|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | |  | | | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | Fresh new post from LittleKendra.com | | |  | | --- | | [View this email in your browser](http://us12.campaign-archive2.com/?u=f4a5f79a02ee0195e71b8327c&id=4353bd9145&e=61706f20b2) | | |  |  |  | | --- | --- | | |  | | --- | | https://gallery.mailchimp.com/f4a5f79a02ee0195e71b8327c/images/f29c14a2-fcf3-4b20-bcd0-ecc959c5f7f6.png | |  |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | |  | | --- | | **LittleKendra.com**  Me talk query someday. | | | | | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | |  | | --- | |  | | |  |  |  |  |  | | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | [Altering an INT Column to a BIGINT (Dear SQL DBA Episode 11)](http://littlekendra.us12.list-manage1.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=b1739a59bf&e=61706f20b2)  *By Kendra Little on Aug 04, 2016 08:00 am*  You need to change an INT column to a BIGINT in a large table. Learn why this schema change can make your transaction log explode, and how to avoid it.  *This is a “listen-able” 21 minute video. Prefer a podcast instead? Find it at at* [*littlekendra.com/dearsqldba*](http://littlekendra.us12.list-manage1.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=e1b3b8953a&e=61706f20b2)*.*  *Show notes (transcript with links) are below the video.*   |  | | --- | | [Watch the Video](https://www.youtube.com/watch?v=sCoLRRsjM6o) |   **Here’s our question for the week**  Dear SQL DBA,  I have a 120GB table with 2 billion records and the Clustered Primary Key is an INT column. What’s the best way (and fastest) to convert this INT to a BIG INT?  The database is in the full recovery model, and I can take downtime up to a day on the weekend.  With Concern,  Billions and Billions Served  **What happens when you run out of INTs?**  Error msg 8115The integer data type in SQL Server allows numbers up to 2.147 billion rows (with a little change).  If you try to insert a row with a value above that, it fails with error 8115:  Msg 8115, Level 16, State 1, Line 142  Arithmetic overflow error converting IDENTITY to data type int.  At that point, you have to do *something* to insert more rows.  **Altering the data type of the column is a “size of data” operation**  This seems straightforward at first. Let’s change the column to a bigint!  The first thing you notice is that you’re going to have to remove the Clustered Primary Key. You can’t change a data type with that there.  So we’re already in outage territory (even before you think about whether or not you have foreign keys defined against this PK or replication on the table).  Once you get to the point where you can alter the data, you run into a different problem:   * Altering the column uses a huge amount of transaction log space * It’s reaaaalllll slow   This is because changing from INT to BIGINT is a “size-of-data” operation.  SQL Server has to go through and update Every. Single. Row.  And log it.  And SQL Server has to reserve extra space in the log in case it needs to roll back (because that will require additional logging).  If you get this far without thinking about how much log file you’re going to need and you run out, then things get extra slow, because most of that rollback’s going to be done by just one thread.  **You can choose to “reseed” an identity column**  The integer data type can support down to -2.147 billion rows (and some change).  You can reset your identity column to -1 and work your way backwards by negative increments.  You can reset your identity all the way to the lowest possible value and work your way back towards zero.   * **The good news**: it’s fast. * **The bad news**: I have never met a Change Approver who loves this option. It makes them very itchy and uncomfortable.   Reseeding an identity column like duct tape: it’s a temporary solution, and you’re going to have to fix it for real later. And it’s unfamiliar duct tape.  Who looks at a brand of off brand duct tape and wants to use it to repair their roof?  If you do reseed to get things moving right away, the table just gets bigger, and the job of fixing it gets incrementally harder as time moves on.  **Inserting the data into a new table is usually faster than altering the column for large tables, particularly if you can get minimal logging**  This is pretty simple if you can take an outage. During the outage:   * Create a new table using the new data type * Set identity\_insert on for the new table if you have an identity * Insert all the rows with tricks to minimize transaction log use * Create all the indexes, constraints, etc to make it identical * If it’s an identity column, don’t forget to fix that up so new inserts generate a new identity * Use renames or schema transfer to make the new table active   Do test this against a restored backup first so you can make sure you have all the details right and know how much data log and transaction log space you’re going to need. The work with all the indexes can still require substantial log usage.  **How to minimize logging**  SQL Server can do “minimal logging” in the Simple or Bulk Logged recovery model. NOT in the full recovery model. If your database is in the Full recovery model, you’re going to have to spend some time thinking about recovery models and backups.  With Bulk Logged recovery model:   * You are going to have a very large and possibly very slow log backup if you do a lot of bulk changes * You lose point-in-time recovery for any log backup that contains bulk logged changes   With Simple Recovery model:   * Changing to this breaks your transaction log backup chain * You can restart the transaction log backup chain with either a full or differential backup after the change has done (but again, you’re making a lot of changes so that differential might be pretty slow, depending)   If your database is already in the Simple recovery model, then party, it’s easier. Just make sure you don’t try to get minimal logging while a backup is running, that doesn’t work.  The next step is to read the “Understanding Minimal Logged Operations” in the [Data Loading Performance Guide](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=cc78e3d7b9&e=61706f20b2).  If you’re inserting rows into a heap with no nonclustered indexes, typically you can get minimal logging by just using a TABLOCK hint.  