Intro to blogging. I’ve been putting off blogging for a variety of reasons for a very long time. After promising to publish two times a week for a while I realized that I kept writing articles and then rewriting them because during the process of writing I would learn a lot of things. Often times that would change my premise. I’ve referred to this as the rabbit hole of SQL server. At this point I realized I’m going to be learning forever and things are going to change. The only way to begin a project is to begin it and this is step one.   
  
 My plan here is to begin with the information that I wish I began with and as I gain knowledge refine these articles over time. This is going to be a departure from the current trend of abstracted coding where you write as much code as possible and then try to figure out what you’re doing wrong. I want to provide guard rails so that when you do things you know aren’t the best way you can have a bit of insight into why before it goes too far.

This first article came to me when my kids were sitting around eating skittles. My youngest had a pile of them and was sorting them in to colors and counting them. It for some reason made me think of a heap. A heap might look like



A heap is a table without a clustered index (1). We probably don’t want to store most data like that but it does have its place. (2) If the table is very small, the data is never retrieved or the table is only written to never updated or deleted from then you’d have an ordered heap or something so small no performance concerns would be raised. Regardless if you do retrieve data you have to look at everything to get what you want. A heap would be equivalent to a book with no chapters or page numbers. It might be in the right order but you’d have to read the entire book to be sure.

Further using the book example, if we inserted the book into a SQL database we could have a heap table with a column for page number, a column for chapter name and a column for the text on each page. This would leave us with a data page for the integer page numbers, likely leaving the chapter names on the same page and a page for text as that would likely be a text or other large object field. Depending on size there may be more pages but distinctly there would be at least two. Although in a table, the data being stored as a heap would make for a confusing book potentially coming out different each time. That wouldn’t be acceptable to we need order.

We want order in our book to make it possible to read. So I would add an index. If I want the data to be ordered I use a clustered index (one per table). If I decide I don’t need it ordered (and/or its not unique) but would like it more easily searchable, and I have a large number of distinct values I can use a non-clustered index.

In skittles logic a nonclustered index in the first column might look like this.



Somewhat distinct although not unique. No particular order. The table is still a heap since it has to clustered index. The values would be orange, purple, green, red and yellow. The query optimizer might not find a use for that index and instead opt for a table scan but if it was useful it could speed things up. In the book example

* Yellow is chapter one
* Orange is chapter two
* Red is chapter three
* Green is chapter 4
* And purple is chapter 5

If we wanted chapter one that could be returned but without an order by statement we’d get back a somewhat random order.

To go back to the book example if I was designing a database like a book the clustered index would be page numbers as they are unique (although not required to be unique) and need to be ordered specifically. The chapters would be a nonclustered index as they should be relatively distinct but not as specific as a page number. A skittles clustered index might look like this.



* Yellow is page one
* Orange is page two
* Red is page three
* Green is page four
* Purple is page 5

They are unique, appropriately ordered but still require an order by statement to assure order. We retrieve the book in page order as we would normally expect to read it. If we wanted a section we would have the ability to pull the chapter out using the nonclustered index.

In summary we started with a heap, added a nonclustered index and then changed from a heap table into a clustered table by adding clustered index to order the table on the disk for ease of retrieval.

1. Microsoft. (n.d.). Heaps (Tables without Clustered Indexes). Retrieved February 13, 2016, from https://msdn.microsoft.com/en-us/library/hh213609.aspx
2. Microsoft. (n.d.). Using Nonclustered Indexes. Retrieved February 13, 2016, from https://technet.microsoft.com/en-us/library/aa933130(v=sql.80).aspx