏感数据可以按密文的形式存放，这样能降低安全管理对系统的依赖程度，大大提高系统抗击非法入侵者绕过系统安全机制直接访问数据的能力。当然，数据以密文形式存放之后，每次访问它们时，都要进行加密和解密操作，将会增加时间开销；特别地，对密文数据的查询至今尚未找到十分有效的方法。因此，安全数据库中，密文数据量的选择要有恰当的尺度。在系统层中，主要功能是完成数据库中数据的直接存取和相应的加密、解密操作，同时对数据进行一致性检查和校验。

(2)函数层 函数层包括密钥分配机制、快速索引机制和推导控制。它可以完成密钥的生成、管理与分配，完成数据库密钥共享与密文的快速索引机制，并利用推导控制防止使用合法手段获取非授权信息的攻击行为。该层决定了安全管理系统的整体性能，是整个系统的关键。

(3)接口层 接口层直接面向用户。主要包括授权管理与用户验证、数据库维护操作和查询管理功能。它在函数层的基础上建立起授权机制，并提供一套数据维护和查询操作。授权管理主要进行授权用户权限的维护，查询管理部分则主要完成用户对密文数据快速查询的功能。

(4)应用层 应用层中的用户可以通过交互方式，也可以通过命令方式进行数据库管理。在应用层中有管理员、授权用户和普通用户。管理员负责系统维护、密钥的分配管理，以及用户的授权；授权用户可以进行他所授权的数据操作和普通用户所能进行的操作；普通用户则只能进行一般的数据操作。

6.3 安全数据库研究现状

目前，数据库的安全研究已进入一个新阶段。从基础性的关于安全数据库语义的确立表达，到实际的授权机制和多级模型的建立实现，人们已经做了许多研究工作。有关数据库安全保密的实现技术和策略方法也是多种多样的。有些理论性的研究已日趋成熟实用，有些还处于原型探讨阶段，有些甚至还处于初步的设想和建议。本节主要讨论通过数据加密实现数据库安全的研究现状。

数据加密是已知的一种保护静态数据不被侵犯的很好方法。将密码技术与数据库系统有机地结合起来，把敏感数据以密文的形式存放在数据库中，无疑可以大大增强数据的安全性。同时，由于用户持有用户密钥可以访问数据库，这样数据库安全对系统本身的依赖性较小，只要用户所持密钥不泄露，即使系统在安全管理上存在某些问题，也仍然可能保证数据库中敏感数据的安全。因此，数据库加密是数据库安全保密的一个重要发展方向。不过，在实际应用中，由于数据库本身所具有的特点，对数据库进行加密并进行各项管理工作还面临着许多研究课题，诸如数据库加密算法、密钥分配管理、授权模型以及密文数据快速索引，等等。

1. 数据库加密算法的选择

加密、解密算法最初是由于保密通信的需要而产生的。在密码学中，一般认为加密、解密算法是公开的，系统的安全性依赖于密钥的保密性。评价一个加密算法主要依据两个方面：

(1)安全性　不论窃听者窃取多少密文，在不知道密钥的情况下，他均不能通过对加密、解密算法和所获密文的分析推算出相应的明文或者密钥。也就是说，加密算法必须能抵抗密文攻击。除此之外，算法还应能抵抗：明—密文攻击、选择明文攻击以及其它形式的攻击。对安全性的分析可以从理论上研究推导，也可以采用各种破译技术对算法进行攻击以直接测试其抗攻击能力。

(2)高效性　加密算法本身要易于工程上实现，运行效率高。一般说来，传统密码算法，比如 DES 在一定程度上可以满足高效性的要求；而公开密码算法，比如 RSA 往往计算量较大，运行效率较低。对于数据库中多达几十万条甚至更多的记录进行频繁的加、解密运算，如果没有高速密码算法来满足用户对响应时间的需求，那么数据库加密就会成为一句空话。

数据库中数据的加密与通信中的安全保密虽然有相似之处，但数据库自身的特点使加密算法的应用面临许多问题。

首先是加密数据块的选择。如果以整个数据库为整体进行加密，一方面容易受到对密文搜索的攻击，同时也不利于对数据库数据的更新和查询等操作。对于不需要加密的数据或敏感程度不同的数据，这种“大、一、统”的加密算法显然也不能满足要求。如果以数据库中的每个数据项为单位进行加密，不同的数据项使用不同的密钥，这会造成密钥量大，不易进行系统管理，而且密钥所占空间很大。这种粒度划分很小的做法显然

