

数据类型

数据类型特性

kBase 系统提供丰富的数据类型，以尽可能满足各种应用需求。

kBase 系统中基本字段类型分为四种，基本数值型、文本型数据、二进制数据、增强型数据。

- **基本数值型**包括 CHAR、VCHAR、MVCHAR、AMVCHAR、INTEGER、NUM、DATE、TIME、AUTO 等，这类数据特点与作用都同传统关系数据库类似；
- **文本型数据**包括 STRING、STRCHAR、MTEXT、MTEXTCHAR、TEXT、TEXTCHAR、LTEXT，主要是为了管理非结构化文档而设计的，它是 kBase 系统与传统关系数据库主要区别之一，实现了高速简捷的全文检索；
- **二进制数据** SOB、LOB、DOB 是为方便管理二进制数据而设计；
- **增强型数据**包括 VSM、TNAME、DNAME、MIXTEXT、MIXTEXTCHAR、MIXTEXTAUTO 等，这些字段类型各有特点，为 KBase 系统引入很多有特色的功能。

以下是各种数据类型的简表：

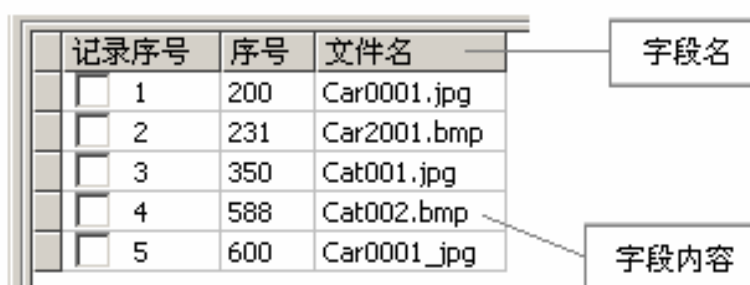
名称	分词/字	最大长度	变长	说明
CHAR		1-254(需指定)	否	字符串，支持“单一”索引
VCHAR		254	是	变长字符串，支持“单一”索引
MVCHAR		4K	是	多值字符串
SOB		254	是	二进制，没有索引
LOB		16M	是	二进制，没有索引
STRING	分词	254	是	文本
MTEXT	分词	4K	是	文本
TEXT	分词	16M	是	文本
LTEXT	分词	16M	是	文本
STRCHAR	分字	254	是	文本
MTEXTCHAR	分字	4K	是	文本
TEXTCHAR	分字	16M	是	文本
INTEGER		1-12(需指定)	否	整数，长度为用字符串形式保存数据的长度
NUM		1-32(需指定)	否	浮点数，长度为用字符串形式保存数据的长度
DATE		8-32(需指定)	否	日期
TIME		14-32(需指定)	否	时间
AUTO		12	否	自增，从 1 自动增加，不能指定值 总是有索引
VSM		16M	是	相似索引字段
TNAME		-		虚拟字段，返回表名称

DNAME		-		虚拟字段，返回表的显示名称
MIXTEXT	分词	-		虚拟字段，其它子段混合而成
MIXTEXTCHAR	分字	-		虚拟字段，其它子段混合而成
MIXTEXTAUTO	自动	-		虚拟字段，其它子段混合而成
DOB		2G	是	数字对象字段

CHAR

这种数据类型，用于存储定长的字符串，其最大长度由设计者限定，但最大长度不能超过 254 个字符。它通常用于容纳如人名、公司名、地址等内容。在 KBase 系统中，如果是需要精确查询的内容，应定义为 CHAR 类型（或是具有相似特性的 VCHAR）。

例如，一般对保存文件名的字段，应将它定义为 CHAR 型，这样，我们就可以对它进行精确匹配检索。



记录序号	序号	文件名	字段名
1	200	Car0001.jpg	
2	231	Car2001.bmp	
3	350	Cat001.jpg	
4	588	Cat002.bmp	
5	600	Car0001.jpg	

