Atitit 索引引擎与算法

目录

[1.1. 三种索引算法BTree和Hash fulltxt 1](#_Toc25966)

[1.2. 索引的触发 1](#_Toc15968)

[1.3. 索引对重建 1](#_Toc32300)

[1.4. 分区后的索引 全局索引 vs 分区索引 1](#_Toc12305)

[1.5. 分区与约束 unique或primary key约束 外键等 2](#_Toc4856)

## 三种索引算法BTree和Hash fulltxt

常用两种索引算法BTree和Hash，两种算法检索方式不一样，对查询的作用也不一样。fulltxt

## 索引的触发

可以使用mq事件机制。。键入数据的时候，推送一个索引建设事件

## 索引对重建

遍历数据，对某个索引重建

## 分区后的索引 全局索引 vs 分区索引

但是，分区表的索引是使用不同存储模型存储的（根据索引是分区索引还是非分区索引）。

非分区索引的优势包括：

* 可以相互独立地对各个索引进行重组
* 能够提高删除索引操作的性能
* 删除各个索引时，空间将立即可供系统使用，而无需进行索引重组

分区索引的优势包括：

* 能够提高数据滚入和滚出性能
* 由于索引进行分区，因此能够减少对索引页的争用
* 每个索引分区均采用索引 B 树结构，这有如下优点：
  + 提高插入、更新、删除和扫描性能，这是因为，索引分区的 B 树所包含的层数通常少于引用表中所有数据的索引
  + 改进分区消除生效时的扫描性能和并行性。尽管分区消除功能既可用于分区索引扫描也可用于非分区索引扫描，但用于分区索引扫描却更为有效，这是因为，每个索引分区都只包含相应数据分区的键。此配置可导致必须扫描的键数和索引页数少于对非分区索引执行的类似查询。

虽然非分区索引始终保留索引列的顺序，但分区索引在某些情况下可能会在各分区之间丢失一些顺序；例如，如果分区列与索引列不匹配，并且将访问多个分区。

可以同时建立分区索引和全局索引，增加查询操作

分区索引分为本地(local index)索引和全局索引(global index)。局部索引比全局索引容易管理, 而全局索引比较快。

全局索引和本地索引是Oracle的功能，MySQL(包括PostgreSQL)只实现了本地索引，并且因为有全局约束的问题，MySQL分区表明确不支持外键，并且主键和唯一键必须要包含所有分区列，否则报错。

？ OLAP系统中多用局部索引，OLTP系统上，全局索引更为常见。可用性角度：局部索引更可用，就算一个索引分区出问题了也不影响其他，而全局索引很可能会成为一个故障点，一旦出现问题则整个索引都不可用。维护性角度：局部索引更好维护更灵活，DBA决定移动一个表分区，只需要重建与维护一个索引分区。对全局索引，很多情况下都需重建。 sql效率：因为局部索引随表分区，可以涉及出最优的执行计划。

## 分区与约束 unique或primary key约束 外键等

局部索引与唯一约束 分区表字段想用unique或primary key约束，一般是使用全局索引来保证唯一性，这是一般的做法。因为局部索引只保证分区内部的键的唯一性，而不能跨分区，如果你的确想用局部索引来保证整个表的唯一性，就得把分区键加到约束当中，也成。如果oracle允许局部索引(不包含约束的情况)就能轻易来保证全表的唯一性，那么所有的update与insert，都得扫每一个分区，这样可用性与可扩展性都会丧失殆尽。