Article
Kyle Baxter · Jul 19, 2016 2m read

Improve SQL Performance for Date Range Queries

Date range queries going too slow for you? SQL Performance got you down? I have one weird trick that might just help you out! (SQL Developers hate this!)*

If you have a class that records timestamps when the data is added, then that data will be in sequence with your IDKEY values - *that is, $TimeStamp_1 < TimeStamp_2$ if and only if $ID_1 < ID_2$ for all IDs and TimeStamp values in table* - then you can use this knowledge to increase performance for queries against TimeStamp ranges. Consider the following table:

```
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];

}
```

Populating this with 30,000,000 random rows with dates over the last 30 days, will give you 1,000,000 rows per day. Now if we want to query the information for a given day you might write the following

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

A reasonable query, to be sure. On my system, however, this took 2,172,792 global references and 7.2 seconds. But, knowing that the IDs and TimeStamps are in the same order, we can use the TimeStamps to get an ID range. Consider the following query:

```
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)
```

The new query completes in 5.1 seconds and takes only 999,985 global references**!

This technique can be applied more pragmatically to tables with more indexed fields and queries that have multiple WHERE clauses. The ID range generated from the subqueries can be put into bitmap format, generating blazing speed when you get a multi-index solution. The Ens.MessageHeader table is a great example where you can put this trick to work.

Let's be clear - this is an *EXAMPLE* of a win. If you have many conditions in the WHERE clause in the same table (and they are indexed, duh!), then this technique can give you **BIGGER** wins! Try it out on your queries!

* SQL Developers don't really hate this, but if the internet has taught us anything is that catchy blurbs get more traffic.

** When testing queries that return so many rows, the SMP cannot handle it, and most of the time is taken in displaying the data. The proper way to test is with embedded or dynamic SQL, running through the results, but not outputting them for time, and using the SQL Shell for your global counts. You could also use SQL Stats for that.

#Best Practices #Code Snippet #SQL #Worldwide Response Center (WRC)

Source URL: https://community.intersystems.com/post/improve-sql-performance-date-range-queries