如上图示，在表（表名为“TEST”）中，其中“文件名”字段是 CHAR 字段。我们可以对文件名进行多种方式的查询：

- 精确查找。**查询 SQL 语句如下：
SELECT * FROM TEST 文件名=' Car0001.jpg'
则返回“序号”为 200 的一条记录。
在 KBase 系统中，所有检索都是大小不敏感的，因此如下 SQL 语句：
SELECT * FROM TEST 文件名=' CAR0001.JPG'
同样命中“序号”为 200 的记录。
- 前方一致查找。**检索词加上？结束。查询 SQL 语句如下：
SELECT * FROM TEST 文件名=' Car?'
则返回“序号”为 200、231 的二条记录，即查出所有“文件名”中以子串‘Car’开始的内容。
语法要求是，以？为检索内容的后缀组合成检索词。
- 包含查找。**示例查询 SQL 语句如下：
SELECT * FROM TEST 文件名=' ?001'
则返回“序号”为 200、231、350 的三条记录，即查出所有“文件名”中包

含子串'001'。

语法要求是，以?为检索内容的前缀组合成检索词。

注意：这种 CHAR 字段的包含查找采用的是遍历记录的方式，如果表中记录数很大，性能将受到严重限制，应谨慎使用！需要这种使用方式时，可采用后面说明的 STRCHAR 字段类型。

4. **位置查找。**查询 SQL 语句如下：

SELECT * FROM TEST 文件名=' Car*bmp'

则返回“序号”为 231 的一条记录。

位置查找还有两种更为精确的用法，如：

SELECT * FROM TEST 文件名=' Car0001?.jpg'

则返回“序号”为 200、600 的二条记录。

SELECT * FROM TEST 文件名=' C???001.jpg'

则返回“序号”为 200 的一条记录。而不会命中“序号”为 350 的记录。

语法要求是，以位置描述标记分隔两个字串 STR1 和 STR2 组合成检索词，并且具有前方一致查找的特性，命中记录的相应字段必须以 STR1 开始，区别是还必须包含 STR2。其中：

STR1*STR2 表示 STR1 与 STR2 相隔 0 个以上的字符；

STR1?STR2 表示 STR1 与 STR2 相隔 1 个字符；

STR1?...?STR2 表示 STR1 与 STR2 相隔 n 个字符，n 等于表达式中?号的个数；

注意：这里字符的意义同 GB18030 标准，而不是指字节。一个 ASCII 是一个字符（占一字节），一个汉字也是一个字符（占两字节）。

5. **比较查找。**对 CHAR 字段，支持所有类型的比较查找，如 > 大于、< 小于、>= 大于等于、<= 小于等于。例如：

SELECT * FROM TEST 文件名>' Car'

则返回“序号”为 350、588 的二条记录。

同前，需要注意的是，字符串的比较是忽略大小写的。

对 CHAR 字段，系统提供几种索引方式，可是无索引（NON），正常索引（NORMAL），唯一索引（UNIQUE）。如果是无索引，只能支持精确查找，对正常索引和唯一索引都支持前面提到的所有检索方式。正常索引和唯一索引在使用上是一致的，其差别仅体现在内在性能上。唯一索引更适合不重复的字段。

VCHAR

同 CHAR 字段，这种数据类型，也是用于存储的字符串，但它是变长的，不需用户来设定最大长度，可用于存储不超过 254 个字符的串。其用途、特性与 CHAR 完全一致，甚至，在检索方式及性能上也完全一致。两者间的唯一差别是在数据的存放方式上，存放方式的差异带来存取效率的差别。

通常情况下，VCHAR 会比 CHAR 更节省磁盘空间，尤其是在不同记录数据长度差别很大的时候。但对 VCHAR 的存取相对于 CHAR 来说更费时。系统对 CHAR

