



RUNAS RADIO



<http://www.runasradio.com>



Richard
Campbell

RunAs Radio is a weekly Internet Audio Talk Show for IT Professionals working with Microsoft products. The full range of IT topics is covered from a Microsoft-centric viewpoint.



Greg
Hughes

Text Transcript of Show #076
(Transcription services provided by [PWOP Productions](#))



Kim Tripp Indexes Everything!
September 24, 2008



[Music]

Brandon Wenn: From runasradio.com, you're listening to RunAs Radio, the Internet audio talk show for IT professionals with Richard Campbell and Greg Hughes. This is Brandon Wenn, announcing show #76, with guest Kimberly Tripp, recorded Wednesday, September 24, 2008. RunAs Radio is produced each week by PWOP Productions, providing professional media and podcasting services online at pwop.com.

Richard Campbell: You're listening to RunAs Radio, I'm your host Richard Campbell with me, as always, my friend Greg Hughes.

Greg Hughes: Hey Richard, how are you today?

Kim Tripp: What about me?

Richard Campbell: Oh, Kim you're my friend too but I was trying to do our formal intro.

Kim Tripp: I'll shut, up, I'll shut up. Go ahead do your intro, I'm sorry.

Greg Hughes: Who is that?

Richard Campbell: Who was that?

Greg Hughes: RunAs Radio.

Richard Campbell: So, how are you sir?

Greg Hughes: I'm doing good. I'm doing just fine. How about yourself? You're back from more travels again...

Richard Campbell: Oh, yes.

Greg Hughes: And I'm about to take off on some...

Richard Campbell: I'm back from Nepal, unbelievable experience. I could do a whole show just on that but I don't know that anyone actually cares. It's well worth my time and I'll go back someday but not this week. This week I got to work.

Greg Hughes: That's terrific. So, just so everybody knows Richard goes to Nepal and this weekend I am going to Philly. Nothing wrong with Philly.

Richard Campbell: Almost the same.

Greg Hughes: Nothing wrong with Philly but you know it is a little different, but that's great.

Richard Campbell: Hey, I'm really looking forward to TechEd Barcelona. We've got a great line-up there.

Greg Hughes: Yeah. That's going to be a lot of fun. We'll both be in Barcelona. I think Kim is going to be in Barcelona too, right Kim?

Kim Tripp: I will be there and Paul my husband, he'll be there as well, so a bunch of people, it will be quite fun.

Richard Campbell: So, if anybody hasn't guessed yet, our guest this week, the fabulous, the amazing and my very, very dear friend Kim Tripp.

Greg Hughes: Hello, Kim.

Kim Tripp: Hello.

Richard Campbell: And I managed to pry you away from Paul for a change, so I don't have to...

Kim Tripp: Hey, he's teaching today. This is a rare event where like...

Richard Campbell: Yeah, no kidding.

Kim Tripp: I'm in the house and it's empty and he's working. We were saying this morning that we've literally just spent about six or eight weeks, I mean I have to look at the calendar really but six or eight weeks straight where we actually didn't even separate in any way, shape or form like 24 hours a day for eight straight weeks. So, it's kind of funny, it was almost weird when he was going to work this morning.

Richard Campbell: Don't leave me!

Kim Tripp: Yeah, I don't know how to be kind of around here when you're not around.

Richard Campbell: Well, you two have been inseparable for years now.

Kim Tripp: Yeah. I guess a couple of years.

Richard Campbell: Yeah.

Kim Tripp: It's actually going really well. I really like the whole working together, "playing" together as well but the work side is, it makes things so much less stressful to have somebody else on stage. I really didn't do that many co-presenting presentations over the last few years. It's great because I forget things, he forgets things, I remind him or sometimes I just have a different spin on something or he does and it's really nice because I



think we actually can convey more information and cover more aspects of the information because we have different perspectives and yet we don't have to work quite as hard because we're not both standing up by ourselves for eight straight hours.

