支持快速查询的数据库加密方法研究

崔宾阁[1,2] 刘大昕[2] 王 桐[2]

(山东科技大学信息科学与工程学院 青岛 266510)[1]

(哈尔滨工程大学计算级科学与技术学院 哈尔滨 150001)[2]

摘 要 为了解决数据库中加密字符串数据的查询问题,提出了为待加密的字段建立辅助索引字段的两阶段查询方法。索引字段的内容由原始数据的划分值和特征值两部分组成,它可以用来支持字符串数据的精确匹配查询和模糊匹配查询。查询加密数据时,首先利用索引字段对加密数据进行一次粗糙查询,然后在解密的数据上再进行一次精确查询。实验表明,其性能较传统的先解密后查询方法有较大的提高。

关键词 数据库加密,两阶段查询,粗糙查询,精确查询

Practical Techniques for Fast Searches on Encrypted String Data in Databases

CUI Bin-Ge[1,2] LIU Da-Xin[2] WANG Tong[2]

(College of Information Science and Technology, Shandong University of Science and Technology, Qingdao 266510)[1]

(Computer Science and Technology Institute, Harbin Engineering University, Harbin 150001)[2]

Abstract To solve the problem of querying on encrypted string data in databases, an auxiliary index column is built for each attribute need to be encrypted, which will be used in two-phrase query. The contents of the index column are composed of two different parts. The first part corresponds to the partition value of the encrypted string data, which will be used in the exact match query. The second part corresponds to the feature value of the encrypted string data, which will be used in the fuzzy match query. When querying data, coarse query is performed first on the encrypted data based on the index column, which filtrates those records not related to the query conditions. The remaining records will be decrypted, and refined query is performed on the decrypted data. Results of a series of experiments validate the functionality and usability of the approach.

Keywords Database encryption, Two-phrase query, Coarse query, Refined query

1 引言

一般而言,数据库系统提供的基本安全技术能够满足一般应用的要求。但对于一些重要部门的应用,仅靠上述这些措施难以充分保证数据的安全性。某些用户尤其是内部用户,仍可能非法获取用户名、口令字,或利用其他方式越权使用数据库,甚至可以直接打开数据库文件来窃取或篡改信息。因此,有必要对数据库中存储的重要数据进行加密处理,以强化数据存储的安全保护[1~4]。

在对数据库中的数据进行加密之后,如何对加密数据进行查询成为一个难题。一种方法是首先对加密数据进行解密,然后对解密数据进行查询。但由于要对整个数据库进行加/解密操作,开销巨大,在实际操作中是不可行的[5,6]。另一种方法是直接对加密数据进行查询,即首先对查询语句中的条件值进行相同的加密处理,然后与数据库中存储的加密数据比较。然而,由于数据加密后一些固有的属性(如数据的有序性、相似性、可比性等)遭到破坏[7],该方法的适用范围是非常有限的。

本文提出一种对加密数据的两阶段查询方法。在查询加密数据时,首先利用索引字段对加密数据进行一次粗糙查询,过滤掉大部分不满足查询条件的记录;然后对剩余记录中的加密数据进行解密,在解密的数据上再进行一次精确查询,最终实现查询目标。这种查询方法可以减少操作代价较大的解密数据,从而提高对加密数据查询的性能。

2 相关的研究工作

Hacigumus 等[8]提出一种基于数据库-服务-提供者模型对加密数据进行查询的方法。该方法对加密数据查询的支持是通过将属性的值域划分为不相交的区间,属性值之间的比较转化为它们所隶属区间范围的比较实现的,因而它可以支持各种数据类型的精确匹配查询。但对于字符串数据的模糊匹配查询(如包含),该方法是无能为力的。

王正飞等[9]提出一种对加密字符串数据进行模糊匹配查询的方法。该方法利用一个特征函数将字符串数据映射到一个二进制位串,并将属性值之间的相似性比较转化为它们所对应的二进制位串中位的比较,因而它可以有效实现对加密字符串数据的模糊匹配查询。但对于加密字符串数据的范围查询(如大于、小于),该方法是无能为力的。

本文提出的方法继承了前两种方法的优点,它将字符串数据的划分值和特征值合并于同一个索引字段中。划分值用于支持字符串数据的精确匹配查询,而特征值用于支持字符串数据的模糊匹配查询。通过修改查询语句的转换函数,本文方法能够实现各种类型的查询请求,同时系统的性能也得到了提高。