有更高的存取速度，更优的缓存效率。

具体使用参见 CHAR 的说明，这里不再赘述。

MVCHAR

MVCHAR，从名称来看，就知道它是 VCHAR 的孪生弟兄。其特性与 VCHAR 有很多相同的地方，但差别也很多。

MVCHAR 即是 Muti-VCHAR，多值变长串，也就是说，一条记录可以有多个值。多个不同的值之间用分隔符分隔。

MVCHAR 分隔符可以为：

半角字符（ASCII 码）： \ @ , ;

全角字符（GB18030字符）：, 、 : @ ¥

	记录序号	篇名	专题代码
<input type="checkbox"/>	1	百公里油耗仪设计	C023
<input type="checkbox"/>	2	色彩传感器在节水农业中的应用研究	C024;C025
<input type="checkbox"/>	3	基于BJ212半主动悬挂的可控性与...	C023
<input type="checkbox"/>	4	小功率电动机高精度数显机械功率...	C029
<input type="checkbox"/>	5	垂直辅助线平面法求相贯线	C021
<input type="checkbox"/>	6	2BJ-C型浇水播种机的研究	C025
<input type="checkbox"/>	7	车用液化石油气(LPG)特性与改装...	C023

如上示例的表（表名为CJFD2000）中，“专题代码”即是一个MVCHAR字段，这里用的分隔符是分号(;)。检索及相应结果示例如下：

当检索SQL语句为：

SELECT * FROM CJFD2000 WHERE 专题代码='C025'

结果如下：

	记录序号	篇名	专题代码
<input type="checkbox"/>	1	色彩传感器在节水农业中的应用研究	C024;C025
<input type="checkbox"/>	2	2BJ-C型浇水播种机的研究	C025

当检索SQL语句为：

SELECT * FROM CJFD2000 WHERE 专题代码='C024'

结果如下：

记录序号	篇名	专题代码
1	色彩传感器在节水农业中的应用研究	C024;C025

由上可以看出，“篇名”为“色彩传感器在节水农业中的应用研究”的记录，它的专题代码字段有两个不同的值‘C024’和‘C025’。

如果想查询“专题代码”即为‘C024’又为‘C025’的记录，用如下SQL语句：

SELECT * FROM CJFD2000 WHERE 专题代码='C024;C025'

是无法达成目的，因为系统认为你在查一个值为‘C024;C025’的专题代码。正确的SQL语句如下：

SELECT * FROM CJFD2000 WHERE 专题代码='C024' AND 专题代码='C025'

当前，系统没有限定一个MVCHAR字段最多可对应多少个字段值，只限定了MVCHAR字段总长最大可达到4096字节。这4096字节包含所有字段值及其分隔符的总长。

INTEGER 和 NUM

同传统数据库，KBase 系统也提供几种常用的数值型字段，INTEGER 即是其一。INTEGER，整型数据。这种数据类型的字段一般存储的是计数、数量、年龄等。整型数据也被频繁用于保存 ID 号如客户编号、订单号。

INTEGER 本质上与 CHAR 类似，具有单一数值，可以精确查询，也可进行比较查询。其查询语法与 CHAR 一致。只是 INTEGER 型数据不能进行前方一致的查找、包含查找及位置查找，这几种查找对数值型数没有意义。

如前面 CHAR 中的示例，表 TEST 的“序号”字段即是一个 INTEGER 型字段。如：

SELECT * FROM TEST 序号=350

则返回第 3 条记录。

注意，在 KBase 系统中，数据的表达式书写形式相对比较宽松。上述 SQL 语名也可写为：

SELECT * FROM TEST 序号='350'

实际上，几乎所有数据类型，都可以用单引号（'Field Value'）引起来表达，也可以用双引号（"Field Value"）引起来表达，在不发生歧义的情况下，也可以不用引号。这里单引号、双引号都必须是 ASCII，它们作为特殊符号出现，而不是中文全角的标点引号。