Greg Hughes: Sure, plus it's just so much fun to listen and watch the two of you interact with each other.

Kim Tripp: Yeah. Well, I hope that's going to be the case. I'm actually a little bit nervous of our possible spinoff here that we talked about now for like over a year.

Richard Campbell: Yes.

Greg Hughes: That's right.

Kim Tripp: Yeah. I think it will be really fun. I mean it will be one of those things where we'll have to figure out what we want to talk about and then maybe not tell each other what we're going to say.

Richard Campbell: Of course we're talking about KPI which is going to be Kim and Paul show on SQL server when it actually launches, which is imminent I know, soon.

Kim Tripp: It is. We're actually working on the website...

Richard Campbell: Nice.

Kim Tripp: And coming up with a few of our themes where our name, Kimberly, Paul and then I is all over the map. It's investigate, introduce, insanity depending on the show. So, we're going to kind of have a different theme almost for every show. I mean you guys kind of go all over different technologies. Obviously, we're going to focus on just SQL but we'll focus on different aspects of SQL and we'll have interviews that will be another I. It will be quite fun. I mean we started to get all the equipment put together and we had to rearrange part of the house to make sure we could have a place to record that would be really good for recording.

Greg Hughes: Right.

Kim Tripp: It's been a long process but we're getting really, really close.

Richard Campbell: It's exciting.

Greg Hughes: I'll be really excited to see what you come up, it will be great.

Kim Tripp: Yeah, I hope so, I hope so. At a minimum it will be fun and hopefully informational or informative.

Richard Campbell: So today's "I" is indexing.

Kim Tripp: Oh yeah, indexing.

Richard Campbell: There you go.

Greg Hughes: iPhone, oh wait.

Kim Tripp: There are so many I's.

Richard Campbell: So many I's because SQL 2008 is really, they've done some big things with indexing.

Kim Tripp: You know, 2008 in terms of indexing it has one I think really interesting and kind of big new feature and that's filtered indexing. They didn't really change any of the internal structures for the most part. You have the same kind of indexing strategies that we had in 2000, 2005 which is to create a decent clustered index so that your non-clustered indexes are not unnecessarily wide. I even had a few questions, I taught an indexing class in Edenborough about two weeks ago and it was a nice small class, that was actually really quite fun and the fun part about being nice and small is, we actually went into more depth and more areas than I've gone into in some of my indexing classes because there's just so many more people that you end up with a lot of questions but...

Richard Campbell: Right.

Kim Tripp: Less people you get I think a little bit more depth. So, anyway, I had interesting questions and sometimes people are almost shocked by the fact that a clustered index has to be unique and the end results of a clustered index wasn't unique is that every non-clustered index is unique and it was kind of this realization I think for a few folks where they kind of went, "Wow, I never even thought about that." The selectivity of a combination of columns in a non-clustered cluster clustering key, of course if your clustering key is unique is one.

Richard Campbell: Right.

Kim Tripp: And it kind of makes for some interesting, like I said, realizations about indexes and what they look like and that's what the class was a lot about was index internals and good indexing strategies for better performance. It was quite fun, it is quite fun and talking about internals for a couple of days seems really geeky but I had many people that told me, "The light bulb is on now. I get it. I see why it's so important to make a good decision on a



clustered index and then the impact of that in non-clustered." I don't know if you want me to just go on.

Richard Campbell: But I think you're hitting on a very interesting -- this is really the line of what is a DBA do these days because account creation is largely handled, you know, SQL Server does so much of its own care and feeding. Here's the one area I think more than anything where really takes a person looking at the way the app behaves against the database to say, "What are the right indexes?"

Kim Tripp: Yeah. It's so funny that you say that because one of the things I think I've been saying a lot more in even just the last few months is that good indexing and having the right indexes for performance purposes is a combination of both an art and a science.

Richard Campbell: Right.