也是不适宜的。乔治(George)等人将子密钥的概念引入数据库加密系统中，并以中国剩余定理为基础构造了一个面向记录的数据安全保密模型。模型中将数据库数据加密分为三种：基于记录或块的加密、多密钥记录或块的加密和分离字段密钥的记录加密。前两种方法以记录或块为单位进行，其缺点在于进行某些数据操作时需要对不必要的数据进行加密和解密操作；第三种方法是将记录的加密与各个字段的解密结合起来，加密时以记录为单位进行，解密时可对各字段单独解密，这种方法的不足是应用面太窄。

因此，根据不同需要合理划分加密数据块，适当选择加密算法将是今后的研究重点。其中特别需要指出的是，对于数据库中的短字段，当需要对它们进行加密时，就要设计相应的短字段加密算法，以尽量减少数据库的膨胀。

其次是数据库的密钥分配共享与管理。在人们试图利用加密解决数据库安全问题的同时，随之而来的是密钥分配共享与管理问题。由于数据库用户较多，每个人的权限又各不相同，因此会有许多用户密钥；数据库中存放的数据也很多，对应的有很多加密数据的密钥。如何建立一套合理的密钥分配体制，对这些密钥进行有效的管理和保护也成为人们研究的焦点。

再次是数据库授权机制。与数据加密在数据库安全中的应用相比，人们在数据库授权机制方面的研究则要长得多，几乎在进行数据库理论研究的同时，关于建立数据库授权管理模型的讨论也已展开。已有的授权模型都是建立在关系、层次或网络数据模型上，以关系或关系属性为授权单位。对于处理简单对象，这种传统模型已较为成熟，且有其独特优点，但对于复杂对象数据管理的数据库来说，需要再进一步扩展。拉卜铁(F·Rabtti)等人提出了一种复杂对象的授权管理模型，该模型试图克服旧有授权模型的不足，但它并未能更多地反映复杂对象，比如面向对象概念的语义。

安全数据库中的安全性和高效性始终是一对人们致力解决的矛盾。如何在不降低数据查询效率的前提下提供足够的保护措施，至今仍未能找到满意的答案。

2. 推导控制技术

推导控制是专门针对信息推导问题而采取的一类控制技术。对推导控制技术的要求包括以下几点：

(1)保证所讨论的数据库的安全性。

(2)能实现各种通用的查询统计功能。

(3)将其加到现有的数据库上不会有实际困难。

(4)每次查询增加的开销不会太大。

目前，这方面的推导控制方法有三种：数据抑制法、数据伪装、查询分析法。经常采用的方法主要有数据抑制法和数据伪装。数据抑制法采用的策略是所有与敏感数据有关的推导，均不提供或有条件地提供统计结果。数据伪装则采取用抽样或随机扰动结果代替全面准确的统计结果，从而增加推导出隐含敏感信息的难度和准确性。这两种方法在增加了安全性的同时，也降低了系统的可用性，减少了用户获得的数据量。第三种推导控制方法是查询分析法，即跟踪一个用户多次查询的结果，并分析这些查询联合起来能否攻击敏感数据，可惜它开销太大，而且很难对付多人联合起来进行攻击的复杂情况。

有关数据库安全保密方面的研究还包括安全数据库语义的研究，如确定安全规则和人机界面接口条件，建立系统安全数据模型；安全保密确信度的研究，如安全测试和安全条件的验证技术等。

6.4 数据库的完整性

数据库是许多有用数据的汇集，这些数据都要防止丢失或破坏。对数据完整性的保护是最基本的安全要求。

数据库的完整性主要涉及三个方面：

(1)数据库的完整性　这涉及从总体上保护数据库，以防止由于磁盘驱动器的读写失败或数据索引的破坏等而引起的损害。这些问题应在操作系统和恢复程序中解决。

(2)元素的完整性　它要求特定数据项的值只能由授权用户写入或修改，因此要有恰当的访问控制保护数据库以防止非授权用户的破坏。

(3)元素的准确性　这涉及对数据库的元素只能写入正确的值。

因此，数据库的完整性主要是指它的内容的准确性。而由于原始数据输入出错、应用程序错误，以及滥用数据库等，都会影响数据的准确性。其中与应用程序或用户联系最密切的是元素的完整性和准确性，或者说要保证数据库运行的完整性和对并发性事务正确处理。