INTEGER 同 CHAR 一样，也需要指明长度，长度可定义为 1 字节到 12 字节。与一般传统数据库不一样，KBase 系统中 INTEGER 的精度由其长度决定。其最大可能的数据范围是 (-2147483648, 2147483647)。如果定义长度为 4，则可能范围是 (-999, 9999)，这时如果试图存入一个超出范围的数据，系统会自动进行截断，例如存入的数据是 -987654，则会截断为 -987，这样，你就不会得到想要的结果，因此一定要注数据的定义及允许的数据范围。

NUM 为浮点数，它有着比整数更大的数据范围，但在计算中会产生四舍五入的误差。它的使用方法与 INTEGER 完全一致，不同的是数据的表达式及数据范围。其长度可定义为 1 字节到 32 字节，

其数据范围可达(-3.4028235e+038,3.4028235e+038)，数据也可用科学记数法表达。

AUTO

AUTO 是自动增加的 INTEGER 字段，一般在表中用作每条记录的唯一标识。其在索引、检索属性上与 INTEGER 完全一致。

DATE 和 TIME

kBase 系统提供 DATE 和 TIME 两种数据类型来处理时间相应的数据。这两种数值型数据类型基本用法与 INEGER 用法类似，只是在表达方式上存在差别。

DATE 需定义长度，可在 8 字节到 32 字节之间选择。TIME 则是在 14 字节到 32 字节间选择。

DATE, 表示日期，可能的格式有：

格式一：yyyy/mm/dd 例如 2002/01/23，表示 2002 年 1 月 23 日；

格式二：yyyy:mm:dd 例如 2002:01:23，也表示 2002 年 1 月 23 日；

格式三：yyyy-mm-dd 例如 2002-01-23，同样表示 2002 年 1 月 23 日；

格式四：yyyy.mm.dd 例如 2002.01.23，同样表示 2002 年 1 月 23 日；

格式五：yyyy,mm,dd 例如 2002,01,23，同样表示 2002 年 1 月 23 日；

格式六：yyyy 年 mm 月 dd 日 例如 2002 年 01 月 23 日, 也是。

格式七：yyyymmdd 例如 20020123 日, 也是表示 2002 年 1 月 23 日。

并且，格式一到格式六中，月份和日子可以用一位数表示，而年份还可以用两位数表示，记为 YY

如 02/1/23，表示 2002 年 1 月 23 日

如 74/2/1，表示 1974 年 2 月 1 日

TIME, 表示时间，其格式由 日期 + 分隔符 + 时间 组成，每部分独立
其中日期部分与 DATE 完全一致。

时间部分格式如下：

格式一：hh-mm-ss hh 表示小时，mm 表示分，ss 表示秒，下同

格式二：hh:mm:ss

格式三：hh/mm/ss

格式四：hh 时 mm 分 ss 秒

格式五：hhmmss

前四种格式可以用一位数表示时分秒

分隔符为 空格或逗号(,) 冒号(:) 连字符(-) 斜杠(/)

例如 2002-01-23:08-30-15，表示 2002 年 1 月 23 日 8 点 30 分 15 秒。

02/1/23, 08:30:15，表示 2002 年 1 月 23 日 8 点 30 分 15 秒。

20020123083015，表示 2002 年 1 月 23 日 8 点 30 分 15 秒。

需要注意的是，DATE 和 TIME 两种类型的数据，其表达式都必须用引号将数据引起来，否则，结果无法预期。

SQL 语句： SELECT * FROM TEST 生日= 2000-06-17

相当于： SELECT * FROM TEST 生日= 2000 NOT 生日=06 NOT 生日=17

显然，不是检索原意；

而下面的 SQL 语句，将导致系统语法解析报错

SELECT * FROM TEST 生日= 2000/06/17

正确写法应如： SELECT * FROM TEST 生日=' 2000/06/17'

DATE 和 TIME 类型还支持模糊查找，如：

SELECT * FROM TEST 生日 % ' 2000/06'

相当于 SELECT * FROM TEST 生日>=' 2000/06/1' AND 生日<=' 2000/06/31'