3 系统的体系结构和控制流

我们提出的系统在应用程序和数据库管理系统之间增加了一个数据库加/解密引擎,它的体系结构和控制流如图 1 所

崔宾阁 博士研究生,主要研究方向:安全数据库;**刘大昕** 教授,博导,主要研究方向:数据库与知识库;**王 桐** 博士研究生,主要研究方向:XML 数据库。

示。

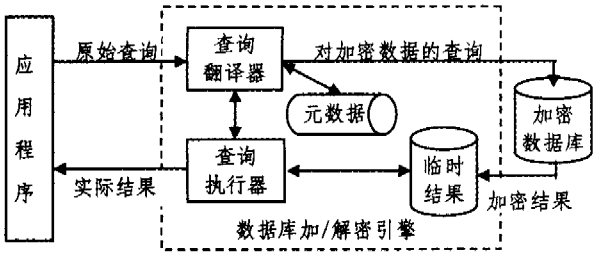


图 1　系统的体系结构

在图 1 中，查询翻译器负责将用户的原始查询转换为对加密数据的查询，在转换的过程中它需要访问元数据；元数据是一些转换规则的集合，用于对查询语句进行修改；查询执行器负责对加密查询结果进行解密，并在解密的数据上再执行一次原始查询；最后查询执行器将精确查询结果返回给用户。这种体系结构不需要对原有的应用程序和 DBMS 进行修改，因而它具有良好的适用性。

4　加密字符串数据的存储与查询

4.1　加密字符串数据的存储

4.1.1　索引字段的定义

索引字段的内容由字符串数据的划分值和特征值两部分组成。为了对这两部分的内容进行解释，我们需要引入一些基本概念。

定义 1　划分函数：将属性 $R.A_i$ 的值域 D_i 映射到划分 $\{p_1, p_2, \cdots, p_k\}$，满足：(1)所有划分的并可以覆盖整个域；(2)任意两个划分不重叠。形式上，定义函数 *partition* 如下：$partition(R.Ai) = \{p_1, p_2, \cdots, p_k\}$。

文[8]中给出了数值型字段值域的划分方法，通常采用等宽或等深划分的方法。对于字符型字段，情况要复杂一些。一种方法是根据属性值的概率分布对属性的值域进行划分，使得每个划分包含的元素数目大致相等。比如说，假定属性 A_i 的取值范围为英文字典中的所有单词，则 A_i 值域一个可能的划分为 $\{(a.\text{-}c.), (d.\text{-}h.), (i.\text{-}o.), (p.\text{-}r.), (s.\text{-}z.)\}$，其中 $(a.)$ 表示以字母 a 开头的所有单词的集合。

定义 2　标识函数：为属性 A_i 的每个划分 p_j 指定一个标识符 $ident_{R.A_i}(p_j)$。图 2 中说明了对属性 A_i 的 5 个划分指定的标识符。

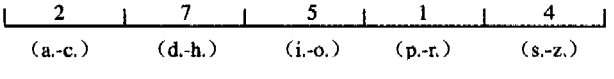


图 2　A_i 的划分和标识函数

定义 3　映射函数：将属性 A_i 域中的一个值 v 映射到 v 隶属划分的标识符。形式上，$Map_{R.A_i}(v) = ident_{R.A_i}(p_j)$，其中 p_j 是包含 v 的划分。例如，在图 2 中，$Map_{R.A_i}(me) = 5$。

映射函数可以划分为两类。如果 $v_i < v_j \Rightarrow Map_{R.A_i}(v_i) \leqslant Map_{R.A_i}(v_j)$，则称该函数是排序保护的；否则称它是随机的。毫无疑问，随机映射函数能够提供更好的安全性，但是针对它的查询翻译策略也更为复杂。为了简化讨论，本文中采用了排序保护映射函数，读者可以参考文[8]获得随机映射函数的查询翻译策略。

定义 4　特征函数：将字符串 $c_1c_2\cdots c_n$ 映射到二进制位串 $b_0b_1\cdots b_{m-1}$。形式上，$PC(c_1c_2\cdots c_n) = (b_0b_1\cdots b_{m-1})$。$H$ 为 Hash 函数。如果存在某个 j，满足 $H(c_j, c_{j+1}) = i, 1 \leqslant j \leqslant n-1, 0 \leqslant i \leqslant m-1$，则 $b_i = 1$；否则 $b_i = 0$。

