ComponentOne LiveLinq

Boost LINQ performance and improve data-binding options

ComponentOne LiveLinq promises to boost the performance of Microsoft's Language-Integrated Query (LINQ) operations by means of indexing LINQ queries. This doesn't mean it adds indexes to your underlying database for LINQ to SQL queries. Instead, LiveLinq is persistence-ignorant—meaning that the indexes it creates are on your data once it's been pulled into CLR memory. This, in turn, translates into increased performance for data-heavy applications, with only a minor amount of effort or coding required. In fact, at the simplest level, adding indexed query functionality is as simple as adding simple extension methods to your object collections, as Web Figure 1 (www.sqlmag.com, InstantDoc ID 103603) shows.

LiveLinq also provides functionality called Live Views. Live Views permit the results of a LINQ query to remain actively attached to the in-memory data source or collection that was queried. This functionality is useful for data-binding scenarios within rich applications such as Silverlight or Windows Presentation Foundation (WPF) applications. However, any object that you want to participate in a Live View needs to derive from an IndexableObject, so you can code your property setters to include calls to the IndexableObject's OnPropertyChanging() and OnPropertyChanged() events. And while having to derive from a different base class is a downer, seeing a Live View in action as a data binding source is very impressive.

Live Views, coupled with indexing capabilities, opens up some intriguing possibilities when it comes to scaling applications. For example, if you had an e-commerce application with a large product catalog, you could let site users just thump the database over and over again as they browsed the site. But with LiveLinq, you can create a middle tier that pulls in large swaths of data, indexes that data, then uses Live View operations to keep data updated as needed. You could then query this middle-tier object as an in-memory database to reduce load on your database and increase scalability.

The LiveLinq installer deploys a large collection of sample applications that work well with the accompanying documentation to bring developers quickly up to speed on options and operations. My only serious criticism of LiveLinq is the omission of examples showcasing how to work with data pulled out of LINQ to SQL and Entity Framework solutions. Instead, the samples favor examples of pulling data out of DataSets (which I consider the spawn of the devil). It also has good examples of pulling data from XML or working with collections of objects.

In putting LiveLinq through the paces with a collection of 80,000 objects that I pulled out of my database, I noticed that the performance benefits stemming from the in-memory indexes placed upon DateTime properties on my objects didn't return as large of a performance boost as I saw on other data types. Most likely, this was due to some slight differences in the DateTime precision of my objects versus that of my search criteria, and I would expect that a bit more work on my part would remedy this problem, as indexes in any solution are susceptible to mismatches like this.. I found coding with LiveLinq to be intuitive and fully capable of delivering on the performance benefits advertised.

I heartily recommend that anyone looking to improve performance, scalability, or leverage rich data-binding capabilities pull down a trial copy of LiveLinq and give it a whirl. SQL

InstantDoc ID 103603

Michael K. Campbell

(mike@overachiever.net) is a contributing editor for *SQL Server Magazine* and a consultant with years of SQL Server DBA and developer experience. He enjoys consulting, development, and creating free videos for www.sqlservervideos.com.

COMPONENTONE LIVELINQ

Pros: Fully persistence-ignorant indexing capabilities that can deliver massive performance benefits for LINQ operations; Live View functionality enables compelling data-binding capabilities for Microsoft Silverlight, WPF, and other rich Microsoft .NET applications; can be leveraged to take significant load off of databases to increase application scalability

Cons: Although the sample applications that ship with LiveLinq are excellent, it's missing sample projects centered around LINQ to SQL and the Entity Framework; Live View functionality requires objects to derive from an IndexableObject, which impinges on persistence ignorance and might conflict with other developer requirements or needs

Rating: ★★★★☆

Recommendation: LiveLinq has some limitations and won't be immediately beneficial to every kind of application or solution. However, it provides some compelling capabilities, performance benefits, and scalability options that shouldn't be ignored. Organizations looking to improve LINQ performance, bolster scalability, or increase data-binding capabilities for rich applications should definitely give LiveLinq a test drive.