STRING 和 STRCHAR

STRING 和 STRCHAR 是文本串数据类型，是 KBase 系统中最小型的处理文本数据的字段类型，其最大长度仅有 254 字节。如果长度超过 254 字节，系统会自动截短为 254 字节。

STRING 和 STRCHAR 区别是，STRING 在做索引时，是分词后索引的。而 STRCHAR 是按字切分索引的，“按字切分索引”通常称之为不分词索引。

它们具有文本型数据分词索引的特点，与其它文本型数据类型的区别是，它不分句，更不分段，即它仅相当于其它文本型数据中的一个句子。因此，如果一个字段只有少量文本数据，采用 STRING 或 STRCHAR，将可能获得最佳的检索性能。

它们与其它文本型数据类型相比，还有一个细微的差异是，STRING 或 STRCHAR 在分词（或分字）后，直接完全索引；而其它文本型数据类型则是在分词（或分字）后，先过滤了停用字词之后，再进行索引。这一点，对检索有时会有一些微妙的影响，例如：

记录序号	序号	篇名	题名
1	200	32枚奥运金牌的背后	32枚奥运金牌的背后
2	231	机场建设费征收细则出台	以航段计费标准不变
3	350	国务院下发十一放假通知	国庆期间放假七天
4	588	2004雅典奥运会开幕	2004雅典奥运会闭幕
5	600	从数字看雅典奥运冠军	12秒91刘翔飞身成英雄

上图所示的表（表名为“TEST”）中，“篇名”字段类型为 STRING，“题名”字段类型是 TEXT，用 SQL 语句：

SELECT * FROM TEST 篇名='的'

能够检索到第一条记录；但是，SQL 语句：

SELECT * FROM TEST 题名='的'

就不能检索到第一条记录，因为在索引时，'的'是停用词被过滤掉了。关于停用词更详细的说明，可参考 TEXT 字段类型的说明。

STRING 和 STRCHAR 这两种字段类型，支持两类匹配检索。

1. 包含查找。这是最基本的文本型数据的检索方式，如前类所示即是。与 CHAR 不同，文本型数据的包含查找不用加上问号 (?) 作内容前缀。

因为文本数据的索引是基于分词的，查找的内容应是一个或多个连续的词汇，如果要查找的内容不是一个准确的概念，仅是其中的一个子串，也可能无法检索到。例如：

SELECT * FROM TEST 篇名='奥运金'

虽然，第一条记录中有'奥运金'这个子串，上述 SQL 查询语句也无法索到这条记录，这是因为“32 枚奥运金牌的背后”索引时是先分词为“32/枚/奥运/金牌/的/背后”，这样，我们只检索其中的一个或多个连续的词。

分词的原则请参考相关说明。

2. 位置查找。用法上与类似于 CHAR 字段类型，如查询 SQL 语句如下：

SELECT * FROM TEST 篇名='机场*细则'

则返回“序号”为 231 的一条记录。

同样，位置查找也可以用一个或多个问号 (?) 来准确限定待查词间距不可超过 n（等于? 的个数）词（或字）。而不是 CHAR 字段类型中，是准确相隔 n 个字符。位置查找示例如下：

SELECT * FROM TEST 篇名='奥运???背后'

是查找篇名中有'奥运'和'背后'两个词，并且，'奥运'在'背后'前，两者最多间隔三个词的记录。因此也会命中“序号”为 231 的记录

语法要求是，以位置描述标记分隔两个字串 STR1 和 STR2 组合成检索词必须包含 STR1 和 STR2，并且，STR1 在 STR2 前面，相隔不超过一定数量的字或词。其中：

