# 变长字段在关系数据库中的实现

复旦大学　沈毅

摘要：关系数据模型具有操作方便等许多优点，然而关系数据模型是一个框架理论，对于关系中每个字段都必须给定长度，即每个字段是定长的，这种限制给许多应用领域带来不便。

本文对关系数据库系统中的数据结构进行了分析，对索引方法进行了修改。对于变长字段采用保序散列技术进行索引，而不是常用的 B 树索引方法，使得变长字段能在关系数据库系统中实现。

## 一、引言

关系数据模型采取了最简单规范的数据结构，运用了先进的数学工具，可以把二维表进行任意的分割和组装，随机地构造出各式各样的用户所需要的表格，即关系。而且关系数据模型具有坚实的理论基础，已作为数据库领域的发展方向。

然而关系模型仍然存在着某些不足之处。例如，缺乏对于变长字符串(或变长字段)处理的能力。这是由于关系模型的理论是一种框架理论，对于每一个字段都必须事先指定其长度，而且大多数的关系数据库系统都使用类 B 树的索引结构，使得关系模型不能有效地处理变长字段。

虽然某些特殊的关系数据库系统考虑到了变长字段的情况，但是对变长字段的处理能力不能令人满意。例如：d BASEⅢ关系数据库系统增加了备注型字段(MEMO 字段)来处理变长字符串，但是用这种方法来处理的一个严重问题是空间浪费太大。它是以 512 字节为一个申请单位，一个申请单位只能用于一个变长字符串，对于超过 512 字节的字符串，必须再申请 512 字节，最多可申请 4096 字节。d BASEⅢ中使用 MEMO 字段的另一个不足对于变长字段没有检索功能。

针对关系模型中存在的这一问题，本文仔细分析了关系数据库系统中的数据结构，对关系数据库系统中常用的索引方法进行修改。对于定长字段仍采用关系数据库系统中原有的方法；对于变长字段，则用保序散列(Ordered Hash )技术来代替 B 树索引方法，使得变长字段能够在关系数据库系统中实现，而且对于变长字段值，具有以词(Word )为单位的检索功能。

具体工作是针对 dbase Ⅲ关系数据库系统进行的。

## 二、保序散列

保序散列就是要求在散列文件中的各记录要按照关键字的大小排列，如果删去穿插在记录之间的空地址，散列文件就变成了顺序文件。顺序文件对于一些与顺序有关的操作，如范围检索等是非常有效的。但是顺序文件在检索效率、维护效率、算法简洁等方面都次于散列文件。因此，散列的保序性是十分吸引人的。另外，如果散列函数的选择具有保序性，则由此对变长字段构造出来的索引文件已经是排序好了，不必再用分类命令对此再进行排序了。

设 M 是散列文件中预先设定的地址总数，每一地址存放一个记录，每一个记录中有一个关键字 K，该记录被分配到散列文件中的一个地址，由 h (K)计算得到，称为 K 的散列地址。

如果某一记录插入时，它的散列地址已经被占，则顺着地址增加方向试探下一个地址。

所谓“试探”是指比较新的关键字与已在该位置上的关键字，用关键字较小的记录放入此位置，用关键字较大的记录去试控下一个地址，如此继续直到遇到一个空地址。

与一般散列方法不同的是：当检索完文件的最后一个记录，不能转到第一个记录去继续试探，否则就可能把较大的关键字记录插到前面而破坏了散列的保序性。

使用这种方法存在的一个问题：随着插入操作的增加，散列文件中的记录逐渐偏向文件的后部。而且如果 M 不能增加的话，就会使算法失效，即虽然文件中还有空地址，但要插入的新记录却插不进去。

导致算法失效的根本原因是插入时的探查操作只允许向后延伸。这个情况可利用文件的有序性来克服。

在插入一个记录时，允许记录按当时的文件中记录的分布情况，或向后延伸，或向前延伸，因此检索操作中第一次比较所得的大小关系就决定了后面各步查找的算法。

算法 2、1：保序散列检索 Search

1.计算 h (K)——>y；

2.if K = (y) then{ 检索成功; goto 7；}

3.if K < (y) then −1 ——>t else 1 ——>t；

4.while ((y) <> 0 & 1 <= y +t <= M & sign (K−(y)) = t) {y +t ——>y；}

5.if K = (y) then { 检索成功; goto 7；} else 失败结束;

