SqlServer索引的原理与应用

http://www.cnblogs.com/knowledgesea/p/3672099.html

索引的概念

索引的用途：我们对数据查询及处理速度已成为衡量应用系统成败的标准，而采用索引来加快数据处理速度通常是最普遍采用的优化方法。

索引是什么：数据库中的索引类似于一本书的目录，在一本书中使用目录可以快速找到你想要的信息，而不需要读完全书。在数据库中，数据库程序使用索引可以重啊到表中的数据，而不必扫描整个表。书中的目录是一个字词以及各字词所在的页码列表，数据库中的索引是表中的值以及各值存储位置的列表。

索引的利弊：查询执行的大部分开销是I/O，使用索引提高性能的一个主要目标是避免全表扫描，因为全表扫描需要从磁盘上读取表的每一个数据页，如果有索引指向数据值，则查询只需要读少数次的磁盘就行啦。所以合理的使用索引能加速数据的查询。但是索引并不总是提高系统的性能，带索引的表需要在数据库中占用更多的存储空间，同样用来增删数据的命令运行时间以及维护索引所需的处理时间会更长。所以我们要合理使用索引，及时更新去除次优索引。

数据表的基本结构

一个新表被创建之时，系统将在磁盘中分配一段以8K为单位的连续空间，当字段的值从内存写入磁盘时，就在这一既定空间随机保存，当一个 8K用完的时候，数据库指针会自动分配一个8K的空间。这里，每个8K空间被称为一个数据页（Page），又名页面或数据页面，并分配从0-7的页号，每个文件的第0页记录引导信息，叫文件头（File header）；每8个数据页（64K）的组合形成扩展区（Extent），称为扩展。全部数据页的组合形成堆（Heap）。

SQLS规定行不能跨越数据页，所以，每行记录的最大数据量只能为8K。这就是char和varchar这两种字符串类型容量要限制在8K以内的 原因，存储超过8K的数据应使用text类型，实际上，text类型的字段值不能直接录入和保存，它只是存储一个指针，指向由若干8K的文本数据页所组成 的扩展区，真正的数据正是放在这些数据页中。

页面有空间页面和数据页面之分。

当一个扩展区的8个数据页中既包含了空间页面又包括了数据或索引页面时，称为混合扩展（Mixed Extent）,每张表都以混合扩展开始；反之，称为一致扩展（Uniform

Extent），专门保存数据及索引信息。
表被创建之时，SQLS在混合扩展中为其分配至少一个数据页面，随着数据量的增长，SQLS可即时在混合扩展中分配出7个页面，当数据超过8个页面时，则从一致扩展中分配数据页面。
空间页面专门负责数据空间的分配和管理，包括：PFS页面（Page free space）：记录一个页面是否已分配、位于混合扩展还是一致扩展以及页面上还有多少可用空间等信息；GAM页面（Global allocation map）和SGAM页面(Secodary global allocation map)：用来记录空闲的扩展或含有空闲页面的混合扩展的位置。SQLS综合利用这三种类型的页面文件在必要时为数据表创建新空间；
数据页或索引页则专门保存数据及索引信息，SQLS使用4种类型的数据页面来管理表或索引：它们是IAM页、数据页、文本/图像页和索引页。
在WINDOWS中，我们对文件执行的每一步操作，在磁盘上的物理位置只有系统（system）才知道；SQL SERVER沿袭了这种工作方式，在插入数据的过程中，不但每个字段值在数据页面中的保存位置是随机的，而且每个数据页面在“堆”中的排列位置也只有系统 （system）才知道。
这是为什么呢？众所周知，OS之所以能管理DISK，是因为在系统启动时首先加载了文件分配表：FAT（File Allocation Table），正是由它管理文件系统并记录对文件的一切操作，系统才得以正常运行；同理，作为管理系统级的SQL SERVER,也有这样一张类似FAT的表存在，它就是索引分布映像页：IAM（Index Allocation Map）。
IAM的存在，使SQLS对数据表的物理管理有了可能。
IAM页从混合扩展中分配，记录了8个初始页面的位置和该扩展区的位置，每个IAM页面能管理512,000个数据页面，如果数据量太 大，SQLS也可以增加更多的IAM页，可以位于文件的任何位置。第一个IAM页被称为FirstIAM,其中记录了以后的IAM页的位置。
数据页和文本/图像页互反，前者保存非文本/图像类型的数据，因为它们都不超过8K的容量，后者则只保存超过8K容量的文本或图像类型数据。而索 引页顾名思义，保存的是与索引结构相关的数据信息。了解页面的问题有助我们下一步准确理解SQLS维护索引的方式，如页拆分、填充因子等。

页分裂

一半的数据将保留在老页面，而另一半将放入新页面，并且新页面可能被分配到任何可用的页。所以，频繁页分裂，后果很严重，将使物理表产生大量数据碎片，导致直接造成I/O效率的急剧下降，最后，停止SQLS的运行并重建索引将是我们的唯一选择!