6.4.1 并发性事务处理造成错误

数据库之所以得到广泛应用，其最主要的优点就是把大量数据收集在一起，供广大用户共同使用。也就是说，数据库系统常常是多用户系统。因此，必须考虑两个用户同时访问同一数据库时会不会出现问题。

如果两个用户同时读取同一个数据项，他们将会得到相同的值，这不会产生矛盾。如果两个用户试图修改同一数据项，则可能会出现错误。例如：设两个事务处理 T_1 和 T_2，读出并修改数据项 A，其次序为

T_1	从表中读出	$A=50$
T_1	在内存中修改	$A:=A+50$
T_2	从表中读出	$A=50$
T_1	写回数据库	$A=100$
T_2	写内存中修改	$A:=A-50$
T_2	写回数据库	$A=0$

结果，T_1 的修改将完全丢失，数据库的完整性将受到破坏。应该说以上两个事务处理本身都是正确的，错误产生的原因只是在于从读数据库的一个值到对该值进行写修改这一段时间的延迟，而恰恰在这段延迟时间里，另一个用户访问了同一数据。

为了解决丢失修改的问题，需要进行独占封锁，即把整个查询—更新周期看作为单

个基本操作，保证数据在其修改的整个时间段内禁止被读取。

若一个事务的处理按某种时间顺序存取若干个数据，如果此时允许其它事务处理修改这些数据，也会造成不正确的结果，从而无法保证数据的正确性。以银行账户处理为例，设 T_1 进行结算，需要把数据 A、B、C 合计；T_2 进行过账，要修改这些数据，原始状态为

$A = 500,\quad B = 300,\quad C = 200$

事务处理步骤如下：

T_1：　$X := A = 500$

T_2：　$A := A - 200 = 300$

T_2：　$C := C + A = 500$

T_1：　$X := X + B = 800$

T_1：　$X := X + C = 1\ 300$

得到的合计是 1 300，而新数据库的状态是 $A = 300$, $B = 300$, $C = 500$，从而使结账发生错误。这是因为所执行的事务处理所面对的数据库状态不一致造成的。因此，无论是相关读序列还是相关的修改序列，都应该进行封锁；只有圆满完成了本事务处理过程后，才允许其它事务处理进行操作。

6.4.2 事务处理的封锁协议

在多事务处理中为了保证数据库的完整性和一致性，应当采取必要的措施，防止失控存取造成的错误。只有当一个事务处理圆满完成，而且数据库系统确认该事务处理所完成的修改具有合法性并予以接受时，数据库系统才会从一种一致状态转变为另一种新的一致状态。

保持数据库的一致性就是研究申请数据封锁、数据封锁步骤和解除数据封锁的充要条件。申请封锁的序列及解除封锁的序列称为事务处理的封锁协议。它需要满足下列条件：

(1)要存取的数据必须先封锁。

(2)对事务处理已占有的封锁，不得重复申请。

(3)一个事务处理必须注意到其它事务处理的封锁。

(4)每个事务处理分作两段：发展期和退缩期。发展期中申请封锁，退缩期中解除封锁。在退缩期不允许申请新封锁。

(5)事务处理结束后，解除全部封锁。

例如，某事务处理要涉及三个数据 A、B、C，其封锁协议可表示如下：

封锁	A	
封锁	B	
封锁	C	发展期
处理	A、B、C	
解除封锁	C	退缩期

解除封锁　　B

解除封锁　　A

把封锁及解除封锁的任务交给数据库系统，可以保证封锁协议划分成发展期和退缩期两个阶段的要求。如果用户随意申请封锁和解除封锁，或者不去申请而任意存取、修改数据，数据库系统就无法保证数据的完整性和一致性。

6.4.3　保护数据完整性的方法

为了保护数据完整性，数据库系统提供了许多有力的封锁方法。写操作要求对有关数据的存取有排它性，而读操作允许共享存取数据。但是，如果因为写操作把整个数据库封锁起来，虽然能满足要求，但却失去了并行能力，影响了数据库的功能。因此，应该只封锁那些真正需要封锁的数据，只封锁与规定的操作有关的数据。通常把这样的数据称为完整性相关域。