6.if K < (y) tken y −1 ——>y；

7.END

其中：(y)表示地址 y 中所存的健值;

k 是关键字;

sign 是符号函数:

$$sign(z)=\begin{cases}1 & z>0\\0 & z=0\\-1 & z<0\end{cases}$$

sign(k −(y)) = t 的意义是：当 t = 1 时表示 k > (y)；当 t = −1 时，表示 k < (y)，而 t 的正负由第三步决定，表示进一步查找的方向。在下列三种情况下可结束第四步的查找过程而进入第五步：(1)遇到空地址，(2)y 不在 1 与 M 之间；(3)向后查找时得 K < (y)，或向前查找时得 K > (y)。第六步中当 K < (y) 时作 y −1 ——>y，是为了保持不论朝什么方向查找，失败时总有 (y) < k。

插入操作有两个延伸方向，可用当时的记录分布情况作为选择的依据。本文取前半文件记录数减去后半文件记录数的值作为延伸方向的参数。插入时总是使记录向文件中央靠近。有两个特殊情况：

(1)最末一个空位的后面只能作向前延伸。

(2)第一个空位的前面只能作向后延伸。

算法 2、2：保序散列插入 Insert

1.用算法 2、1 作检索;

2.若 K = (Y)，文件中已存在被插记录的值，作错误处理;

3.if (y) 为空 then Goto 6；

4.选择延伸方向:

if ((y < B2 &d > = 0) 或 y < B1) then {1 ——>t ; y +t ——>y；}else −1 ——>t :

5.延伸:

if (y) 非空 then {r 与 y 中记录交换位置; y +t ——>y； goto 5；}

6.插入: r ——>y； N+1 ——>N;

7.if y > M / 2 then d −1 ——>d else d +1 ——>d

8.while (B1 <>0) {B1 +1 ——>B1；}

9.while (B2 <>0) {B2 −1 ——>B2；}

10.END

其中: r ——要插入的记录;

d ——前半文件记录数减去后半文件记录数;

B1 ——第一个空地址;

B2 ——最后一个空位置;

N ——散列文件中的记录总数。

## 三、散列函数

在散列方法中，最关键的是散列函数的选择，不同的散列函数适且于不同的应用要求，散列函数的好坏，决定一系统的效率和性能。

对于一个变长字段值，以词(Word)为单位，对其进行分割，对每个词取其前四个字符(对于不足四个字符的词，后面以空格补足)作为键(KEY)。

设每个词的前四个字符是 clc2c3c4，因为 ci(i = 1，2，3，4)是 ASCII 字符，取 ci 的大写 ASCII 字符值并缩小:

ci' = ci 的大写 ASCII 值 −32

其中: 32 是空格的 ASCII 值

令: $H1 = c1' * 26^{-1} + c2' * 26^{-2} + c3' * 26^{-3} + c4' * 26^{-4}$

新英汉字典中第一个单词为 aardvark(土豚)

则: H1 (aardvarl) = H1(AARD)

$= (65-32) * 26^{-1} + (65-32) * 26^{-2} + (82-32) * 26^{-3} + (68-32) * 26^{-4} = 1,32095$

新英汉字典中最后一个单词为 zymurgy (酿造学)

则：H1 (zymurgy) = H1 (ZYMU) = 2，31774

H1 (zymurgy) − H1 (aardvark) = 0，99679

即最后一个单词和第一个单词的间隔为 0，99679

$(65-32) * 26^{-4} = 0.0000722$ (65 是 'A' 的 ASC11 值)

在 H1 (clc2c3c4) 中，对第四个字符 c4 来说，H1 (clc2c3c4) 小数点后的第五位已受其影响，用四舍五入的方法可以得到：$(65-32) * 26^{-4}$ 的小数点后第五位是 7

令：H2 = ((H1 (clc 2c3c4) − H1 (AARDVARK)) / 7 * 10000

H2 ∈ [0，1423，4]

令散列函数为：H = roumd (H2)

round 为四舍五入函数，则 H ∈ [0，1423]

使用上述散列函数可以得到另一个功能是：可以对变长字段值进行题内词检索，这种功能对于文献情报检索系统来说是特别有用。

## 四、实现变长字段的文件结构

根据 d BASEⅢ规则：数据文件的扩展名为 DBF。字段类型为：字符型 (C)，数据型 (N)，日期型 (D)，逻辑型 (L)，备注型 (M)，现在再增加一个类型——变长字段类型，用"V"表示。类似于备注型字段，规定其长度为 8 个字节，用来存放库号 (指针)，指出变长字段值存放的位置。

库号 = 桶号 * 104 + 桶内序号

变长字段值都放在扩展名为 DBV 的数据文件中。DBV 的文件是桶式结构，每桶有 4096 字节。桶号用 4 字节表示，最多可有 9999 桶，所以，约可表示 40 兆的字符。

DBV 的文件的桶式结构是一维数组：char vstr[4096]。

存放形式按：

| 桶号 | 字段值长度 | 变长字符串 | ... | 字段值长度 | 变长字符串 | ... | ... |
|---|---|---|---|---|---|---|---|
| 4B | 4B | | 10B | 4B | | 10B | |