STR1*STR2 表示 STR1 与 STR2 相隔任意个以上的字或词，STR1 在前；

STR1?...?STR2 表示 STR1 与 STR2 相隔 0 到 n 个字符，n 等于表达式中? 的个数，STR1 在前；

所有文本型字段，因为分词索引的原因，不同于数值型的字段，都不提供比较查找。

在这里，还有个细微之处要注意。前例中，

```
SELECT * FROM TEST 篇名=' 机场*细则'
```

可以命中记录，但：

```
SELECT * FROM TEST 篇名=' 细则*机场'
```

不能命中记录，因为没有‘细则’在‘机场’前的数据。但是

```
SELECT * FROM TEST 篇名= 细则*机场
```

可以命中记录。实际上，上一 SQL 语句标准写法是：

```
SELECT * FROM TEST 篇名= '细则' * '机场'
```

相当于：

```
SELECT * FROM TEST 篇名= '细则' AND 篇名='机场'
```

因此，它实际上并不是位置查找，而是一个逻辑组合条件的查询语句，即两个包含查找的 AND 运算。有关运算符*号的使用规则请参考 KSQL 语法的说明。

为了减少类似问题，建议在书写 SQL 词句时，尽量用引号将检索关键字引起来。

TEXT 和 TEXTCHAR

TEXT 和 TEXTCHAR 是 KBase 系统中最早提供的大文本数据类型，在最初的版本中就已经出现。TEXT 与 TEXTCHAR 不用定义长度，其最大长度由系统自动设置，最大可达 16M 字节。这相当于一条记录单一字段容量达到 8 百万汉字，20-30 万字的书可存放 30 本左右。这样，就充许数据库中存储整个文档、产品说明、论文、文献甚至整本书。

TEXT 和 TEXTCHAR 区别是，TEXT 在做索引时，是分词后索引的。而 TEXTCHAR 则是不分词的。

包括后面提到的文本型数据类型，都提供了丰富的全文检索手段。主要的检索方法包括：

1. **包含查找**。这是最基本的文本型数据的检索方式，用法与前面的 STRING 字段类型完全一样。
2. **位置查找**。对比 STRING 字段类型，TEXT 等大文本字段内供了更丰富的位置查找方法。对于几千字几万字的内容，是很有必要进行更准确的位置查找。其技术上的内在原因是，TEXT 等大文本字段在作索引时，不仅记录了每个词在句中的位置序号，还记录了词所在的段落序号，句子序号。而 STRING 字段类型总是作为单一句子来处理。SQL 查询规则如下：

```
SELECT * FROM <表名> WHERE <字段名>=>' STR1 位置描述标记 STR2'
```

例如，要查询表 CJFD2003 的篇名中‘姚明’和‘NBA’出现在同一句中的记录，

SQL 语句如下：

```
SELECT * FROM CJFD2003 WHERE 篇名=' 姚明 # NBA'
```

查询结果如下图：

记录序号	篇名
1	刘永恺漫画NBA——姚明之英雄
2	NBA如何打造“姚明品牌”

这里位置描述标记#表示篇名中有‘姚明’和‘NBA’，并且在‘姚明’和‘NBA’同一句中。全部位置描述标记列表如下：

- ‘STR1 # STR2’ 表示包含 STR1 和 STR2，且 STR1、STR2 在同一句中；
- ‘STR1 % STR2’ 表示包含 STR1 和 STR2，且 STR1 与 STR2 在同一句中，且 STR1 在 STR2 前面；
- ‘STR1 /NEAR N STR2’ 表示包含 STR1 和 STR2，且 STR1 与 STR2 在同一句中，且相隔不超过 N 个词；
- ‘STR1 /PREV N STR2’ 表示包含 STR1 和 STR2，且 STR1 与 STR2 在同一句中，STR1 在 STR2 前面不超过 N 个词；
- ‘STR1 /AFT N STR2’ 表示包含 STR1 和 STR2，且 STR1 与 STR2 在同一句中，STR1 在 STR2 后面不超过 N 个词；
- ‘STR1 /SEN N STR2’ 表示包含 STR1 和 STR2，且 STR1 与 STR2 在同一段中，且相隔不超过 N 个句子；
- ‘STR1 /PRG N STR2’ 表示包含 STR1 和 STR2，且 STR1 与 STR2 相隔不超过 N 段。