1. 谓词封锁

设 D'为数据库中可能的数据集合，D 为实在数据的集合，$D \subseteq D'$。为了使事务处理 T 保持一致状态，至少应该封锁 T 的完整性相关域 $I(T)$的数据，当然 $I(T) \subseteq D'$，从而使 T 中任何施加于 D'(或 D)的操作都置于保护之下。如果有一个定义于 D'的谓词 P 能够封锁 $I(T)$，使得 $I(T) \subseteq S(P)$，其中 $S(P)$是 D'中满足 P 的元素构成的子集，那么事务处理的完整性相关域包含于它的封锁区。当然，如果 $I(T) = S(P)$，则是最理想的，不过这样会使谓词过于复杂，封锁粒度太小，将造成系统开销加大。如果 $S(P)$的限定过于简单，封锁粒度过大，就会把过多的数据排斥于并行事务处理之外。

谓词封锁的主要困难是判定并发事务处理的谓词之间是否相容，即构造谓词分配算法。为此，对所有事务处理 T_1，T_2，…，T_n，必须检查下式是否成立：

$$S(P_i) \cap S(P_j) = \varnothing \qquad i \neq j$$

这个问题在一阶谓词中不可判定。为简单实用起见，可以只判定谓词 $P_i \wedge P_j$ 是否不成立，若不成立，则事务处理 T_i 和 T_j 没有冲突，可允许进行 P_i、P_j 有关的封锁。

2. 修改操作串行化

如果只是对数据库进行读操作，由于数据库状态没有发生变化，也就没有必要采取封锁措施，只有修改操作才可能危及数据库的一致性。因此，对修改操作要进行封锁。所以可采取如下策略：所有读操作可以并行访问数据库，而写操作只能排它性地一个接一个进行。这就要求调度程序引入读操作和写操作的排队，在某些时间段内，排队等候的读操作可以成批进行，而写操作只能在当前进行读操作的所有事务处理结束后再分别进行。由于写操作往往具有局部性质，所以也可以将数据库进行分割，使相应的封锁限定于一定的分割部分，从而使不同分割部分上的写操作可以并行进行。

3. 分层封锁

在保证数据库一致性的前提下，为了使事务处理所封锁的单元最好与完整性相关域一致或稍大一些，同时处理的速度更方便和快捷，可以采用分层封锁的方法。

在分层封锁策略中，根据事务处理的性质，封锁单元可以是整个数据库、各表、各

关系、各记录。它们组成一套层次体系。例如：当封锁单元是某一张表时，则此表内的所有记录和可能的记录都将被封锁。分层封锁如图 6.2 所示。在封锁层次中，对 t_i 独占封锁可有下列选择：

(1)独占封锁数据库 DB。

(2)独占封锁 S_i，但不妨碍其它事务处理存取 S_2，S_3，…，S_n。

(3)独占封锁 R_i，但不妨碍 S_2，S_3，…，S_n 以及 R_2，R_3，…，R_m。

(4)独占封锁 t_i，t_i 以外的数据库可以为其它事务处理开放。

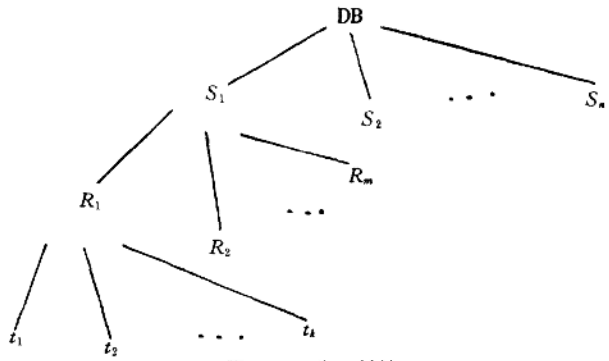

图 6.2　分层封锁

6.5 推理问题

在某些数据库中往往包含了一些敏感数据。所谓敏感数据，指的是那些不应公开的数据。如果一个数据库中的数据全部都是敏感数据，比如国防数据库，那么可以采用访问控制或全部密文处理来解决；当数据库中某些但非全部数据是敏感的，而且它们的敏感程度又不一样时，可以采用权限管理来限制用户的权限，使得非授权用户不得读、写这些数据。但是，这时用户是否能够由一些非敏感数据通过推理得出敏感数据呢?这就是推理问题，它是数据库安全性中的又一问题。

表 6.7　学生数据库

姓　名	性　别	助 学 金	处分情况	成　绩
A	N	1 000	0	85
B	Y	0	0	80
C	N	500	0	82
D	Y	500	0	83
E	N	1 000	1	80
F	Y	0	1	75
G	N	0	0	87

表 6.7 的数据库可以帮助说明推理问题。这是一个大学生的基本情况数据库，每条记录共有 5 个字段：姓名、性别(N 为女，Y 为男)、助学金、处分次数和学习成绩。假设学习成绩是敏感字段，尽管这些值只有与具体的学生相关联时才是敏感的。

