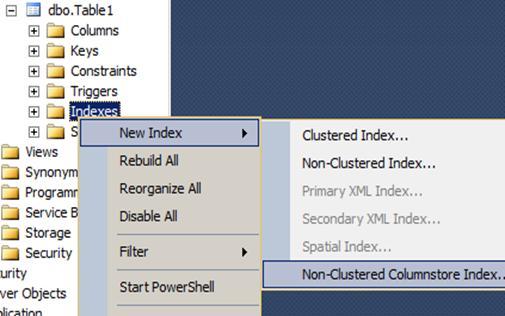
Feature number 1 (Revolution):- Column store indexes

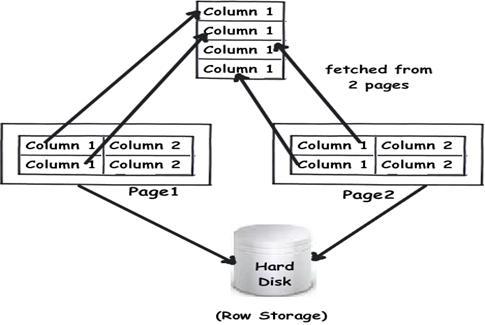
Column store indexes are unexpected and awesome feature. When I read this feature first time I was like, mouth wide open. You can get this feature when you right click on the indexes folder as “Non-Clustered Column store Index” , as shown in the below figure.



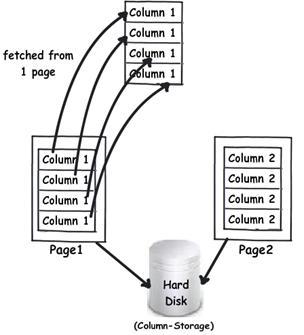
So let’s quickly understand what exactly it does. Now Relational database store data “row wise”. These rows are further stored in 8 KB page size.

For instance you can see in the below figure we have table with two columns “Column1” and “Column2”. You can see how the data is stored in two pages i.e. “page1” and “page2”. “Page1” has two rows and “page2” also has two rows. Now if you want to fetch only “column1”, you have to pull records from two pages i.e. “Page1” and “Page2”, see below for the visuals.

As we have to fetch data from two pages its bit performance intensive.



If somehow we can store data column wise we can avoid fetching data from multiple pages. That’s what column store indexes do. When you create a column store index it stores same column data in the same page. You can see from the below visuals, we now need to fetch “column1” data only from one page rather than querying multiple pages.



Feature number 2 (Evolution):- Sequence objects

This feature is good to have and I personally feel it just mimics Oracle’s sequence objects.  Looks like it’s just a good to have feeling, if Oracle has it why not SQL Server. A sequence object generates sequence of unique numeric values as per specifications. Many developers would have now got a thought, we have something similar like this called as “Identity” columns. But the big difference is sequence object is independent of a table while identity columns are attached to a table.

Below is a simple code to create a sequence object. You can see we have created a sequence object called as “MySeq” with the following specification:-

* Starts with value 1.
* Increments with value 1 Minimum value it should start is with zero.
* Maximum it will go to 100. No cycle defines that once it reaches 100 it will throw an error.
* If you want to restart it from 0 you should provide “cycle”.
* “cache 50”  specifies that till 50 the values are already incremented in to cache to reduce IO. If you specify “no cache” it will make input output on the disk.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

create sequence MySeq as int

start with 1 -- Start with value 1

increment by 1-- Increment with value 1

minvalue 0 -- Minimum value to start is zero

maxvalue 100 -- Maximum it can go to 100

no cycle -- Do not go above 100

cache 50 -- Increment 50 values in memory rather than incrementing from

IO

To increment the value we need to call the below select statement. This is one more big difference as compared to identity.In identity the values increment when rows are added here we need to make an explicit call.

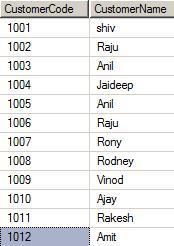
http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

SELECT NEXT VALUE FOR dbo.MySequence AS seq\_no;

Feature number 3 (Revolution):- Pagination

There are instances when you want to display large result sets to the end user. The best way to display large result set is to split them i.e.  apply pagination. So developers had their own hacky ways of achieving pagination using “top”, “row\_number” etc. But from SQL Server 2012 onwards we can do pagination by using “OFFSET” and “FETCH’ commands.