填充因子

索引的一个特性，定义该索引每页上的可用空间量。FILLFACTOR（填充因子）适应以后表数据的扩展并减小了页拆分的可能性。填充因子是从0到100的百分比数值，设为100时表示将数据页填满。只有当不会对数据进行更改时(例如 只读表中)才用此设置。值越小则数据页上的空闲空间越大，这样可以减少在索引增长过程中进行页分裂的需要，但这一操作需要占用更多的硬盘空间。填充因子指定不当，会降低数据库的读取性能，其降低量与填充因子设置值成反比。

索引的分类

SQL SERVER中有多种索引类型。

按存储结构区分：“聚集索引（又称聚类索引，簇集索引）”，“分聚集索引（非聚类索引，非簇集索引）”

按数据唯一性区分：“唯一索引”，“非唯一索引”

按键列个数区分：“单列索引”，“多列索引”。

聚集索引

聚集索引是一种对磁盘上实际数据重新组织以按指定的一列或多列值排序。像我们用到的汉语字典，就是一个聚集索引，比如要查“张”，我们自然而然就翻到字典的后面百十页。然后根据字母顺序跟查找出来。这里用到微软的平衡二叉树算法，即首先把书翻到大概二分之一的位置，如果要找的页码比该页的页码小，就把书向前翻到四分之一处，否则，就把书向后翻到四分之三的地方，依此类推，把书页续分成更小的部分，直至正确的页码。

由于聚集索引是给数据排序，不可能有多种排法，所以一个表只能建立一个聚集索引。科学统计建立这样的索引需要至少相当与该表120%的附加空间，用来存放该表的副本和索引中间页，但是他的性能几乎总是比其它索引要快。

由于在聚集索引下，数据在物理上是按序排列在数据页上的，重复值也排在一起，因而包含范围检查（bentween,<,><=,>=）或使用group by 或order by的查询时，一旦找到第一个键值的行，后面都将是连在一起，不必在进一步的搜索，避免啦大范围的扫描，可以大大提高查询速度。

非聚集索引

sqlserver默认情况下建立的索引是非聚集索引，他不重新组织表中的数据，而是对每一行存储索引列值并用一个指针指向数据所在的页面。他像汉语字典中的根据‘偏

旁部首’查找要找的字，即便对数据不排序，然而他拥有的目录更像是目录，对查取数据的效率也是具有的提升空间，而不需要全表扫描。

一个表可以拥有多个非聚集索引，每个非聚集索引根据索引列的不同提供不同的排序顺序。

创建索引

语法

复制代码

```
CREATE [UNIQUE] [CLUSTERED| NONCLUSTERED ]
INDEX index_name ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )
[with[PAD_INDEX][[,]FILLFACTOR=fillfactor]
[[,]IGNORE_DUP_KEY]
[[,]DROP_EXISTING]
[[,]STATISTICS_NORECOMPUTE]
[[,]SORT_IN_TEMPDB]
]
[ ON filegroup ]
```

复制代码

CREATE INDEX命令创建索引各参数说明如下：

UNIQUE：用于指定为表或视图创建唯一索引，即不允许存在索引值相同的两行。

CLUSTERED：用于指定创建的索引为聚集索引。

NONCLUSTERED：用于指定创建的索引为非聚集索引。

index_name：用于指定所创建的索引的名称。

table：用于指定创建索引的表的名称。

view：用于指定创建索引的视图的名称。

ASC|DESC：用于指定具体某个索引列的升序或降序排序方向。

Column：用于指定被索引的列。

PAD_INDEX：用于指定索引中间级中每个页（节点）上保持开放的空间。

FILLFACTOR = fillfactor：用于指定在创建索引时，每个索引页的数据占索引页大小的百分比，fillfactor的值为1到100。

IGNORE_DUP_KEY：用于控制当往包含于一个唯一聚集索引中的列中插入重复数据时SQL Server所作的反应。

DROP_EXISTING：用于指定应删除并重新创建已命名的先前存在的聚集索引或者非聚集索引。

STATISTICS_NORECOMPUTE：用于指定过期的索引统计不会自动重新计算。

SORT_IN_TEMPDB：用于指定创建索引时的中间排序结果将存储在 tempdb 数据库中。

ON filegroup：用于指定存放索引的文件组。

例子：

复制代码

```
--表bigdata创建一个名为idx_mobiel的非聚集索引，索引字段为mobiel
create index idx_mobiel
on bigdata(mobiel)


--表bigdata创建一个名为idx_id的唯一聚集索引，索引字段为id
--要求成批插入数据时忽略重复值，不重新计算统计信息，填充因子为40
create unique clustered index idx_id
on bigdata(id)
with pad_index,
fillfactor=40,
ignore_dup_key,
statistics_norecompute
```

复制代码

管理索引

复制代码

```
Exec sp_helpindex BigData   --查看索引定义

Exec sp_rename 'BigData.idx_mobiel','idx_big_mobiel'  --将索引名由'idx_mobiel'改为'idx_big_mobiel'

drop index BigData.idx_big_mobiel  --删除bigdata表中的idx_big_mobiel索引

dbcc showcontig(bigdata,idx_mobiel) --检查bigdata表中索引idx_mobiel的碎片信息
```

```
dbcc indexdefrag(Test,bigdata,idx_mobiel)  --整理test数据库中bigdata表的索引idx_mobiel上的碎片

update statistics bigdata  --更新bigdata表中的全部索引的统计信息
```

复制代码

索引的设计原则

对于一张表来说索引的有无和建立什么样的索引，要取决与where字句和Join表达式中。

一般来说建立索引的原则包括以下内容：

- 系统一般会给逐渐字段自动建立聚集索引。
- 有大量重复值且经常有范围查询和排序、分组的列，或者经常频繁访问的列，考虑建立聚集索引。
- 在一个经常做插入操作的表中建立索引，应使用fillfactor(填充因子)来减少页分裂，同时提高并发度降低死锁的发生。如果在表为只读表，填充因子可设为100.
- 在选择索引键时，尽可能采用小数据类型的列作为键以使每个索引页能容纳尽可能多的索引键和指针，通过这种方式，可使一个查询必需遍历的索引页面降低到最小，此外，尽可能的使用整数做为键值，因为整数的访问速度最快。