1. 直接攻击

在直接攻击中，人们试图直接通过查询以得到几个记录来直接搜索并确定敏感字段的值。最成功的技术是形成一种特定的查询，它恰与一个数据项匹配。

例如，在表 6.7 中，一个敏感的查询可以是：

$$\text{List Name Where 性别} = N \wedge \text{处分次数} = 1$$

这是一个明显的攻击。因为用户可以立即得到这位女同学就是 E。

此外，还可以有其它的查询方法，比如

$$\text{List Name Where}(\text{性别} = N \wedge \text{处分次数} = 1) \vee (\text{性别} \neq N \wedge \text{助学金} = 1\,000)$$

结果仍然只有 E。

2. 间接攻击

在统计数据库中，通过一种或多种统计值，通过这些统计量谋求获得个人的敏感数据，这就称间接攻击。

(1)求和　通过求和进行攻击是试图由和数推出所需要的值。例如：对表 6.7 可以统计成绩在 80 分以上的男女学生人数情况，如表 6.8 所示。从中可以看出每个女学生的成绩都不低于 80 分，因此可以断言，任何一个女学生的成绩都在 80 分以上，这样便从另一个方面泄露了女学生的学习状况。

表 6.8　学生成绩统计表

	≥80	<80	总　计
男　生	2	1	3
女　生	4	0	4
总　计	6	1	7

(2)计数　可以把计数、求和结合起来以揭示更多的信息。数据库的这两种统计数据往往是能够给出的，以允许用户确定平均值。表 6.9 表示 80 分以上男女学生受处分的人数，这就很容易通过查询受过处分的男学生姓名 F，并知道他的成绩不足 80 分。

表 6.9　学生受处分统计表

	处分人数
男	0
女	1

(3)代数方法的攻击　采用代数方法，针对数据库的一些弱点，有可能通过一系列查询，它们返回的结果与几个不同的集合有关。例如：由下述查询组成的系统并没有完全揭示数据库中任何一个单一的 C_i 值，然而可以通过它们进行求解，从而将每个 C_i

计算出来。

$$
\begin{aligned}
q_1:&\ C_1 + C_2 + C_3 + C_4 = 350\\
q_2:&\ C_1 + C_2 \qquad + C_4 = 300\\
q_3:&\ C_1 \qquad + C_3 + C_4 = 250\\
q_4:&\ \qquad C_2 \qquad + C_4 = 150
\end{aligned}
$$

该方程组等效于

$$
\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \quad \begin{bmatrix} 350 \\ 300 \\ 250 \\ 150 \end{bmatrix}
$$

因为系数矩阵非奇异，可以求得方程组的解为

$$C_1 = 150, \qquad C_2 = 125, \qquad C_3 = 75, \qquad C_4 = 25$$

此例表明，在统计数据库内利用合法的查询，可以推断出原来不知道的具体信息。

3. 对统计推理攻击的控制

对所有统计攻击的控制是类似的，主要有两种防止推理攻击的方法：控制查询和控制数据库中的个别数据项。

要确定某给定的查询是否会泄露敏感数据是困难的。如果查询的结果并不泄露过多的信息，系统应该给出查询结果。但是，如果查询结果占据了该类信息的相当比例，则应予以警惕。例如：选取的一个人代表了所报告数据的100%，那么这个人肯定与查询相符，所以应该拒绝给出查询结果。因此，控制查询主要依据“n 个项目超过百分之 k”的原则，即如果 n 个项目的结果超过了该类结果的 $k\%$，则系统拒绝提供查询结果。

对数据项施加的两种控制是封锁和隐瞒。所谓封锁就是不提供敏感数据值，拒绝查询；所谓隐瞒就是所提供的答案接近于而决非确切的实际值。

在封锁控制中，如果系统响应查询，则提供的结果都是正确的。但是为了保证安全性，又必须制止许多响应；在隐瞒控制中，系统可以提供查询结果，但是这些结果的准确度较低。因此，这两种控制反映了安全性与准确性之间的矛盾，它们的选择取决于数据库前后文。

(1)有限制的响应封锁　n 个项目超过百分之 k 的原则使某些低频率的数据项不被显示。例如：通过查询控制可以不响应表 6.8 中对低于 80 分的男生人数的查询。但是，从表中人们仍然很容易通过推理得到其准确值。因此，当在一行或一列中要封锁某个数据项时，为了制造某种混乱，必须至少还要封锁其它数据项。封锁的原则是不能通过推理得出某些敏感信息。