Kim Tripp: There's definitely a science to it. I can look at a query, I can tell you what kind of indexes exactly would help that query. I mean the answer to what's the best index for a query is always one answer, a non-clustered, covering, tweakable index.

Richard Campbell: Right.

Kim Tripp: Period.

Richard Campbell: That's the fastest, period.

Kim Tripp: What's that?

Richard Campbell: That is the fastest.

Kim Tripp: Yeah, exactly. I mean if you can cover it, right? That means that you have the smallest absolute structure to use to find your data, if you can make it seekable like the phone book, you only have to go to the section that's of interest to you or the name, for example, in the phone book that is of interest to you, right? So, you get right to the section, all of your data is right there and you don't have to go into the form, let's say tables, to look up the data. You skip all that extra work so if you can have it done...

Greg Hughes: All that overhead.

Kim Tripp: Yeah, all the overhead, absolutely. So, if you can have this non-clustered, covering, seekable index then you can significantly improve your performance but you can't cover everything is the point that is the hard part because, I'm trying to think of the best word, but the desire from a tuning perspective always seems to be let me look at query, tune it, let me look at a different query, tune

it, let me look at a different query, tune it, and the end result is you end up with too many indexes. So, the art part of it is trying to determine if you could let's say add a column to an index which you can't really directly add a column to an index but if you could create an index that is maybe a little bit wider than an existing index and then drop that slightly narrower index, you might actually end up with an index that has a lot more use and a lot more benefit and then that's the other problem that I always seem to have to overcome is the -- I don't know Carl or you would probably say it's from the collective unconscious or something but this desire to create wider indexes or I should say the, what's the word I'm looking for, but this unbelievable, I don't know, repulsion at wider indexes.

Richard Campbell: Yeah.

Kim Tripp: Everybody seems to have it. Nobody wants to have wider indexes and I always find that I'm spending time kind of trying to show or prove the point that having a whole bunch of narrow indexes that aren't being used is way worse than having a couple of wide indexes that have the tremendous numbers of users.

Richard Campbell: Because each one of these indexes has impact on writing performance.

Kim Tripp: Yeah, absolutely and the more indexes in and of themselves even if they're narrower can often have a more profound impact on inserts, updates and deletes than having just let's say one or two wider index.

Richard Campbell: Right.

Greg Hughes: Right.

Kim Tripp: Even though it might be more dissipate, the potential for split if you only have let's say three wider indexes is three.

Richard Campbell: Right.

Kim Tripp: But if you have 14 really narrow indexes you have the potential for 14 splits.

Greg Hughes: Yeah, it's an interesting and sometimes counter-intuitive balancing act, right?

Kim Tripp: Yeah, absolutely and that's the art part of it. That's what you constantly end up with, the struggle between what seems logical, "Oh I have a where clause. I want to put an index on this one column," yet if it's not selective enough, then that index is not going to get used and that index is pretty much nothing but wasted overhead and in fact it might not get used at all." So, as far as new features, 2005



introduced DMVs, the Dynamic Management Views, that will allow you to see if an index does or doesn't get used or if it does get used, how it's getting used and you can even start to see some of the DMVs if it's being used for seeking and for look-ups or if for example it's only being used by modification statement.

Richard Campbell: Right.

Kim Tripp: You can even start to get to feel, "Wait a minute. This index, it's being used but for nothing but updates," and then you can start to look at these DMVs and decide maybe that you should get rid of some indexes but the problem with them -- I don't know how much you guys have had sessions about DMVs but the problem with DMVs is that they're not persistent since the beginning of time.

Richard Campbell: Right.

Kim Tripp: They just kind of give you a right now what does memory look like and what's going on? The usage DMVs are quite nice because they do kind of keep information around but if you shut down SQL server or deep patch a database, you tend to lose a lot of that information in many of the DMVs. So, the idea in 2005 was to kind of persist that data so you can do some trend analysis. In 2008, there's a new feature called performance data collection that will collect certain information about queries so it's not tracking directly index usage information but it's tracking queries so that you can start to do some trend analysis over let's say 14 days, that's the default...