注意：前面所列位置描述标记前后都必须至少包含一个空格用以分隔表达式中不同的部分。

3. **词频查找。**有时候，查询人员在查询一个关键词时，可能要求这个关键词在记录中出现很多次，他才认为这条记录是他想要的。这进他可以借助于词频查找。语法如下：

SELECT * FROM <表名> WHERE <字段名>=’ STR \$ N’

其中，N 表示所查关键词 STR 最少出现的次数。

记录序号	篇名
1	非计算机专业学生的计算机教育
2	量子计算机 世界上最快的计算机
3	非计算机专业计算机课程教学的探索
4	计算机软件的版权保护——《计算机软件保护条例》引发的法律思考
5	普及和提高全民的计算机知识水平才是根本 漫谈计算机考试与计算机...
6	“计算机应用基础”“管理系统中计算机应用”课程(上机部分)考核实施细则
7	非计算机专业计算机公共课程教学改革的思考与探索
8	计算机隐喻与计算机英语隐喻
9	高校非计算机专业《计算机文化基础》课程教学中存在的问题与对策
10	计算机“任务驱动模式”教学法和计算机教学中如何实施“任务驱动”

例如，上图即是我们在表 CJFD2003 中检索‘计算机’这个词在篇名中至少出现两次的记录的结果。检索所用的 SQL 语句是：

SELECT * FROM CJFD2003 WHERE 篇名=’ 计算机 \$ 2’

其实，一般我们不会在篇名这种短字段中进行词频查找，而更多的是在全文这种长字段中进行词频查找，这里只是为了满足讲解的需要。

LTEXT、MTEXT 和 MTEXTCHAR

在 TEXT 和 TEXTCHAR 之后，系统又进一步提供了新的全文字段 LTEXT、MTEXT 和 MTEXTCHAR。新的系统中，应尽可能采用这些字段类型，TEXT 和 TEXTCHAR 是为了兼容早期版本而保留的。

在用法上这几种字段与 TEXT、TEXTCHAR 完全一样，其用法请参照前前的说明。其差别体现在性能上。改进后的 LTEXT、MTEXT 和 MTEXTCHAR，在检索速度有显著提高，并且索引的数据量节省了 25% 左右。

LTEXT 和 MTEXT 是分词索引的，而 MTEXTCHAR 是不分词索引的。

LTEXT 字段类型最大长度可达 16M 字节，也是系统自行设定的；MTEXT 和 MTEXTCHAR 最大长度可达 4096 字节。

这里，用户可能看到，改进行全文字段类型中，没有出现与 TEXTCHAR 完全对应的字段，即既能达到 16M 字节，又按不分词方式索引的字段类型。其实在实际应用中，对超过 4096 字节大小的字段，一般是全文字段，对进行全文检索时，有两个指标：查全率和查准率，理论上不分词能做到 100% 的查全率，但远不能人们查准的要求，特别是在海量文献中，不分词往往找出一大堆仅是匹配上查询关键词但与所查概念完全无关的数据。实践证明，分词对提高查准率有很好的帮助。如：“原子组成分子”，如果不分词，当查询“成分”时，就会返回这条数据，如果选择分词类型字段，因它在索引中表现为“原子/组成/分子”，因此就不会返回这条与“成分”不相关的数据。不分词的字段字型往往用在一些如机构名称、人名、期刊名等需要通过只言片语返回一些线索以进一步查询的内容上。但这此字段内容往往不大，不会超过 4096 字节。当然，如有特殊情况，需要使用超长的不分词全文字段，后面将要提到的 MIXTEXTCHAR 等字段也能满足这种需求。

MIXTEXT、MIXTEXTCHAR 和 MIXTEXTAUTO

KBase 系统中引入的 MIXTEXT、MIXTEXTCHAR、MIXTEXTAUTO 这几种新型的文本类型字段，在性能特性上与前面的 TEXT 几乎一致。KBase 系统创新的引入虚字段的概念，使系统中一个字多进行多种索引，也可将多个字段进行混合索引。