每桶的前四个字节用来存放桶号，从第五个字节开始按照："字段值长度，变长字符串，10 个空格"顺序存放，"字段值长度"的值等于"变长字符串"的长度，后面加 10 个空格是为了处理修改时的情况，即如果对变长字段进行修改，只要其修改后的长度小于原字段值长度+10，则把修改后的值仍放在原先的位置，否则就要把原先的库号作废，而把修改后的变长字段值放到 DBV 文件的尾部，同时须使用新的库号。当修改后的值添加到文件尾部时，对原位置的变长字段值 (修改前的) 不进行删除或压缩，实际上原位置处的值已处于"死区"，不会有记录访问到此，直到有压缩命令 (PACK) 时再收回这块空间。压缩的过程实际上是一个重新构造 DBV 的过程。

对 DBF 文件中某个字段进行索引时，如果索引字段是定长的，则仍用系统中 B 树方法进行索引，生成 NDX 文件。如果索引字段是变长的，则用保序散列方法对其索引，生成扩展名为 NHX 的文件。

NHX 文件是桶式结构，每桶可放 1500 个记录，每个记录由两个值组成，第一个值是关键字，第二个值是指针，即库号，指出该散列值对应的记录号存放在扩展名为 DBR 文件中的位置。

NHX 文件中的库号 = 桶号 * $10^4$ + 桶内序号

NHX 文件的桶式结构是：

```
struct HBUK
{int B1; /* 第一个可以存放的空地址 */
 int B2; /* 最后一个可以存放的空地址 */
 int D; /* 本桶中前半部分记录数减去后半部分记录数 */
 int BN; /* 桶号 */
 char FLAG; /* 本桶是否有溢出桶标志；'1'=有，'0'=无 */
 shruct HBUKI  HBUKT[1500];
 };
 struct HBUK1
  {char HVAL[4]; /* 键 */
 long HPNT; /* 库号，指出该键在 DBR 文件中的位置 */
 };
```

由于前四个字符为 c1c2c3c4 的单词所在的记录可能有 n 个，并且各个单词所对应的记录个数是随机的。所以，将某个单词所对应的记录号都用链结构联起来，把所有的这些链都存放在扩展名为 DBR 文件中。

DBR 文件是桶式结构，每桶可放 1024 个记录，每个记录由两个值组成，第一个值存放某单词所在的记录号，第二个值是一个指针(库号)，用来指出该单词所在的下一个记录号在 DBR 文件中的位置("记录号"指该单词所在的记录在 DBF 文件中的顺序编号，以下同)。

DBR 文件的库号＝桶号＊$10^4$+桶内序号

DBR 文件中的桶式结构是:

```
struct HREC
 {int BNO; / * 桶号 * /
int RNO; / * 桶内已有记录数 * /
struct  REC1  REC[1024];
};
struct REC1
 {long WREC; / * 桶号 * /
long WNEXT; / * 库号，指出下一个有相同键值的记录所在位置 * /
};
```

## 五、算法

算法中使用的符号如下:

KEY: 单词的前四个字符，作为散列键。

M: 一桶可放的记录数

y : 由散列函数 H 计算得到的散列地址。

y [HVAL]: 存放在散列地址 y 中的键值

y [HPNT]: 存放在 y 中的库号，指出 DBR 文件中对应的记录号存放位置。

KNO: 库号

BNO: 桶号

ORDNO: 桶内序号

VSTR: 变长字段值。

RECNO: DBF 文件中的记录号

ORDNO[ WREC] : DBR 文件中存放在桶内 ORDNO 处的记录号值。

ORDNO[ WNEXT] : DBR 文件中存放在桶内 ORDNO 处的库号，指出对应键的下一个记录号值在 DBR 文件中的位置。

算法 5.1 : 检索 SEARCH(KEY)

1.读 NHX 文件的第一桶;

2.计算 H(KEY)——>y ;

3.if KEY＝y [HVAL] then goto 8 ; / * 检索成功 * /

4.if KEY ＜ y [HVAL] then −1 −−−>t else 1 ——>t ;

5.while (y [HPNT] ＜＞ 0 & 1 ＜＝y +t & sign (KEY−y [HVAL]) ＝t )

{y +t ——＞ y ;

if y ＞ M then

if 有溢出桶 then

{ 写内存散列桶;

读溢出桶;

y ＝1 ; / * 对溢出桶从地址 1 开始处理 * /};

else

goto 9; / * 失败 * /

};

6 .if KEY ＜y [HVAL] then y −1 ——＞ y ; / * 保证结束时有 KEY＞y [HVAL] * /

7 .IF KEY＝y [HVAL] then goto 8 else goto 9 ;