例如，s_1 为字符串 abcehklst，Hash 函数 H 将 8 个对偶 ab，bc，…，st 散列到 0～15 之间的某个数。$s_2 = PC(\text{abcehklst}) = (0010100010101001)_2$。可以发现，$s_2$ 中仅出现了 6 个 1，这是由于 8 个不同对偶字符的 Hash 值可能相同，在 s_2 中相同的位数置 1。

在定义了映射函数和特征函数之后，对于任何一个字符串，我们可以求得它的划分值和特征值。为了便于对查询条件进行翻译，我们规定索引字段的前 h 位表示字符串数据的划分值，后 m 位表示字符串数据的特征值，整个索引字段的长度为 $h + m$ 位。如果划分值的位数小于 h 位，则在它的左边填充 0。例如，对于前面的字符串(abcehklst)，它所对应的索引字段的内容为(0200101000101010001)。

4.1.2　加密关系存储模式

对于每个关系 $R(A_1, \cdots, A_r, \cdots, A_n)$，$A_r$ 为待加密的字符串字段，加密后的关系模式为 $R^E(A_1, \cdots, A_r^E, \cdots, A_n, A_r^s)$。其中，$A_r^E$ 是对应于 A_r 的加密字符串字段，即 $A_r^E = E(A_r)$；A_r^s 是对应于 A_r 的辅助索引字段，通过对 A_r 的划分值和特征值组合得到。

4.2　加密字符串数据的查询

根据加密关系的存储模式，使用两阶段的查询方法实现对加密字符串数据的 SQL 查询。第 1 阶段，对字符串字段的查询被转化为对其相应的索引字段的查询。然而，存在一些记录满足索引字段的查询条件，却不满足字符串字段的查询条件，导致“伪记录”的产生。第 2 阶段，对第 1 阶段过滤后的记录进行解密，然后在解密的数据上进行一次精确查询。

4.2.1　查询条件的转化

两阶段查询的关键是如何将对加密字段的查询转化为对索引字段的查询。根据字符串数据查询的种类，分下面几种情况进行讨论。

(1)精确匹配查询：如 $A_i > v$，$A_i = v$，$A_i < v$ 等。

如果查询条件为 $A_i = v$，可以基于索引字段对加密关系进行双重过滤。首先，查找所有与 v 隶属相同划分的记录；其次，继续查找所有与 v 具有相同特征值的记录，从而获得更为精确的查询结果。形式上，$\text{Trans}(A_i = v) \Rightarrow A_i^s = \text{Map}_{R.A_i}(v)$ & $PC(v)$，符号 & 表示字符串连接。例如，$\text{Trans}(A_i = \text{davids}) \Rightarrow A_i^s = (210010100010100001)$。

如果查询条件为 $A_i > v$，可以基于索引字段中的划分值对加密关系进行过滤。由于本文中采用了排序保护映射函数，属性值 $w > v$ 意味着 $Map_{R.A_i}(w) \geqslant Map_{R.A_i}(v)$，因此只需对索引字段中划分值的大小进行比较即可。形式上，$Trans(A_i > v) \Rightarrow A_i^s(Map_{R.A_i}(v)$ & $00\cdots0$，其中 0 的个数为 m。例如，$Trans(A_i > davids) \Rightarrow A_i^s(2100\cdots0)$。对 $A_i \geqslant v$ 查询条件的处理与 $A_i > v$ 相同。

如果查询条件为 $A_i < v$，可以在加密关系中查找所有满足 $Map_{R.A_i}(w) \leqslant Map_{R.A_i}(v)$ 的记录 w。形式上，$\text{Trans}(A_i < v) \Rightarrow A_i^s \leqslant \text{Map}_{R.A_i}(v)$ & $11\cdots1$，其中 1 的个数为 m。对 $A_i \leqslant v$ 查询条件的处理与 $A_i < v$ 相同。

(2)模糊匹配查询：如 A_i like $c_1c_2\cdots c_k$ 等。

如果查询条件为 A_i like $c_1c_2\cdots c_k$，可以基于索引字段中的特征值对加密关系进行过滤。在加密关系中查找在 A_i^s 中第 $h + H(c_j, c_{j+1})$ 位 $(1 \leqslant j \leqslant k-1)$ 为 1 的记录的集合。形式

上，$Trans(A_i \text{ like } c_1c_2\cdots c_k)\Rightarrow A_i^s \text{ like } s$，其中

$$s[i]=\begin{cases}'1', (\exists j)(h+H(c_j,c_{j+1})=i), 1\leqslant j\leqslant k-1\\ '?', \text{其它} \qquad 1\leqslant i\leqslant h+m\end{cases}$$