Greg Hughes: Sure.

Kim Tripp: Trend information right? So 14 days and then you can start to see what queries are the most common and which queries are the most expensive and you can even change the data that's collected and create your own custom data collection and then go back and do some analysis on that. So, all of those things together will hopefully get someone closer to determining if let's say the indexes that they have shouldn't be there and once they start whittling that down, they will probably find out that they don't end up with too many indexes and then they'll probably get a lot of missing index recommendations. So, that's another DMV and I always feel like I need to pause because I mean it's really cool. It's a great capability that they added in 2005 and I think they're getting a lot better with 2008. A lot of people are getting cranky with me in saying, "Why don't they just create the indexes for me or drop the indexes for me?" and that's a whole another discussion. It's really hard to do, I mean really, really hard to do.

Richard Campbell: Didn't the Profiler have a wizard for this at one point?

Kim Tripp: Yeah. Well, actually, no, that's a great point. Profiler or tracing allows you to capture a workload...

Richard Campbell: Right.

Kim Tripp: Which is very much like the idea of performance data collection, you know, capturing the information about queries for example but Profiler can trap a whole bunch of other stuff. It can, if you want I wouldn't really recommend it, but you could just let's say trace all the locks that are going on, you'll really kind of hurt your performance by doing that because it's so expensive just to take that in-memory thing but you could do all sorts of stuff. Once you have a workload, one of the things you can use is the tool that most people associate with Profiler which is I think why you said that but it used to be called index tuning wizard.

Richard Campbell: Right.

Kim Tripp: So, it was a wizard-based tool and the index tuning wizard which was the 2000 tool would allow you to go and look solely at index recommendations. In 2005, they changed the name. It's called the database engine tuning adviser or DTA for short. Everybody has called it DTA and DTA does not just indexes but also partitioning. So, they changed the name but it really is kind of the ITW with quite a few enhancements to be honest, but it's kind of a progression of it. So, it also has I'd say a pretty intuitive UI that's kind of wizardry in that it has recommendations and then you kind of just go. It's good. It's actually in many cases I would say much better than relying solely on the DMV because the DTA will look more comprehensively at a workload and it won't just say this index hasn't been used. How do I want to say this? If you look at the DMV and the DMV had been wiped out a few days ago because somebody bounced the server and no one had persisted the DMV, right?

Richard Campbell: Yeah.

Kim Tripp: Then you'll find out, "Oh this index hasn't been used in the last three days," so then you might kind of make a bad decision to drop it when maybe the index is actually really critical for let's say month end processing.

Richard Campbell: Yeah, that's what I was thinking is I think it's the same problem with the Profiler too which is if you've only taken a one-hour work load...

Kim Tripp: Yeah, exactly.



Richard Campbell: Your month end stuff is not going to show up unless you ran that workload during the month end.

Kim Tripp: Yeah, totally and that's why I really like -- I like DTA's and I like Profiler but it's kind of like all of the tools are fantastic especially if you know kind of exactly what you should do with them.

Richard Campbell: Yeah.

Kim Tripp: Exactly how to use the tool because each tool has pros and cons, weaknesses and even faults in some cases because it can only do, you know, it's the old garbage in, garbage out.

Richard Campbell: Yeah and you hinted at this whole thing of the Profiler exerting non-trivial load on the server to make its record in the first place. So, the reason you're tuning the month end workload is it's already too slow and now you're going to turn Profiler on make it even slower.

Kim Tripp: Yeah, I mean this is a great point kind of in and of itself is profiling for a better performance usually leads to short-term worse performance.

Richard Campbell: Bad performance.