If you’re inserting rows into an empty table with a clustered index, you may need additional hints or Trace Flag 610 on your statement.  There’s a handy dandy table in the whitepaper to make it easy.  Typically inserting into a heap with no indexes is the fastest option. If the speed of the change is important, test different inserts.  If you’re using SQL Server 2005, well, first it’s out of support. Second, it doesn’t allow minimal logging for INSERT SELECT operations, you have to do some SELECT FROM OPENROWSET (BULK ) weirdness.  **What if you can’t take the long downtime?**  Sometimes you just can’t take the outage. In that case, you’ve got to proceed with your own wits, and your own code. This is tricky because changes are occurring to the table.  The solution typically looks like this:   * Set up a way to track changes to the table – either triggers that duplicate off modifications or Change Data Capture (Enterprise Edition) * Create the new table with the new data type, set identity\_insert on if needed * Insert data into the new table. This is typically done in small batches, so that you don’t overwhelm the log or impact performance too much. You may use a snapshot from the point at which you started tracking changes. * Start applying changed data to the new table * Make sure you’re cleaning up from the changed data you’re catching and not running out of space * Write scripts to compare data between the old and new tables to make sure you’re really in sync (possibly use a snapshot or a restored backup to compare a still point in time) * Cut over in a quick downtime at some point using renames, schema transfer, etc. If it’s an identity column, don’t forget to fix that up properly.   **Multi-Terabyte example**  I’ve worked with cases of Extra Extra Large databases where schema changes were staged outside of production using SAN snapshots and new databases.  If you’ve got an XXL change, SAN magic can be a big help.  **Aside: SQL Server 2012 added a new Enterprise feature to eliminate one type of size of data operation**  SQL Server 2012 added a feature where adding a new non-nullable column with default values is no longer a size of data operation.  This is an Enterprise Edition feature and it only works for non LOB types (no nvarchar(max), XML, etc). It’s also picky about rowsize.  Remus Rusanu wrote about this feature here in his article [Online non-NULL with values column add in SQL Server 2012](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=7ad09903c8&e=61706f20b2).  This doesn’t help our case here, because our existing Primary Key isn’t the same as a “default value”. But it might help some readers.  **Want another take on this problem?**  Kenneth Fisher wrote a great post about his experience with altering a column in a 750GB table in his article – [Altering a column in a large table: a case study](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=24a6d94c31&e=61706f20b2).  If you’d like to know more about how altering columns works internally, check out Remus Rusanu’s article, [SQL Server table columns under the hood](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=9fd8bbffb5&e=61706f20b2).  [Read in browser »](http://littlekendra.us12.list-manage1.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=988ef0fe20&e=61706f20b2) [share on Twitter](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=c715f97be8&e=61706f20b2)[Like Altering an INT Column to a BIGINT (Dear SQL DBA Episode 11) on Facebook](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=755372ad16&e=61706f20b2)  **Recent Articles on LittleKendra.com:**  [Free Poster: Game of Performance Tuning](http://littlekendra.us12.list-manage1.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=c417205163&e=61706f20b2) [Should I Automate my Windows Updates for SQL Server? (Dear SQL DBA Episode 10)](http://littlekendra.us12.list-manage1.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=5e08ce8515&e=61706f20b2) [Free Poster: Troubleshooting SQL Server After a Migration or Upgrade](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=e5cb38a3b3&e=61706f20b2) [Why an Upgrade can Cause Performance Regressions (Dear SQL DBA Episode 9)](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=6b29a49593&e=61706f20b2) [Free Poster: Cat DBA Says “Not Right Meow”](http://littlekendra.us12.list-manage1.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=af36f917d1&e=61706f20b2) | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | |  | | --- | | [https://cdn-images.mailchimp.com/icons/social-block-v2/color-twitter-48.png](http://littlekendra.us12.list-manage2.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=b0f9640714&e=61706f20b2) | | | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | |  | | --- | | [https://cdn-images.mailchimp.com/icons/social-block-v2/color-instagram-48.png](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=c0de44ffe1&e=61706f20b2) | | | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | |  | | --- | | [https://cdn-images.mailchimp.com/icons/social-block-v2/color-link-48.png](http://littlekendra.us12.list-manage.com/track/click?u=f4a5f79a02ee0195e71b8327c&id=a1453484e2&e=61706f20b2) | | | | | | | |  |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | |  | | --- | | *Copyright © 2016  Kendra Little, All rights reserved.* You're receiving this hand-crafted mail because you signed up at littlekendra.com.  **My mailing address is:**  Kendra Little  340 S Lemon Ave #4363  Walnut, CA 91789  [Add us to your address book](http://littlekendra.us12.list-manage.com/vcard?u=f4a5f79a02ee0195e71b8327c&id=08999f6c74)  Want to change how you receive these emails, or not get them at all? You can [update your preferences](http://littlekendra.us12.list-manage.com/profile?u=f4a5f79a02ee0195e71b8327c&id=08999f6c74&e=61706f20b2) or [unsubscribe from this list](http://littlekendra.us12.list-manage1.com/unsubscribe?u=f4a5f79a02ee0195e71b8327c&id=08999f6c74&e=61706f20b2&c=4353bd9145)  [Email Marketing Powered by MailChimp](http://www.mailchimp.com/monkey-rewards/?utm_source=freemium_newsletter&utm_medium=email&utm_campaign=monkey_rewards&aid=f4a5f79a02ee0195e71b8327c&afl=1) | | | | | |

http://littlekendra.us12.list-manage.com/track/open.php?u=f4a5f79a02ee0195e71b8327c&id=4353bd9145&e=61706f20b2