'?'是代表单个字符的通配符。例如，$Trans(A_i \text{ like vid})\Rightarrow A_i^s$ like '?????? 1????? 1?????'。

(3)复合查询：它是由 and 和 or 将两个或多个简单查询组合而成。

复合查询的转化方式如下：$Trans(A_i \text{ op}_1 v_1)\text{and}(A_i \text{ op}_2 v_2))\Rightarrow Trans(A_i \text{ op}_1 v_1)\text{and } Trans(A_i \text{ op}_2 v_2)$；$Trans((A_i \text{ op}_1 v_1)\text{or}(A_i \text{ op}_2 v_2))\Rightarrow Trans(A_i \text{ op}_1 v_1)\text{or } Trans(A_i \text{ op}_2 v_2)$。

4.2.2 *查询算法*

算法 1 两阶段查询算法

第 1 阶段：粗糙查询

(1)利用元数据(如映射函数、特征函数等)中的规则，转换查询 SQL 中的条件语句；

(2)执行已经转换的 SQL 语句，返回查询结果，并抛弃索引字段。

第 2 阶段：精确查询

(1)解密第 1 阶段返回的记录，存放到一个临时表；

(2)修改 SQL 语句，将原始 SQL 语句中的关系表用临时表替换；

(3)执行调整后的 SQL 语句，得到查询结果。

实际上，算法 1 中的第 1 阶段是用来过滤与查询条件无关的记录，以减少第 2 阶段解密的记录数。相对于查询操作，解密操作的代价要高许多。算法 1 的方法正是减少解密操作的代价，从而提高系统的查询性能。

5 算法分析

5.1 安全性分析

在加密关系中，敏感字段通过分组加密算法(如 DES，AES)加密后，以密文形式存在加密数据库中，从而保证它的安全性。关于加密算法本身的安全性，不在本文的讨论范围。但新增的索引字段存在潜在的不安全因素，需要进一步分析。

(1)划分值的安全性。由于对属性值域的划分是基于属性值的概率分布，并且每个划分包含的元素数目大致相等，因此用户很难通过统计攻击获得划分值和划分的对应情况。每个划分包含的元素数目不能太少(一般不少于 10 个)，因此用户也不太容易通过已知明文攻击获得划分值所对应的属性值。

(2)特征值的安全性。由于在特征值定义中使用了 Hash 函数，攻击者很难通过特征值直接分析出敏感字段中的值。但是当特征值位数 m 越大时，不同的对偶字符经过散列后，其对应于特征值中不同位的可能性越大。攻击者可以通过统计得到特征值中各位出现 1 的概率。此外，攻击者也容易得到各对偶字符在英文中出现的概率。通过比较这两种概率，攻击者很可能根据特征值推断出敏感字段的值。解决方法是限制 m 的大小，使得特征值中每位对应的对偶字符数目不会太少。同样，当 m 越大时，不同的字符串经过特征函数处理后，其对应的特征值不同的可能性越大，攻击者可以通过已知明文攻击来获取敏感字段的值。解决方法同样是限制 m 的大小，使得不同的字符串可以对应于相同的特征值。

5.2 过滤效率分析

在算法 1 中，字符串字段的查询被转化为对索引字段的查询。然而有一些并不满足查询条件的记录可能满足索引字段的查询条件，导致"伪记录"的产生。这种"伪记录"的数量与划分值位数 h 和特征值位数 m 的大小密切相关。当 h 越大时，划分数目越多，每个划分包含的元素数目越少，在查询的第 1 阶段出现的伪记录越少(至多一个划分包含伪记录)，所以过滤效果越好。同样地，当 m 越大时，不同的字符串经过特征函数处理后，其对应的特征值重复的可能性越少，在查询的第 1 阶段出现的伪记录越少，所以过滤效果越好。反之，过滤效果越差。

5.3 存储空间分析

由于在加密数据库表中增加了索引字段，因此相应地增加了存储空间。很明显，新增存储空间的大小与索引字段的位数成正比。当索引字段位数较大，新增存储空间较大；反之，新增存储空间较少。假设数据表的记录数为 n，索引字段的位数为 $h+m$ 位，那么新增存储空间为 $n*(h+m)$ 位。