Kim Tripp: The good news is that there are all sorts of tips and tricks out there on how to do what's called a server side trace which is kind of lighter weight than using the Profiler UI to actually profile. You can use the UI to kind of generate the details of a trace which is all code based and then you can create that trace on the server so that the server runs the trace and logs it directly to a file on the server, nothing goes over the network, nothing goes to the UI and you can even then programmatically like start and stop the trace through jobs for example or even you could have an alert like arrayserror message that raises the user defined error that then triggers an alert which kicks off a trace. I mean you can do some really cool stuff when you use kind of decode aspects of server side tracing as opposed to just using Profiler itself. I actually did a webcast. I don't even -- I think everything's still downloadable because it's still fairly clickable, but I did a MSDN webcast series a few years ago now and part 9 was on profiling and server side tracing. I've got a link to that on my homepage as well and I think all that still is just easily downloadable, but yeah, you're always going to pay a penalty.

Greg Hughes: What are some of the other indexing questions you get? I mean you do a lot of webcasts and you certainly do a lot of presentations. People come up to you afterwards. I assume you probably get a lot of clarifying questions. Are there

some that are pretty common, maybe good things just to sort of talk about here?

Kim Tripp: Yeah. Tons, I mean tons. One is always related to the structures and why does a clustered index needs is unique because SQL server doesn't tell you that you have to make the clustered index unique but yet internally if your clustering key is not unique, SQL server will "uniquefy" that clustered index.

Greg Hughes: Okay.

Kim Tripp: So what ends up happening is your insert performance, your update performance can actually be slowed by having to check and then "unique-fy" the row.

Greg Hughes: Right.

Kim Tripp: So, a lot of people ask me like, "Why does the clustered index need to be unique?" and it actually has to do with non-clustered index. So if you think of -- I'm trying to think of a really simple example that would be perfect like imagine a table, that is employee data, I use this one in a lot of presentations but imagine a table that is employee data and this data happens to be ordered by let's say last name, okay, which is not a clustered index I would recommend but somebody chose this because they think range queries are going to be better with a clustered index by last name. So, they decide they're going to cluster it by last name. Well, then, they decide they also need to look up employees by social security number so they create a non-clustered index on Social Security Number. In SQL server as of 7.0 when they re-architected the storage engine, SQL server 7.0 and higher uses if a table is clustered the clustering key to do the row look-ups. So, again, imagine this non-clustered index on Social Security Number and you run a query, select star from table where Social Security Number = 123456789. You go into this index and sitting there with Social Security Number blah, blah, blah is the last name Smith. Well, I don't know if that's going to help you all that much, right? You might have a couple of hundred Smiths.

Greg Hughes: Right.

Kim Tripp: So, then you have to go back let's say and scan through all the Smiths to find the Smith that has the Social Security Number of 123456789. So, they don't allow the clustering to be non-unique because of this exact problem. They wouldn't know exactly which row to reference. So, when you create a clustered index on last name instead of like the 200 rows all being Smith, Smith, Smith, each non-unique row that goes in gets uniqueified so they add a 4 byte integer that internally is stored in the variable block of the row as you can't



really see it, it's kind of hidden, I mean you could if you did a CTCP PAGE and actually looked at the raw row format themselves but outside of that it's totally hidden and it's a 4 byte uniquefier that ends up not only getting stored in the rows, so you have extra 4 bytes in the row but then you have the extra 4 bytes in the index so that now 123456789 Smith is not just Smith, it's Smith 47, right?

Greg Hughes: Gotcha.

Kim Tripp: So, then you know exactly which row it is to look up to find the rest of the data. I think that's really interesting and most people would then say to me 4 bytes, you know, Kim, give me a break, is that really a big deal?

Richard Campbell: What's the big deal?

Kim Tripp: Exactly, but you've got 4 bytes extra in the row, 4 bytes extra for every uniquefier that's needed in every non-clustered index, it's kind of like unnecessary overhead that you're not benefitting from and it just doesn't make sense. So, that's one thing.