(2)组合结果　组合结果的一种方式是把行或列组合起来，可以把敏感信息和非敏感信息混合在一起，从而产生敏感性较低的结果。组合结果的另一种方式是出示值的范围，而不透露准确值，这样即便得到某个数据项的信息，那也只是非确切的值。还有一种组合方式是舍入，把实际值舍入到离其最近的一个确定值，比如 26～74，全部计为 50，这也可以防止泄露准确信息。

(3)随机数致乱　这是通过小错误扰乱数据库的值。如果 x_i 是数据项 i 的确切值，ε_i

是对统计结果加在 x_i 上的随机错误项，ε_i 值可正可负。所以，某些报告结果可能高于真实值，也可能低于真实值，它们可以接近于真实值，但又不能使人确信它的真实性。随机数致乱易于实现，因为它易于保存所有 ε_i 的值，以便对等价查询产生同样的结果。

(4)查询分析　一种更复杂也更安全的方法是查询分析。它要对每个用户维护一个查询记录，并且要判断在上下文中给定前面的结果后，再进行查询可能会推断出什么，从而确定是否响应查询。显然，查询分析的实现是相当困难的，特别是当 k 个用户联合起来进行攻击的时候。

6.6 多级安全数据库

在一些有安全性要求的数据库中，数据的安全性级别往往不是保密或非保密两种，可能需要几种安全性级别；同一记录中不同字段的安全性要求也可能不同。因此，数据库中数据安全性的粒度可能要划分到字段，同时用户的权限也应该不同。所以，多级安全数据库对于数据库的数据元素和用户都要求两级或多级安全性。

在多级安全数据库中，数据的完整性和保密性要求更高。那么，如何保证低级用户不能读写高级数据，而高级用户也不能往低级的数据项中写值呢?在多级情况下，两个不同级别的用户进行相同的查询，可能由于随机数致乱而得到两个不同的答案，而为了保密，系统也只好向高级用户报告非准确的数据。可以想象，如果用户和数据项的安全性都是多级别(>2)时，多级安全数据库的实现将是十分繁杂的。

下面将给出实现多级安全的若干建议。所谓建议，就是说这些方法还是不成熟的，它们都存在这样或那样的缺陷。

1. 划分

对多级数据库按其安全性进行划分，构成各自的数据库，每一个有其统一的安全级别，就像在房间内有着不同级别的文件柜一样。但是，这种方法增加了数据的冗余度，而且影响了高级用户访问低级数据。

2. 加密

把敏感数据进行加密。它能够较好地解决用户和数据项安全性级别的需求，同时将敏感数据以密文的方式存放在数据库中，防止了任何形式的泄密，比如低级用户非法越权访问。但是，用户读写数据时都要进行加、解密运算，使系统的开销增大，特别地对查询操作，可能要进行大量数据的解密操作，更会使用户难以忍受。

3. 完整性锁

完整性锁是希望既能提供数据完整性又能控制对数据库访问的一种方法。完整性锁的每个数据项由三个域组成，分别存放数据、密级与校验和，如图 6.3 所示。为了保持效率，数据以明文形式存储在第一个域。第二个域存放该数据的敏感级标记，该标记应该加密存放，保证其隐蔽性和不可伪造，以便限制用户对数据的访问。第三个域是密码校验和，保证数据值和安全级不被改动，它的值是由前两个域的值以及该字段在数据库中的位置等因素确定。

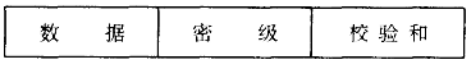

图 6.3　完整性锁

4. 交换过滤器

交换过滤器是数据库管理系统可信赖接口的一种简化，它过滤筛选用户的请求，如果必要则对其进行改编，所以只将有适当敏感度的数据返回用户。

一个交换过滤器是一个过程，它是用户和数据库管理程序间的一个界面。在用户提交查询请求后，过滤器重新改编查询，以便让数据库管理程序先筛选出不可查询的记录，然后由过滤器再进行第二次筛选，选出哪些用户有权访问的数据。