从上面对安全性、过滤效率和存储空间的分析中可以看出，安全性与过滤效率和存储空间存在相互制约的关系，如表 1 所示。

表 1 *索引字段位数对安全性、过滤效率和存储空间的影响*

索引字段位数	安全性	滤效率	存储空间
h,m 较大时	较差	较好	较大
h,m 较小时	较好	较差	较小

6 实验与性能分析

实验目的是验证本文方法在查询时间和过滤效率方面的有效性。根据 TPC-H 标准[10]，利用 dbgen 工具自动生成数据库，数据库的大小为 10MB，也就是比例因子取 0.01。TPC-H 的数据库包括 8 个表，其中 lineitem 被用来作为本次实验的数据源，它的字段 comment 作为敏感字段，需要进行加密处理。加密算法是 safer++，由 C 语言编写，分组长度为 128b。密钥长度也为 128b，实验环境为 Windows 2000 操作系统，SQL Server 2000 数据库服务器，PIV 2.6GHz 的 CPU，256MB 的内存。每个实验数据都做了 10 次，取其平均值。

6.1 实验 1：过滤效率的测试

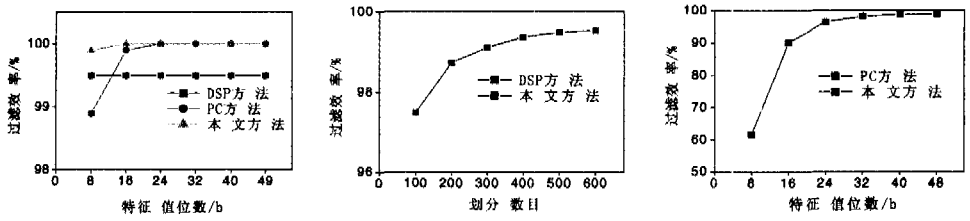
为了准确表示算法 1 的第 1 阶段过滤掉不符合查询条件的记录数的能力，定义过滤效率如下：

定义 5 如果数据表中记录数为 N，在查询算法 1 中第 1 阶段返回的记录数为 n_1，在第 2 阶段返回的记录数(即真正符合查询条件的记录数)为 n_2，则过滤效率$=\frac{N-n_1}{N-n_2}$。其中，$N-n_2$ 表示不符合查询条件的记录数，$N-n_1$ 表示第 1 阶段过滤掉的记录数。

在第 1 个实验中，测试查询算法 1 中第 1 阶段过滤的效率。过滤效率与划分数目和特征值位数密切相关。图 3(a)、图 3(b)和图 3(c)分别反映了本文方法与文[8]中的方法(简称为 DSP 方法)和文[9]中的方法(简称为 PC 方法)对于 3 种不同查询条件过滤效率的比较。

图 3(a)给出了对于查询条件 $A_i=v$，特征值位数的变化对于 3 种不同查询方法过滤效率的影响，属性 address 的值域被划分为 200 个桶。DSP 方法不受特征值位数变化的影响。本文方法和 PC 方法随着特征值位数的增加，过滤效率都相应增加。这是由于特征值位数增加，不同字符串被 Hash 函数散列后，对应不同特征值的概率增加。也就是说，同一特征

值对应的字符串数量减少。而查询是根据特征值来过滤无关记录，从而导致过滤效率提高。由于本文方法根据划分值和特征值进行双重过滤，因而它的过滤效率更高。但从图 3(a)中可以看出，对于查询条件 Ai = v，3 种方法的过滤效率都接近于 100%，进一步提高的余地不大。由于特征值位数增加，存储空间增加，安全性降低，因此综合考虑选择特征值位数为 32。



(a) $A_i = v$　(b) $A_i > v$ ($A_i < v$)　(c) A_i like $c_1 \ldots c_k$

图 3　过滤效率测试

图 3(b)给出了对于查询条件 $A_i > v$($A_i < v$)划分数目的变化对于过滤效率的影响。PC 方法不支持此类查询。随着划分数目的增加，本文方法和 DSP 方法过滤效率都相应增加。这是由于划分数目增加，每个划分包含的元素数目减少。而根据前面的查询方法，只有一个划分，即 v 隶属的划分包含无关记录，所以过滤效率提高。本文方法和 DSP 方法过滤效率是相同的。由于这两种方法过滤效率都已经达到 99%，出于同样的安全性考虑，选择划分的数目为 200。