Richard Campbell: So what if you don't create a clustered index?

Kim Tripp: So, if you don't create a clustered index then you'll end up with a heap and a heap structure is an ordered structure so the order of the rows is irrelevant and rows can move freely, for example, if they get modified but then something interesting happens because imagine having a table, this is actually great discussion. I really like this one because I think this surprises people, imagine having a table where the data doesn't have any order then you have a heap but then put this non-clustered index on Social Security Number on it. Well, if we have to look at the row and the table doesn't have any order, then where's the row, right? So, we have to have this thing called the row ID and the row ID is an 8 byte value. So, that's fine. That's not a big deal, okay? So, now we have a non-clustered index that has this row ID that points us to the physical location of the row but then what if the row itself gets updated and the update makes the row larger such that the row cannot fit on that page that it's currently on because of the heap because SQL server doesn't do splits, they do something called record relocation, so they'll take this row independently and they'll move it to another page owned by this table or they might even have to allocate a new page but they'll move this row to where it would fit and then imagine what would have to happen? They would have to tell the non-clustered indexes where this row moved. So, instead of making these reads what we call, it used to be called kind of a volatile reads meaning that when the rows moved around, the reads would have to change so all the

non-clustered indexes would have to be notified essentially for this row that it had moved and if you have let's say 10 non-clustered, 20 non-clustered, 30 non-clustered which is a lot but if you did, all this record relocation would have to be fixed in all of your non-clustered indexes and kind of having said that, that was 6 bytes, 6 bytes did that. Every time a row moved all of the non-clustered index would have to be touched with the new locations, that's what I wanted to call it before, the new physical read so they had actual physical reads in pre 7.0 days. So, they changed that in 7.0 and higher. They used this what they call forward pointing system to allow a row to move but to not change the non-clustered indexes look-up by essentially leaving the non-clustered indexes pointing to the original location the fixed read where it got inserted and then if it moved, they have an 8-byte forwarding pointer that points to the new location of the row.

Richard Campbell: And every time the row gets updated, that pointer will be updated.

Kim Tripp: Every time the row moved, then the pointer gets updated. You don't end up with a chain. It just goes to the new final location.

Richard Campbell: Right, but now you're paying the penalty on the select for the double seek as opposed to the penalty on the update when all the indexes need updating.

Kim Tripp: Exactly.

Richard Campbell: Interesting tradeoff.

Kim Tripp: Yeah and this is one of my reasons for why seeks are not necessarily the best choice in more and more volatile tables because the more volatile the table, the more of the forwarding pointers you can end up with and the more random your IOs is going to start to become...

Greg Hughes: Right.

Kim Tripp: Because your rows might be doing two physical look-ups in the table structure essentially.

Richard Campbell: Right.

Kim Tripp: And then that always leads to the question, "Well, if a non-clustered just has this one look-up into a heap and a non-clustered on a clustered has to go through a tree and the tree of a clustered index could be let's say three or four levels, doesn't that create more excessive IOs than this look-up to the actual read?" because think about it, if you only have to go to the read and even if it has a 40 pointer, let's say that's two IOs max, one maybe two



IOs, but in the clustered index that has four levels, you'd argue you have four IOs no matter what.

Richard Campbell: Right.

Kim Tripp: Right? So two versus four is a pretty easy, "Wow, two sounds better than four." Ironically and this is the thing that I think people have the hardest time with, ironically, the four is actually better and that -- okay, let me explain. The levels of the tree, the root and the intermediate levels are rarely not in cache.

Richard Campbell: Right.

Kim Tripp: They're almost always in cache. The place where you're most likely to encounter a physical IO is in what we call the leaf level and the leaf level of the clustered, right, a clustered index is always one IO, whereas, in a heap since you have the potential for a forwarding pointer, the potential in a heap especially with forwarding records is that you could have two physical IOs.

Richard Campbell: Right.

