

## Re: Indexes and primary keys, from Delaney

**Source:**

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-03/1100.html>

---

**From:** Ray Higdon (*sqlhigdon\_at\_nospam.yahoo.com*)

**Date:** 03/04/04

Date: Thu, 4 Mar 2004 05:36:19 -0500

The two red alerts I see here is having such a wide clustered key that would bloat all nonclustered indexes and the likelihood of page splitting. As all nonclustered indexes contain the clustered key (unless non-unique) in David's example the nonclustered index would look like this:

SSN,ACCOUNT,DATE, SEQUENCE NUMBER,SSN,ACCOUNT

Assuming the likelihood of having more than one nonclustered index this may be undesirable. If you have a clustered index on SSN,ACCOUNT,DATE, SEQUENCE NUMBER, you will need to watch your page splitting and perhaps play with fill factors and the timing of reindexing.

One thing I see from your post is this

"I have a database with a big transaction table"

I assume this to mean this is a heavily inserted table. I would not want the table to have to evaluate the physical storage of a wide composite clustered key and would in fact use a compact surrogate key as my clustered index to create your insert HOT SPOTS for faster inserts and much more compact nonclustered indexes.

HTH

--

Ray Higdon MCSE, MCDBA, CCNA

---

"DW" <None> wrote in message news:Xns94A1E1BFB6A19DWalker@207.46.248.16...

> I want to ask a semi-general question, which I think I can do without  
> giving my whole database definition.

>

> Books and articles about table design say things like "Columns that are  
> the primary keys or that are unique are most likely to be joined and  
> frequently queried... When no naturally efficient compact key exists,  
> it's often useful to manufacture a surrogate key using the identity  
> property on an int column". (Inside MS SQL Server 2000 by Kalen Delaney  
> and Ron Soukup, as many of you might recognize, page 272.) Then it says  
> to use this surrogate key for joining and data retrieval. I see similar  
> advice frequently.

>

microsoft.public.sqlserver.programming: Re: Indexes and primary keys, from Delaney

> I understand this..... but I have a database with a big transaction  
> table. The people who supply the daily updates claim that the "key"  
> consists of the SSN, account number, transaction date, and a sequence  
> number. They insist that all four make up the key, although they also  
> claim that the sequence number is unique and never reused. Okay.  
>  
> I don't want a wide primary key, because I'll never query on this -- the  
> sequence number has no meaning and doesn't appear in any other table.  
>  
> I generally query and/or group on SSN, often on the combination of SSN  
> and account number (which identifies a unique account), and often query  
> on transaction date ranges.  
>  
> I originally had all four fields as a composite primary key. I think  
> this is a bad idea; the index is too big. The sequence number could be  
> the primary key, but there's really no need for it be indexed. If I  
> created a surrogate key just to have a primary key, I would never query  
> or join using it.  
>  
> I fond myself querying on the SSN/account number combination, or joining  
> on SSN and Account Number in the account table, for example, being equal  
> to the SSN and Account Number in the transactions table. (SSN/Account  
> Number is the primary key in the account table; that makes sense and  
> seems normal to me.)  
>  
> I'm having trouble figuring out how this good advice applies to my  
> Transactions table. Reams of good advice can't be all wrong.  
>  
> Even without you guys knowing all the things I do with this data, is it  
> reasonable cases like this to have a primary key (the transaction  
> sequence number) that's not indexed? I don't really care if the  
> sequence number is unique or not, but I could make a Unique constraint.  
> (I always make a primary key if only so I can open my SQL table from MS  
> Access, as a project, if I want to.)  
>  
> Where the book says that you can't drop the index created as a result of  
> a primary key, but must instead drop the constraint, is kind of  
> confusing -- dropping the constraint drops the index? I might want to  
> drop the index and leave the Unique constraint.  
>  
> Is a unique constraint as efficient (or more) than having an unnecessary  
> index?  
>  
> Thanks.  
>  
> David Walker