8 .显示:

y [HVAL]——＞ KNO; / * 库号 * /

while (KNO＜＞ 0 )

{ 将库号 KNO 分解: 桶号−−−＞BNO; 序号——＞ORDNO;

读 DBR 文件中桶号为 BNO 的桶;

ORDNO[WREC]−−−＞RECNO; / * 取记录号值 * /

显示 DBF 文件中记录号为 RECNO 的记录和对应的变长字段值;

ORDNO[WNEXT]−−−＞KNO; / * 下一具有相同键值的记录号在 DBR 文件中的位置 * /

};

成功结束;

**9 .失败结束**

算法 5 .1 中第四步决定进一步查找方向，t ＝1 时表示向后找，t ＝−1 时表示向前找。在三种情况下可结束第五步的查找: (1 )y [HPNT]＝0 , 即遇到空地址; (2 ) 向前查找时超出下界 1 , 即 y +t ＜1 , 这里对上界没有限制，当地址 y ＞M 时，可到溢出桶中找; (3 ) 向后查找时

得 KEY＜y [HVAL]，或向前查找时得 KEY＞y [HVAL]，即此时找到了 KEY 应在的位置。第六步中，当 KEY＜y [HVAL]时作 y－1 ——＞y，是为了保持不论朝什么方向查找，失败时总有 KEY＞y [HVAL]。如果查找成功，则转第 8 步，将结果(对应的信息)显示出来。

算法 5、2：插入 INSERT (KEY，RECNO)

1.RECNO ——＞DBR 文件尾部，并取对应的地址值(库号)——＞KNOR;

2.调用算法 5、1 SEARCH(KEY)作检索;

3.if y [HPNT]＝0 then goto 7；／＊散列表中无 KEY，转 7 做插入操作＊／

4.if KEY＝y [HVAL] then

{／＊散列表中已有该键，所以只需在 DBR 文件中加上 KEY 所在的记录号 RECNO 即可＊／

y [HPNT]——＞KNO;／＊键首指针(库号)＊／

while (KNO ＜＞0)

{将库号分解：桶号－－－＞BNO;序号——＞ORDNO;

读 DBR 文件中桶号为 BNO 的桶;

ORDNO[WNEXT]——＞KNO;

};

KNOR－－－＞ ORDNO[WNEXT];／＊将新库号加到链尾＊／

goto 11;

};

5.选择延伸方向：if((d ＞＝0)或(y ＜B1))then{1——＞t；y +t ——＞y；}

else －1 ——＞t;

6.延伸:

while (y [HPNT]＜＞0)

{y [HVAL]＜＝＝＞KEY;／＊“＜＝＝＞”表示相互交换＊／

y [HPNT]＜＝＝＞KNOR;

y +t ——＞y;

if y ＞M then

if 有溢出桶 then

{写内存散列桶;

读溢出桶;

y ＝1；／＊新桶从地址 1 开始＊／

}

else／＊无溢出桶＊／

{置本桶有溢出桶标志;

写内存散列桶;

清内存散列桶并置初值;

y ＝1；／＊新桶从地址 1 开始处理＊／

goto 7；／＊转 7 做插入操作＊／

}

};

7 .插入：KEY－－－＞y [HVAL];KNOR－－－＞y [hpnt];

8 .if y ＞M／2 then d －1 ——＞d else d +1 ——＞d;

9 .while (B1 ＜＞0){B1 +1 －－－＞B1；}／＊B1 不为空时，累进＊／

10.while (B2＜＞0){B2 －1 ——＞B2；}

11.END

算法 5、2 中先将记录号 RECNO 加到 DBR 文件尾部，并返回其对应的地址值。第 4 步指出 KEY 已在散列表中，此时需根据链首 y (HPNT)找到链尾，把记录号 RECNO 在 DBR 文件中的地址加到链尾。第 5 步是选择查找方向，当进一步查找方向是向后时，还需作 y +t ——＞y，即 y +1 ——＞y 操作，这是因为调用算法 5、1 SEARCH(KEY)，当查找不成功时，总有 KEY＞y {[HVAL]。第 6 步是延伸的过程，直到找到空地址为止，当溢出时，则调入溢出桶处理。第 8 －－－第 10 步是修正参变量。

**参考文献**:


1.楼荣生“关于保序散列的一些问题’，＜＜计算机应用与软件＞＞，1987 年第 2 期，Vol、4，NO、2

2.张学平，“DBASE3 奥秘”。

3.Kurt Mehihorn，“Data structure adn algorithms 1：Sorting adn Searching”，1984

4.Donald E, Knuth, “The art of computer programming：, Vol,3, 1973

5.Robert J, Baron, Linda G Shapiro, “Dats structure techniques”, 1980


▲