Kim Tripp: Because it's not likely to have that whole table in cache. So I mean isn't that ironic that it's like the clustered index has always gotten the bad name because it has multiple levels and it just looks bad, four looks worse than two, but when you actually break it down to the cost of IOs it tends to be in a table especially that it has a lot of forwarding pointers, a more expensive process to do the IOs in the heap than it is in the clustered table. I always find that really interesting.

Richard Campbell: You know, we're walking around the subject here around these clustered indexes which require this uniqueness. I don't even know if I should tip this over, the natural key versus artificial key debate.

Kim Tripp: Oh no, Richard, how much time do we really have?.

Richard Campbell: You don't even want to go there.

Kim Tripp: I can go there with reasonable -- it's funny, I do sound passionate, I do love SQL server...

Richard Campbell: Right.

Kim Tripp: I actually have a hell of a lot of fun with it. I mean I really do.

Richard Campbell: Yeah.

Kim Tripp: Sorry. I mean I really enjoy this product but I'm not one of the ones that would get on to newsgroups or forums and become like psycho-crazy woman on some point and maybe that's one of the reasons why people could agree with me on things that I'm not totally psycho-normalization person or...

Richard Campbell: Right.

Kim Tripp: Or not a totally psycho. I have to have a natural key.

Richard Campbell: Yes.

Kim Tripp: To me, my philosophy has always been know how the products work and if you're lucky enough to let's say only have to work with one product like for me SQL server, the good thing is that you can make your decisions based on knowing your data, knowing how it's going to be used and knowing how the darn product is going to use it, right?

Richard Campbell: Yup.

Kim Tripp: And with those three things you can make so much better decisions than just going, "Oh, I need to believe the normalization."

Richard Campbell: Yeah, there's no one right way.

Kim Tripp: Yeah and that's -- oh, people get so frustrated in me and I know Paul does this, every question starts with the answer it depends.

Richard Campbell: Yes.

Kim Tripp: And I know it's frustrating. It's frustrating but it's a good thing. The SQL server is incredibly flexible and of course that has its bad side too. It's incredibly flexible, you know what you're doing and to make really good decisions based on your data and how it's being used, you can actually set some great performance. If you just want to take the simple route and just apply rules, there are definitely some rules that are better than others but I rarely am a always-abide-by-relational-theory-rule kind of person.

Richard Campbell: Well, so one of those rules being it sounds to me like creating a clustered index is a good idea.

Kim Tripp: There's a rule, there's a rule that I like, creating a clustered index, and not just create the clustered index on the primary key which is the default behavior...

Richard Campbell: Right.



Kim Tripp: And I've never really totally agreed with that but it does kind of make sense because your primary key has to be unique...

Richard Campbell: Yes.

Kim Tripp: So, hey, clustered index has to be unique, your primary key should be non-volatile, to be honest that's another attribute of a clustered index, you don't want it changing because then it would cost me to change it in all the non-clustered.

Richard Campbell: Right.

Kim Tripp: There are things that make sense but then certain primary keys can be really problematic like a natural key. I mean often a natural key is one that takes three or four or five, I've seen 10 columns to actually make the row unique.

Richard Campbell: Right.

Kim Tripp: And instead of creating a surrogate or a part number, for example, they want to go with this natural key. Well, I can name like 10 problems with it. It's going to make the non-clustered indexes wider than they would normally need to be because you're going to waste all this space duplicating all 10 columns in all of your non-clustered, right?

Richard Campbell: Right.

Kim Tripp: It is unique, yes, but now certain aspects might change because there are 10 columns involved. I doubt that all of them are completely static but maybe, I mean let's say they are, right? Now, what about your foreign keys? Your foreign keys are unnecessarily wide but now you've not only widened your base table, well maybe not widened your base table, but you've widened all of your foreign reference tables...

Richard Campbell: Yes.