虚字段中的这个“虚”字体现在，其字段内容不是来自于专属的数据存储区，而是从表内相关字段或表信息获取，即它不占实现的物理数据存储空间。虚字段却有其实际的功能。对 MIXTEXT、MIXTEXTCHAR、MIXTEXTAUTO 这几个字段类型而言，它给对应的字段作一种新的索引，这样，可以实现一个字段的多个索引。

这里以 CNKI 中的中国学术期刊全文数据库为类，其表中有期刊名称字段。在查

询时，用户有时需要进行精确查找，就象 VCHAR 字段类型提供的特性，有时，也可能想进行比较核模糊的查找，查得更多，更全，就象 MTEXTCHAR 字段类型提供的特性。这时，我们可以表中定义一个“中文刊名”字段，其字段类型是 VCHAR，再定义一个“精确刊名”字段，其字段类型是 MIXTEXTCHAR，并将“精确刊名”映射到“中文刊名”上。这样，我们就可以在“精确刊名”进行很严格查询刊名，并在“中文刊名”进行象包含等较模糊的查找。这时，因“精确刊名”和“中文刊名”在物理上拥有同一份数据，因此，修改“精确刊名”的同时，也同步更新了“中文刊名”，“中文刊名”内容总与“精确刊名”完全一致。

这几种字段还可以进行一对多的映射，即一个字段，其数据内容来自于多个字段。这里还是以 CNKI 中的中国学术期刊全文数据库为类，其表中“篇名”、“关键词”、“中文摘要”、“英文篇名”、“英文关键词”、“英文摘要”等字段。用户在查询一个概念时，往往想在这几个字段同时检索，如查“超导”，可能的 SQL 语句如下：
`SELECT * FROM 期刊全文数据库 WHERE 篇名='超导' OR 关键词='超导' OR 中文摘要='超导' OR 英文篇名='超导' OR 英文关键词='超导' OR 英文摘要='超导'`

这样，数据库检索引擎会在 6 个不同的索引中进行检索，并将检索结果进行逻辑运算得到最终结果返回给用户，这样，会显著降低系统性。

为解决这个问题，我们可以定义一个新的字段，如“主题”字段，将其定义为 MIXTEXT 字段，将它映射到前述的 6 个字段上，这样，SQL 语句：

`SELECT * FROM 期刊全文数据库 WHERE 主题='超导'`

即可返回上述所有 6 个字段中包含‘超导’的记录，并且，由于在一个索引中进行查找，大大提高了检索速度，检索速度的提高几乎是线性的。

三种字段的区别是，MIXTEXT 在做索引时，是分词后索引的；MIXTEXTCHAR 则是不分词的；而 MIXTEXTAUTO 则是根据所映射的字段来决定是否分词，如果其所映射的字段中有一个不分词的字段就不分词索引，否则，分词索引。

TNAME 和 DNAME

这也是两种虚字段，TNAME 的返回的内容总是表的名字，而 DNAME 返回的则总是表的显示名称。

VSM

VSM 是 KBase 系统新引入的向量字段类型，它在多种场合有着广泛的应用。

其字段内容格式为：‘ $W_1 : F_1 , W_2 : F_2 , \dots , W_n : F_n$ ’

W_i 是特征词， F_i 是权重。

基于向量运算，我们可以利用它来做相似检索、复杂的权限控制等。

SOB、LOB 和 DOB

这三个字段用于处理无结构字节流，如存储压缩视频图像、音频数据、可执行代码等数据。**SOB** 处理小对象，最大长度为 254 字节，**LOB** 处理大对象，最大长度为 16M 字节，而 **DOB** 有能力则处理更大的对象，长度可达 2G 字节。

这三种字段都没有索引，一般不用于检索。

SOB 和 **LOB** 字段类型，其读定方法与其它字段一致，写入或修改必须是一次性的；而 **DOB** 字段则支持完全的流式读写。