For instance let’s says we have the following customer table which has 12 records. We would like to split the records in to 6 and 6.



So doing pagination is a two-step process: -

* First mark the start of the row by using “OFFSET” command.
* Second specify how many rows you want to fetch by using “FETCH” command.

You can see in the below code snippet we have used “OFFSET” to mark the start of row from “0”position. A very important note order by clause is compulsory for “OFFSET” command.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

select \* from

tblcustomer order by customercode

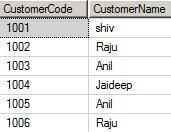
offset 0 rows – start from zero

In the below code snippet we have specified we want to fetch “6” rows from the start “0”position specified in the “OFFSET”.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

fetch next 6 rows only

Now if you run the above SQL you should see 6 rows.



To fetch the next 6 rows just change your “OFFSET” position. You can see in the below code snippet I have modified the offset to 6. That means the row start position will from “6”.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

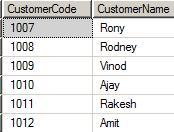
select \* from

tblcustomer order by customercode

offset 6 rows

fetch next 6 rows only

The above code snippet displays the next “6” records , below is how the output looks.



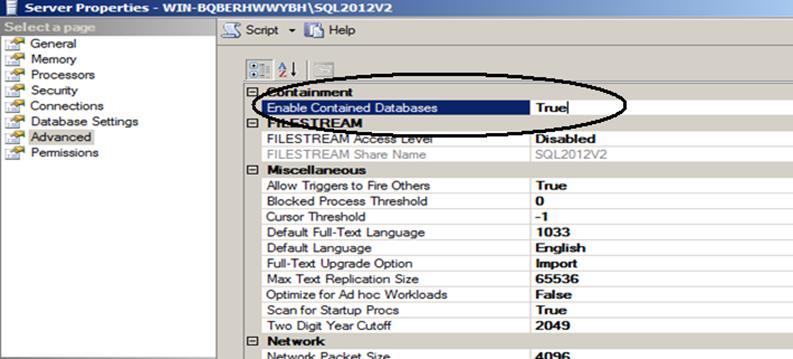
Feature number 4 (Revolution):- Contained database

This is a great feature for people who have to go through pain of SQL Server database migration again and again. One of the biggest pains in migrating databases is user accounts.  SQL Server user resides either in windows ADS or at SQL Server level as SQL Server users.  So when we migrate SQL Server database from one server to other server these users have to be recreated again. If you have lot’s of users you would need one dedicated person sitting creating one’s for you.

So one  of the requirements from easy migration perspective is  to create databases which are self-contained. In other words, can we have a database with meta-data information, security information etc with in the database itself. So that when we migrate the database, we migrate everything with it.  There’s where “Contained” database where introduced in SQL Server 2012.

Creating contained database is a 3 step process: -

**Step 1: -** First thing is to enable contained database at SQL Server instance level. You can do the same by right clicking on the SQL Server instance and setting  “Enabled Contained Database” to “true”.



You can achieve the same by using the below SQL statements as well.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

sp\_configure 'show advanced options',1

GO

RECONFIGURE WITH OVERRIDE

GO

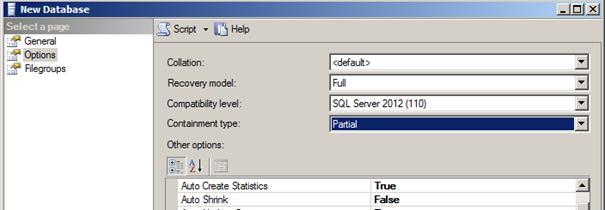
sp\_configure 'contained database authentication', 1

GO

RECONFIGURE WITH OVERRIDE

GO

**Step 2 -** The next step is to enable contained database at database level. So when create a new database set “Containment type” to partial as shown in the below figure.



