Atitit Atitit sql优化之性能优化细则 attilax总结

[1. 改动较小且立竿见影的 1](#_Toc16565)

[1.1. 建立索引 在 where 及 order by 涉及的列上建立索引。 join 裂伤 1](#_Toc28166)

[1.2. 全文索引 模糊查询like  %% 操作在所在字段要设置全文索引 1](#_Toc12871)

[1.3. 函数索引 2](#_Toc13337)

[1.4. 物化视图 （视图索引） 2](#_Toc26959)

[1.5. 7强制查询使用索引： 2](#_Toc26204)

[1.6. 复合索引 2](#_Toc29467)

[1.7. count(\*) 的问题 使用触发器构建表行数索引即可 3](#_Toc29530)

[1.8. 使用触发器+索引表系列构建业务特定索引 大力提升性能 3](#_Toc10394)

[1.9. 局部索引 3](#_Toc10899)

[2. 效果实现繁琐或者麻烦的 3](#_Toc13422)

[2.1. 适当降低隔离级别 3](#_Toc18618)

[2.2. 分区 3](#_Toc4784)

[2.3. 选择非事务性存储引擎 3](#_Toc2797)

[2.4. 拆分sql语句 4](#_Toc8137)

[2.5. 使用存储过程 减少数据往返带来的性能消耗 4](#_Toc10414)

[2.6. 在通用操作方面，可以使用aop模式 触发器 4](#_Toc14083)

[2.7. 根据场景选择长事务与短事务 数据库大多都是短事务（几秒内） 。性能影响非常大 4](#_Toc21158)

[2.8. 适当建立多个聚集索引 4](#_Toc9927)

[3. 其他的细则 5](#_Toc1144)

[3.1. Not in 排除大数量时候，也是可以用到索引 5](#_Toc12191)

[3.2. ！= 《》操作符问题 5](#_Toc10794)

[3.3. Select \* 大多数情况下不会影响性能 除非字段超多超大 5](#_Toc11382)

# 改动较小且立竿见影的

## 建立索引 在 where 及 order by 涉及的列上建立索引。 join 裂伤

## 全文索引 模糊查询like  %% 操作在所在字段要设置全文索引

全文索引可以使用数据库自带实现，方便开发高效，或者编程语言实现，性能更高，但更繁琐

## 函数索引

1. 使用函数作为条件是需要建立函数索引。如果数据库原生不支持函数索引，可在表格中增加一列存储函数索引结果，使用触发器实现索引同步

或者也可以使用单独表存储函数索引

函数索引模式当然也可以使用编程语言模式实现，索引存储可以放入表内，也可以写入文件模式

解决在sql中使用函数表达式的问题 .注意表达式操作，函数比较 默认可能走不了单列索引，需要建立函数索引才可以走到

select id from t where num/2 = 100

## 物化视图 （视图索引）

1. 一般情况下数据库已经大多支持物化视图了，如不支持或实现的不好，使用触发器或者 编程语言实现物化视图功能

## 7强制查询使用索引：

.如果在 where 子句中使用参数，也会导致全表扫描。因为SQL只有在运行时才会解析局部变量，但优化程序不能将访问计划的选择推迟到运行时；它必须在编译时进行选择。然 而，如果在编译时建立访问计划，变量的值还是未知的，因而无法作为索引选择的输入项。如下面语句将进行全表扫描：

select id from t where num = @num

可以改为强制查询使用索引：

## 复合索引

在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致。

## count(\*) 的问题 使用触发器构建表行数索引即可

select count(\*) from table；这样不带任何条件的count 按道理是可以用到索引，每个数据库几乎都有对表的行数统计信息，是可以直接提取的而不是扫描。。看数据库具体实现

如果count(\*) 影响性能，那么可以直接读取系统表的统计信息，直接提取表行数统计即可。

1. select count(\*) from table； 带有一定条件的count可能会引起表扫描，

此时可以使用触发器构建表行数索引即可

## 使用触发器+索引表系列构建业务特定索引 大力提升性能

## 局部索引

可以只对部分数据做索引。。一些数据库可能不支持，，此时可使用触发器实现局部索引

# 效果实现繁琐或者麻烦的

## 适当降低隔离级别

可能更具业务场景，需要在触发器或者 编程语言方面补偿此方面的影响

## 分区

## 选择非事务性存储引擎

## 拆分sql语句

有时常见场景拆分sql语句到多条，方便分块优化与索引，有时需要合并多个到一条，根据具体场景而定

## 使用存储过程 减少数据往返带来的性能消耗

某些数据库也可以使用编程语言实现内置存储过程，如果不支持内置，可以外置模式

## 在通用操作方面，可以使用aop模式 触发器

也可以使用编程语言实现触发器 aop 模式

## 根据场景选择长事务与短事务 数据库大多都是短事务（几秒内） 。性能影响非常大

业务场景允许情况下，可以实行非数据库型长事务（几小时甚至几天几个月）可以拉长事务处理周期，压平性能 可 使用存储过程，编程语言实现 一般性能更优

短事务也可以外置编程语言来实现，更高性能。。数据库存储引擎带不带事务功能，对性能影响非常大

使用编程语言实现事务功能，带来更高性能，但更代码繁琐些，数据库内置事务功能消耗较大相对来说

## 适当建立多个聚集索引

1. 很多数据库一个实例一个表内之允许建立一个，但可以部署多个实例多个表，建立多个聚集按照不同维度

可以使用触发器构建索引维护机制

# 其他的细则

## Not in 排除大数量时候，也是可以用到索引

## ！= 《》操作符问题

## Select \* 大多数情况下不会影响性能 除非字段超多超大

如有大字段，可以拆分存放，大表查询可以join

字段过多，也可拆分存放

或者使用物化视图也可解决问题