如果过滤器用在记录级、属性级或字段级，则应满足下述条件：

(1)在记录级，过滤器要验证数据及校验和，并确认用户的可访问性。

(2)在属性级，过滤器要检查该用户是否有权访问该属性的值；返回时，删除用户无权访问的属性。

(3)在字段级，鉴别该用户的级别是否与该字段的安全性相适应。

5. 视图

一个视图是数据库的一个子集，一个用户依据其级别可以确定他的一个视图，即该用户的查询访问只能局限在该视图之内。这一子集保证用户不能访问超出允许数量的值。

视图也是数据库中的一种关系。因此，当数据库中的数据变化时，视图子集中的数据也随之改变。

此外，还有一些其它的方法试图解决多级数据库的安全性，不过所有这些方法还都处于初级阶段，它们虽有各自的优点，但也有相应的缺陷，实现真正的多级安全数据库系统还需要一段时间。

6.7　密文数据库

在上一节提到的几种多级安全数据库方案中，数据大都是以明文的形式存放。对于恶意的攻击者来说，他可能设法绕开安全机制或利用其漏洞从而获取他不应得到的数据。因此，在某些实际情况中，密文数据库是一种令人满意的安全措施。

6.7.1　密钥转换

1. 密文数据库的简化形式模型

密文数据库的简化形式模型可描述如下：

数据库 $D=\{d_1, d_2, \cdots, d_n\}$，$d_i$ 代表安全性要求相同且可用同一数据密钥加密的一类数据，它可以是记录级、属性级，也可以是字段级；相应的数据密钥集 $KD=\{kd_1,$

$kd_2, \cdots, kd_n\}$；每类数据还对应一参数，称为多向锁，用集合 $ML = \{ml_1, ml_2, \cdots, ml_n\}$ 表示。加、解密算法为(E, D)；这样密文数据库可表示为 $C = \{c_i \mid c_i = E(kd_i, d_i), i = 1, 2, \cdots, n\}$。用户集 $U = \{u_1, u_2, \cdots, u_m\}$，相应地用户密钥集 $KU = \{ku_1, ku_2, \cdots, ku_m\}$，$u_i$ 可访问的数据是数据库 D 的一个子集 D_i。

用户 u_i 为了访问 D_i 中的数据，需要先提交用户密钥 ku_i，系统应将 ku_i 通过变换得到 kd_i 才可能实现。因此，把用户密钥转换为数据密钥的转换函数 f 必须满足

$$\forall\ i \forall\ j(f(ku_i, ml_j) = kd_j \quad \text{当且仅当 } d_j \in D_i) \qquad \langle 1 \rangle$$

图 6.4 表示了用户访问数据库的过程。其主要步骤是：

(1)用户 u_i 向安全管理系统(SMS)提供用户密钥 ku_i 及查询请求 q。

(2)根据 q，SMS 通过数据库管理系统 DBMS 从数据库中得到要查询的(ml_j, c_j)。

(3)SMS 利用 ku_i, ml_j 做 $d \leftarrow D(f(ku_i, ml_j), c_j)$。

(4)SMS 将 d 返回给用户 u_i。

其中第(3)步是关键。根据〈1〉式，当且仅当 u_i 可访问 d_j 时，$f(ku_i, ml_j) = kd_j$，从而 $d = d_j$，这样保证了系统的安全读取。上述过程实现了从用户密钥到数据密钥的安全转换。

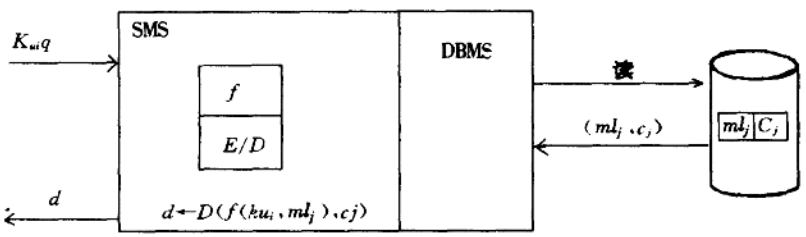

图 6.4　密文数据库访问过程

可以有不同的方法实现密钥转换，较好的方案之一是转换表形式。

2. 转换表方案

转换表方案可描述如下：

(1)用 i 代替多向锁 ml_i，密文数据库

$$C = \{(i, c) \mid c_i = E(kd_i, d_i),\ i = 1, 2, \cdots,\ n\}$$

式中：i 作为数据安全类 d_i 的标志。

(2)每个用户 u_i 有唯一的用户标志 i，访问数据库时 u_i 提供(ku_i, i)。

(3) *SMS* 中有转换表 $T = t[1 \cdots m, 0 \cdots n]$。并有一传统加解密算法$(E', D')$，$T$ 中元素是

$$t[i, j] = \begin{cases} E'(ku_i, kd_j) & d_j \in D_i \quad 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n \\ \text{无定义} & d_j \notin D_i \end{cases} \qquad \langle 2 \rangle$$