You can also create database with “containment” set to “partial” using the below SQL code.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

CREATE DATABASE [MyDb]

CONTAINMENT = PARTIAL

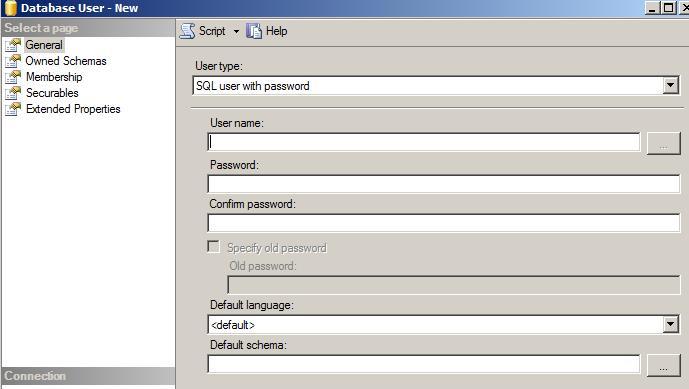
ON PRIMARY

( NAME = N'My', FILENAME = N'C:\My.mdf')

LOG ON

( NAME = N'My\_log', FILENAME =N'C:\My\_log.ldf')

**Step 3: -** The final thing now is to test if “contained” database fundamental is working or not. Now we want the user credentials to be part of the database , so we need to create user as “SQL User with password”.



You can achieve the same by using the below script.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

CREATE USER MyUser

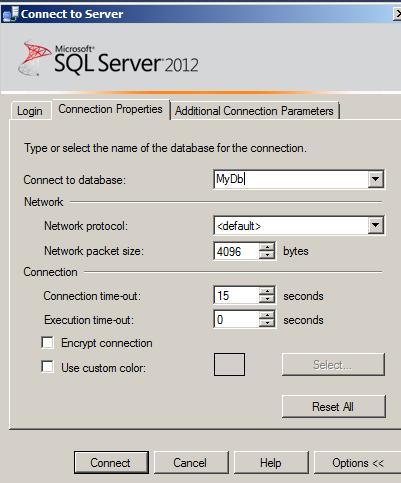
WITH PASSWORD = 'pass@123';

GO

Now if you try to login with the user created, you get an error as shown in the below figure. This proves that the user is not available at SQL Server level.



Now click on options and specify the database name in “connect to database” , you should be able to login , which proves that user is part of database and not SQL Server



Feature number 5 (Evolution):- Error handling

As a developer I am personally very comfortable with using “try/catch/throw” syntax structure for error handling in c# or vb.net. Thanks to SQL Server team in 2005 they brought in “try/catch” structure which is very much compatible the way I as a developer was doing error handling in c#.  It was nightmare handling error using “IF” conditions and “@error” code before SQL Server 2005. Below is a sample code which shows how “try/catch” code looks.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

begin try

declare @n int = 0;

set @n = 1/0;

end try

begin catch

print('divide by zero');

RAISERROR ( ‘Divide by zero‘, 16, 1) ;

end catch

But what still is itching me in the above code is when it comes to propagating errors back to the client I was missing the “THROW” command.  We still need to  use “RAISEERROR” which does the job, but lacks lot of capabilities which “THROW” has. For example to throw user defined messages you need to make entry in to “sys.messages” table.

Below is how the code with “throw” looks like.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

begin try

-- The code where error has occurred.

end try

begin catch

-- throw error to the client

Throw;

end catch

If you want to throw exception with a user defined message defined you can use the below code. No entry need in the “sys.messages” table.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/526621/Top-exciting-features-of-SQL-Server-Part)

THROW 49903, 'User define exception.', 1

From SQL Server 2012 onwards use “Throw” rather than “raiseerror” , looking at the features of “throw” looks like sooner or later “raiseerror” will be deprecated . Below is a comparison table which explains the difference between “throw” vs “raiseerror”.

|  | **Throw** | **RaiseError** |
| --- | --- | --- |
| **User & system exception** | Can generate only user exception. | Can generate user and system exception. |
| **“Sys.Messages” table** | You can supply adhoc text does not need an entry in “Sys.Messages” table. | You need to make an entry in “Sys.Messages” table. |
| **Original exception.** | Original exception is propagated to the client. | Original exception is lost to the client. |

Which are the next 5 features?

In the next part we will talk about the below features :- <http://www.codeproject.com/Articles/561797/Top-20-exciting-features-of-SQL-Server-2012-Part-2>

Feature number 6 (Evolution):- User defined roles

Feature number 7 (Evolution):- Windows server core support

Feature number 8 (Revolution):- Tabular Model (SSAS)

Feature number 9 (Revolution):- Power view

Feature number 10 (Revolution):- DQS Data quality services

If you ever get a chance do visit my site for  [.NET and SQL Server interview questions and answer](http://questpond.com/) videos.