Kim Tripp: And your indexes are going to be really wide and really wide indexes when you start talking about joining really wide indexes, it's not as ideal, right? I always talk about wide indexes in the context of seeking data and finding data but now joining data is different. I usually don't want to have crazily wide indexes there. So, it is one of those things that world takes time to really bring together but I don't hide them to just to base theory on which they started and believe me, I do recommend knowing relational theory but I don't necessarily apply every rule to the way that I design a table or a database and so the clustered index for me should be something that's unique, narrow, static, and preferably something that's like ever increasing so that I can

create a pattern to my data that ends up having less splits, less fragmentation and often even better performance through the hot spots and that's another big question that I get. I don't know how much more time we have but another big question I get is what about hot spots? Are hot spots good? I'll just sum this up by saying actually because of low level locking, hot spots can be good.

Richard Campbell: Right.

Kim Tripp: And then that's another very counter intuitive concept that people struggle with the first couple of hours and then I beat it out of them.

Richard Campbell: That having a very active part of the table, the whole thing ends up cached.

Kim Tripp: Yeah, absolutely.

Richard Campbell: And low level locking means they don't conflict. We got to hate hot spots in the older versions of SQL server because the pages are unlocking.

Kim Tripp: Richard, you always surprise me with your little past nuggets of information that I just wouldn't have expected you to know all the way back, but you're right, that's absolutely accurate.

Richard Campbell: The only memories I have are painful ones.

Kim Tripp: Yeah, that's true. They're harder to get rid of.

Richard Campbell: Yeah. This is just scar tissue you're talking about.

Kim Tripp: Yeah, totally, totally. Page level locking was a nightmare and it made us all not want to have ever increasing trees because we'd serialize our inserts.

Richard Campbell: Right.

Kim Tripp: And it was horrible.

Richard Campbell: When you hit that moment where the right thing to do is to increase the lock contention level because it actually worked better, that's where you went home and sobbed in your pillow.

Kim Tripp: Yeah.

Richard Campbell: I remember actually setting to serializable and then it worked better.



Kim Tripp: Well, yes. Sadly, that could be because you end up holding locks for a longer period of time so you end up with sometimes less deadlock, sometimes more, it depends.

Richard Campbell: Right. You enforce the order of insert through the stored procedure, things worked better.

Kim Tripp: Yeah, that's true in some cases absolutely.

Richard Campbell: When it actually happened to me, I swear I cried because when it gets every gut reaction you had that this is shouldn't be better, why is this better? I'm sad that it's better.

Kim Tripp: The good old days, right, the good old days. I laugh at that too. SQL Server 6.5, you honestly did have to work a lot harder for better performance and I think a lot of people forget that there have been so many improvements in the product that do make things way easier like memory management, cache management especially that people get hung up on how complex indexes is but at the same time a lot of stuff is made easier and then we need to finesse and figure out the right types of indexes but at least not indexing every single column just because you can and avoiding some of those mistakes. If you can avoid those and slowly work on better strategies, it is a lot more effective.

Richard Campbell: Start with a good clustered index, build your non's carefully, fewer is better, wider isn't automatically a sin.

Kim Tripp: Wow. There you go. We should end there. That's a great summary, and it's perfect, perfect, I totally agree.

Richard Campbell: Oh, it's awesome. Kim, I think we are out of time.

Kim Tripp: I'm not surprised.

Richard Campbell: We would change gears now and go another 40 minutes easy but maybe we'll save that for another show.

Kim Tripp: I'm always surprised when we run out of time.

Richard Campbell: And sometime this fall we're going to get to see the first episodes of KPI.

Greg Hughes: Can't wait for that.

Kim Tripp: Absolutely, absolutely. They are eminent.

Richard Campbell: Awesome. Kimberly Tripp, thank you so much.

Greg Hughes: Thanks, Kim.

Kim Tripp: Thanks, you guys.

Richard Campbell: And we'll talk to you next week on RunAs Radio.