Atitit.数据索引 的种类以及原理实现机制 索引常用的存储结构

[1. 索引的分类 1](#_Toc753)

[1.1. 按照存储结构划分btree,hash,bitmap,fulltext 1](#_Toc4341)

[1.2. 索引的类型 按查找方式分，两种，分块索引 vs编号索引 1](#_Toc26687)

[1.3. 顺序索引  vs 散列索引 2](#_Toc25295)

[1.4. 按索引与数据的查找顺序可分为 正排与倒排索引 2](#_Toc9351)

[1.5. 单列索引与多列索引 复合索引 2](#_Toc29460)

[1.6. 分区索引和全局索引  2](#_Toc5249)

[1.7.  Trie树一般指字典树 又称单词查找树，Trie树 2](#_Toc28784)

[1.8. 稠密索引 vs 稀疏索引 3](#_Toc9648)

[1.9. 多级索引 vs 单击索引 3](#_Toc19899)

[1.10. 索引模式extent和blockmap 3](#_Toc21863)

[2. 索引建立，更新的流程使用触发更新索引的事件 4](#_Toc31086)

[3. ISAM算法 索引顺序存取方法”（Indexed Sequential Access Method） 索引常用的存储结构 B树文件 叫做“，缩写为。 4](#_Toc27877)

[4. 索引文件的合并问题 5](#_Toc463)

[5. 参考 5](#_Toc2896)

# 索引的分类

Uniq

全文索引

Norma

Hash 索引（编号索引）

l

## 按照存储结构划分btree,hash,bitmap,fulltext

## 索引的类型 按查找方式分，两种，分块索引 vs编号索引

一种是分块》分块类型。。一种是不分块,编号顺序排列类型

## 顺序索引  vs 散列索引

## 按索引与数据的查找顺序可分为 正排与倒排索引

倒排索引

## 单列索引与多列索引 复合索引

## [分区索引和全局索引](http://www.baidu.com/link?url=KNgkbiJNS2sCOd-TkWjbEUtgK1HZD70MXwk7YapqPxpKnFOqAeQWK8B7NOSA36T9kEDHv7dsB0n8mQBj8tSuYq" \t "http://www.baidu.com/_blank)

## Trie树一般指字典树 又称单词查找树，[Trie树](http://baike.baidu.com/view/1436495.htm" \t "http://baike.baidu.com/_blank)

 Trie树一般指字典树 又称单词查找树，[Trie树](http://baike.baidu.com/view/1436495.htm" \t "http://baike.baidu.com/_blank)，是一种[树形结构](http://baike.baidu.com/view/540464.htm" \t "http://baike.baidu.com/_blank)，是一种哈希树的变种。典型应用是用于统计，排序和保存大量的[字符](http://baike.baidu.com/view/263416.htm" \t "http://baike.baidu.com/_blank)串（但不仅限于字符串），所以经常被搜索引擎系统用于文本词频统计

。它的优点是：利用字符串的公共前缀来减少查询时间，最大限度地减少无谓的字符串比较，查询效率比[哈希](http://baike.baidu.com/view/99075.htm" \t "http://baike.baidu.com/_blank)树高。

Trie树，又称字典树，单词查找树或者前缀树，是一种用于快速检索的多叉树结构，如英文字母的字典树是一个26叉树，数字的字典树是一个10叉树。

Trie一词来自re**trie**ve，发音为/tri:/ “tree”，也有人读为/traɪ/ “try”。

Trie树可以利用字符串的公共前缀来节约存储空间。如下图所示，该trie树用10个节点保存了6个字符串tea，ten，to，in，inn，int：

在该trie树中，字符串in，inn和int的公共前缀是“in”，因此可以只存储一份“in”以节省空间。当然，如果系统中存在大量字符串且这些字符串基本没有公共前缀，则相应的trie树将非常消耗内存，这也是trie树的一个缺点。

Trie树的基本性质可以归纳为：

（1）根节点不包含字符，除根节点意外每个节点只包含一个字符。

（2）从根节点到某一个节点，路径上经过的字符连接起来，为该节点对应的字符串。

（3）每个节点的所有子节点包含的字符串不相同。

## 稠密索引 vs 稀疏索引

稠密索引：文件中的每个搜索码值都有一个索引项。

稀疏索引：只为搜索码中的某些值建立索引项

## 多级索引 vs 单击索引

B+树索引结构就是多级索引结构中的一种，因其较快的查询速度，在文件系统中广泛用作建立索引。且其能在插入或删除数据时（虽然这两个过程非常复杂，但是对磁盘块I/0次数少，可以接受），保持器执行效率

## 索引模式extent和blockmap

先说blockmap，这是用在ufs、sco htfs、ext2/3、reiserfs上的索引模式，意思是每个文件的分配块都有一个索引与之对应，是一对一的索引关系

  再谈extent方式，用于ntfs、Vxfs、jfs、ext4等文件系统。其实现方式和blockmap不同的是，索引是按分配的片断记录，只记起始块、连续块数、及文件内部块位置(NTFS叫VCN的东东

果这个文件分配时是连续分配的，只需记录3个数字:（文件内部块号：0，文件系统分配起始块：x,连续块数:1024)，不再需要1024个索引空间来描述。当然，如果这个文件有多个碎片组成，则需要多条记录来实现。

extent其优点是索引空间占有率较少，连续读写时会有优势，但缺点是算法复杂度略高。比如一个文件由100个片断(碎片)组成，需要定位到文件内部10M的偏移，则需要二叉查找属于哪个片断，再根据片断的起始地址计算到具体的分配块地址，才可以把数据读出来。如果像NTFS一样，片断本身都由变长方式实现，则内核判断上就更麻烦，文件系统崩溃的可能性也就很大了。

    以个人来看，纯粹的blockmap方式，在现在文件都很大的应用环境下，稍显不太适应，慢慢的应用会越来越少。而类似ntfs的变长extent记录方式，则感觉有点得不偿失，这似乎和微软不考虑性能，不考虑健壮性，只考虑功能实现的理念相吻合。

# 索引建立，更新的流程使用触发更新索引的事件

1 大量数据插入的时候，考虑先删除索引，然后重建索引。这样做的缺点是业务不能同时进行

说明索引是类似与触发器，每增加一条记录触发一次创建立索引的流程

# ISAM算法 索引顺序存取方法”（Indexed Sequential Access Method） 索引常用的存储结构 B树文件 叫做“，缩写为。

所谓索引，就是以某个字段为关键字的B树文件。假定有一张”雇员表”，包含了员工号（主键）和姓名两个字段。可以对姓名建立索引文件，该文件以B树格式对姓名进行储存，每个姓名后面是其在数据库中的位置（即第几条记录）。查找姓名的时候，先从索引中找到对应第几条记录，然后再从表格中读取。

这种索引查找方法，叫做“索引顺序存取方法”（Indexed Sequential Access Method），缩写为ISAM。它已经有多种实现（比如C-ISAM库和D-ISAM库），只要使用这些代码库，就能自己写一个最简单的数据库。

# 索引文件的合并问题

当索引文件越来越大时候,就需要分布式存储在多个增量索引文件上..到时合并或者不合并.....

或者使用2进制方式增量存储..

# 参考

paip.索引的种类以及实现attilax 总结 - attilax的专栏 - 博客频道 - CSDN.NET.htm

字典树\_百度百科.htm （有代码实现

Atitit.数据索引 的种类以及原理实现机制

Atitit.索引的种类

文件系统中的索引：B+树索引结构 - 莫语的日志 - 网易博客.html