图 3(c)给出了对于查询条件 A_i like $c_1 c_2 \cdots c_k$，特征值位数的变化对于过滤效率的影响。DSP 方法不支持此类查询。随着特征值位数的增加，本文方法和 PC 方法过滤效率都相应增加，并且它们的曲线是完全相同的。当特征值位数增加到 32 位后，过滤效率增加幅度减缓。由于查询的过滤效率已经达到 98%，出于安全性考虑，选择特征值位数为 32。

从图 3 中可以看出，本文方法可以同时支持上述 3 种不同类型的查询，而 DSP 方法和 PC 方法只能支持其中的两种，所以本文方法的适用范围更为广泛。在特征值位数为 32 位、划分数目为 200 时，过滤效率大于 98%，因而过滤效果比较理想。

6.2　实验 2：查询性能测试

在第 2 个实验中，就测试算法 1 查询所花费的时间代价与传统的先解密后查询方法、DSP 方法和 PC 方法进行比较。图 4 中给出了索引字段中特征值位数不同时，4 种查询方法所花费的时间代价。为了简单起见，我们只考虑了查询条件为 $A_i = v$ 的情况。

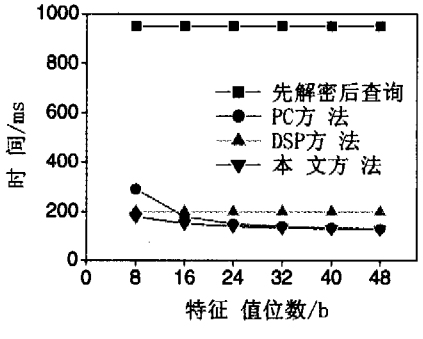


图 4　查询时间代价测试

在图 4 中，随着特征值位数的增加，本文方法和 PC 方法所花费的时间减少。这主要是因为 safer++算法解密速度慢，所花费的时间大约是查询数据所花费时间的 12 倍，而算法 1 采取的方法正是减少了查询时需要解密的记录数，从而提高了查询的性能。从图 4 中还可以看出，各种查询方法所花费的时间大约是先解密后查询所花费时间的 1/5，从而在很大程度上提高了系统的性能。

结论　通过在加密数据表增加索引字段，将加密数据的查询转化为对索引字段的查询，实现了加密字符串数据的两阶段查询方法。该方法需要计算每个字符串数据的划分值和特征值，并将它们存放在一个索引字段中。该方法可以支持字符串数据的精确匹配查询和模糊匹配查询，同时算法的过滤效率也得到了提高。只要索引字段的位数取合适的值，该方法可以具有较好的安全性，并且增加的存储空间很少。查询所花费的时间代价比先解密后查询方法减少了 80%左右。当然该方法还存在一定的不足，比如说对于包含有'<>'或'not like'的查询条件，唯一可用的方法仍然是对整个关系解密后再查询，这需要在今后做进一步的研究。

参 考 文 献

1　Waters B R, Balfanz D, Durfee G, et al. Building an encrypted and searchable audit log. In: The 11th Network and Distributed System Security (NDSS) Symposium, San Diego, California, 2004

2　Chang Yan-Cheng, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data. In: Proceedings of ACNS 2005, Lecture Notes in Computer Science 3531, Springer-Verlag, 2005. 442～455

3　Song D Xiaodong, Wagner D, Perrig A. Practical techniques for searches on encrypted data. The IEEE Symp on Security and Privacy, Oakland, California, 2000

4　Hacigumus H, Lyer Bala, Mehrotra S. Providing database as a service. The 18th Int'l Conf On Data Engineering, San Jose, California, 2002

5　He Jingmin, Wang Min. Cryptography and relational database management system. Interactive Dialogue with Educator form Across State, 2001

6　Oracle. Oracle9i database security for eBusiness. http://www.oracle.com/technology/deploy/ security/oracle9i/pdf/9isecbpa.pdf, 2001

7　Fanghanel T. Using encryption for secure data storage in mobile database systems: [Ph D dissertation]. Jena, Germany: Friedrich Schiller University, 2002

8　Hacigumus H, Lyer B, Li Chen, et al. Executing SQL over encrypted data in the database-server-provider model. The 2002 ACM SIGMOD Int'l Conf on Management of Data, Madison, Wisconsin, 2002

9　王正飞，王曼，汪卫，等. 数据库中加密字符数据的存储与查询. 计算机研究与发展，2004，41(suppl)：66～71

10　TPC-H. Benchmark specification. http://www.tpc.org, 2004