$$t[i, 0] = E(ku_i, c_0) \qquad c_0 \text{ 是任意常数}$$

式中：$t[i, 0]$用于身份验证。表 T 由系统管理员实现。

此时，用户读取数据的过程是：u_i 提供(ku_i, i)及查询请求 q，系统根据 q 找到对应信息(j, c_j)；然后 SMS 以 i, j 查表 T 得到 $t[i, j]$，进行操作：$d \leftarrow D(D'(ku_i, t[i,$

$j]$)，C_j)。由〈2〉式，当且仅当 u_i 可访问 d_j 时，$d = d_j$，因此可以实现安全读取。

3. 系统主要管理功能与操作

系统管理员掌握数据库主密钥 mk，所有的数据密钥加密存储于一维表 $KC = kc[1\cdots n]$ 内，其中 $kc[i] = E'(mk, kd_i)$，$1 \leqslant i \leqslant n$，这时系统的主要管理功能及相应操作如下：

(1)增加一用户　系统管理员在验证其身份，确定其可访问的数据类 $D' \leqslant D$ 之后，在 T 中加一行，行号为 i，给用户一个标志 i，用户自选密钥 ku_i，则所加第 i 行内容为：

$$t[i,j] = \begin{cases} E'(ku_i, D'(mk, kc[j])) & d_j \in D' \\ \text{无定义} & d_j \notin D' \end{cases} \quad j = 1,2,\cdots,n$$

$$t[i,0] = E'(ku_i, c_0) \qquad c_0 \text{ 为常数}$$

(2)删除一用户　在表 T 中删除相应的行。

(3)更换用户密钥　用户先提供(ku_i, i)，系统通过比较 $E'(ku_i, C_0)$ 与 $t[i,0]$ 验证用户身份，正确后系统再要求用户提供新密钥 ku'，然后对表 T 的第 i 行修改如下：

$$t[i,j] \leftarrow E'(ku', D'(ku_i, t[i,j])) \qquad j = 1, 2, \cdots, n$$

$$t[i,0] \leftarrow E'(ku', c_0)$$

(4)修改用户权限　由系统管理员在表 T 的对应行上增加或删除一些 $t[i,j]$ 的值。

(5)修改数据安全类标志　只在密文数据库中做相应的改动，表 T 无变化。

(6)增删数据　与一般数据库类似，只是需要确定新数据的安全类标志。

(7)互换数据密钥　系统必须知道所有用户密钥，然后既要更新数据库，又要对表 KC 及表 T 中与这些密钥对应的列进行修改。

(8)增删数据安全类　既要改动数据库，又要对表 T 做列的增删。

从上可以看出，转换表基本满足了密钥转换的各种要求，同时系统有关安全管理操作都是在转换表上进行，而对数据库和用户几乎没有影响。此外，采用转换表之后，依据 i, j 的值可立即得到 $t[i, j]$，因此可以在常数时间内快速完成密钥转换，而与数据及用户的安全类级别数无关。这些都是转换表的优点。所不足的是操作第(7)和第(8)步都较为困难，这些弱点可以在两级转换表中得到克服。

4. 两级转换表

两级转换表主要依据如下设想：数据库中数据安全类的数目较少，同时用户安全类的数目也不多。即很多用户属于同一用户安全类。

设 $UC = \{uc_i | \text{对所有 } u_j \in uc_i,\ u_k \in uc_i,\ \text{满足 } D_j = d_k\}$，亦即 uc_i 是一个用户安全类，其中各用户可访问的数据类是相同的，仍用 D_i 表示。每个用户安全类设定一个用户类密钥 kuc_j，这样，密钥转换可以分两级完成。第一级从用户密钥 ku_i 通过表 T_1 转换得到 kuc_j，第二级通过表 T_2 从 kuc_j 转换得到 kd_k。

因为一个用户只属于一个用户类，所以表 T_1 是一维表，$T_1 = t_1[1\cdots m, 1\cdots 3]$，其中

$$t_1[i,1] = E'(ku_k, kuc_j) \qquad u_i \in uc_j$$

$$t_1[i,2] = j \qquad \text{用户类标志} \quad 1 \leqslant i \leqslant m \qquad \langle 3 \rangle$$