文章
Michael Lei · 八月 7 阅读大约需 2 分钟

最佳实践之改善日期范围查询的SQL性能

根据日期范围查询的SQL性能让你失望？ 我有一个比较特别的技巧，可能会帮助你解决这个问题!
(SQL开发人员讨厌这个!)*

如果你有一个类，在添加数据时记录时间戳，那么这些数据将与你的IDKEY值保持顺序--也就是说，当且仅当ID1<ID2时，表内所有ID和时间戳值的TimeStamp1<TimeStamp2--那么你可以利用这一知识来提高对时间戳范围的查询性能。 考虑一下下面这个表:

```
Class User.TSOrder extends %Persistent
{

Property TS as %TimeStamp;

Property Data as %String (MAXLEN=100, MINLEN=200);

Index TSIdx on TS;

Index Extent [type=bitmap, extent];

}
```

用过去30天内的30,000,000条随机行来填充，每天将得到1,000,000条行。
现在，如果我们想查询某一天的信息，你可以这样写：

```
SELECT ID, TS, Data
FROM TSOrder
WHERE
     TS >= '2016-07-01 00:00:00.00000' AND
     TS <= '2016-07-01 23:59:59.999999'
```

可以肯定的是，这是一个合理的查询。 然而，在我的系统上，这需要2,172,792个Global引用和7.2秒。
但是，知道ID和时间戳的顺序是一样的，我们就可以用时间戳来获得ID范围。 考虑一下下面的查询:

```
SELECT ID, TS, Data
FROM TSOrder
WHERE
     ID >= (SELECT TOP 1 ID FROM TSOrder WHERE TS >='2016-07-01 00:00:00.00000' ORDER
 BY TS ASC) AND
     ID <= (SELECT TOP 1 ID FROM TSOrder WHERE TS <='2016-07-01 23:59:59.999999' ORDE
R BY TS DESC)
```

新的查询在5.1秒内完成，只需要999,985个Global引用**!

这种技术可以更实际地应用于有更多索引字段的表和有多个WHERE子句的查询。
从子查询中产生的ID范围可以被放到位图格式中，当你得到一个多索引的解决方案时，会产生惊人的速度。
Ens.MessageHeader表是一个很好的例子，你可以把这个技巧用于实际工作中。

我们把话说清楚--这只是一个***性能提升****的例子*。
如果你在同
一个表中的WHERE子
句中有许多条件（而且它们是有索引的），那么这个技术可以给你带来**更大的提升**，在你的查询中试试吧!

* SQL开发人员并不真的讨厌这个，但如果互联网教会了我们什么，那就是抓眼球的广告语会带来更多的流量。

** 当测试返回这么多行的查询时，SMP无法处理，大部分的时间都花在了显示数据上。
正确的测试方法是使用嵌入式或动态SQL，运行结果，但不输出它们的时间，并使用SQL Shell进行Global计数。
你也可以用SQL Stats来做这个。

#Code Snippet #SQL #InterSystems IRIS for Health #全球响应中心 (WRC)

源URL:
https://cn.community.intersystems.com/post/%E6%9C%80%E4%BD%B3%E5%AE%9E%E8%B7%B5%E4%B9%8B%E6%94%B9%E5%96%84%E6%97%A5%E6%9C%9F%E8%8C%83%E5%9B%B4%E6%9F%A5%E8%AF%A2%E7%9A%84sql%E6%80%A7%E8%83%BD