التعامل مع أنواع البيانات Ntext و Text و image :

مع نمو قواعد البيانات سواء في الحجم أو التعقيد نوعاً ما أضحت عتاديات الحاسب و برمجياته تسمح لنا بأن نخزن حجوم هائلة من البيانات و بمختلف أنواعها من :

- الوسائط المتعددة (كالصوتيات و الرسوميات) و بلواحق عدة : JPG, PNG, MP3
- الوثائق و النصوص الكبيرة : DOC/RTF, HTML, Unicode, and XML

لهذه الأنواع من البيانات تقدم sql server الأنواع التالية : image, text, and ntext

بشكل عام يمكننا القول بأن :
النوع text : لتخزين بيانات من نوع محارف AscII فقط .
النوع Ntext : لتخزين بيانات من نوع محارف UniCode .
النوع Image : لتخزين البيانات الثنائية .

أحياناً ينتابنا بعض القلق فيما يخص حجم التخزين ، أليس كذلك ؟
أظن أن الحجوم التالية سترضيك :

النوع text يعطيك (2 ^ 31 - 1) أي 2,147,483,647 من محارف الـ non_unicode أي آسكي ..
النوع Ntext يعطيك (2 ^ 30 - 1) أي 1,073,741,823 من محارف الـ Unicode .
النوع Image يعطيك (2 ^ 31 - 1) أي 2,147,483,647 بايت .

لاحظ أخي الكريم أن حجم التخزين للنوع Ntext هو ضعف عدد المحارف المدخلة .. طبعاً لأنها محارف Unicode .

ولكن كيف يمكننا التعامل مع أنواع البيانات هذه ؟
في الحقيقة أن التعامل معها يتم باستخدام مؤشرات إلى البيانات ، وبعض الدوال المخصصة التي تسمح للمؤشر بإجراء عمليات الإضافة و القراءة و الحذف منها .الجدول التالي يوضح قوة و إمكانيات هذه الأنواع :

مالا يمكنها أن تفعله	ما يمكنها أن تفعله
العمل مع عبارات الـ DECLARE, SET, or FETCH أي لا يمكننا أن نعاملها معاملة بقية أنواع البيانات . بالطريقة التقليدية .	أن تعمل كبارامتر دخل/ خرج للإجرائيات المخزنة .
لا يمكنها أن تحل محل البيانات المعادة من دوال المستخدم . User Defined Functions	يمكنها أن تكون بارامتر دخل للدوال .
لا يمكن استخدامها مع المؤشرات Cursors في تعليمة Fetch ما لم يجرى لها تعديل .	تملك ما يعادل 8000 بايت قابلة للتحويل إلى بقية أنواع البيانات

يمكنها أن تعمل مع نوع البيانات sql_variant .	يمكنها أن تعمل بعبارة Union All .
يمكن مقارنتها و تخزينها أو حتى استخدامها في التجميع Group By . الإستثناء الوحيد للعملية هو عند استخدام Is Null (اختبار فيما إذا كانت القيمة Null) أو مع عبارة Like إلا في حالة استخدام تحويل بسيط لأنواع البيانات إلى أنواع أخرى مثل varchar و Nvarchar و Char.	
أن تشترك في عبارة Union و ذلك لأن أنواع البيانات هذه لا يمكن ترتيبها .	

الدوال الخاصة بالتعامل مع أنواع البيانات Ntext ، Text ، Image :

سأشرح بعون الله عدد من الدوال مبيناً الصيغة القواعدية لكل منها مع مثال :

الدالة : TEXTPTR :

تعيد هذه الدالة قيمة ست عشرية من 16 بايت و هي قيمة مؤشر النص الذي يشير إلى البيانات سواء في النوع Text ، Ntext ، Image .

Syntax: TEXTPTR (column)

مثال :

```
--TEXTPTR sample, create a text-pointer, see its value
create table #t (n ntext)
insert #t values('ArabTeam')
DECLARE @ptrval binary(16)
SELECT @ptrval = TEXTPTR(n) FROM #t
print @ptrval
drop table #t
```

الخرج :

```
0xFFFF6900000000004D00000001000000
```

الدالة : TEXTVALID :

تعيد الدالة هذه قيمة صحيحة و التي ستكون 1 فيما إذا كان مؤشر صحيح و إلا ستعيد 0 .

Syntax: TEXTVALID ('table.column' , text_ptr)

مثال :

```
--TEXTVALID sample, creates a text-pointer, tests it
create table #t (n ntext)
insert #t values('الفريق العربي للبرمجة')
DECLARE @ptrval binary(16), @ptrval2 binary(16)
SELECT @ptrval = TEXTPTR(n) FROM #t
if TEXTVALID('#t.n',@ptrval)=1
  print '@ptrval has a valid text pointer.'
else
  print '@ptrval has an invalid text pointer.'
if TEXTVALID('#t.n',@ptrval2)=1
  print '@ptrval2 has a valid text pointer.'
else  print '@ptrval2 has an invalid text pointer.'
drop table #t
```

الخرج :

```
@ptrval has a valid text pointer.
@ptrval2 has an invalid text pointer.
```

الدالة : SET TEXTSIZE

تقوم بتثبيت الحجم المعاد من أنواع البيانات Ntext و Text بقيمة صحيحة عند استخدام عبارة Select .

Syntax: SET TEXTSIZE { number }

مثال :

```
--SET TEXTSIZE sample
create table #t (n ntext)
insert #t values('T-Sql For ArabTeam')
SET TEXTSIZE 10--ntext is unicode, 2 bytes/character
select * from #t
SET TEXTSIZE 20--ntext is unicode, 2 bytes/character
select * from #t
drop table #t
```

الخرج :

```
T-Sql
```

```
T-Sql For
```

أما @@TEXTSIZE :

تعيد قيمة عددية صحيحة تمثل حجم البيانات المعادة من عبارة Select لأنواع البيانات Ntext ، Text .

هذه القيمة يمكن تغييرها كما سلف باستخدام العبارة : Set TextSize {Number}

Syntax: @@TEXTSIZE

مثال :

```
--@@TEXTSIZE sample
SET TEXTSIZE 10--ntext is unicode, 2 bytes/character
print @@TEXTSIZE
SET TEXTSIZE 20--ntext is unicode, 2 bytes/character
print @@TEXTSIZE
```

الخرج :

```
10

20
```

الآن نأتي إلى مرحلة استخدام دوال الإضافة و الحذف و التعديل في أنواع البيانات هذه :

الدالة : WRITETEXT :

تشبه إلى حد كبير عملية إسناد قيمة إلى حقل ما حيث تكتب هذه الدالة البيانات الجديدة بعد مسح البيانات الموجود

و تستخدم مع الأعمدة من الأنواع الثلاثة : text, ntext, image

Syntax: WRITETEXT { table.column text_ptr } [WITH LOG] { data }

مثال :

```
--WRITETEXT sample
create table #t (n ntext)
insert #t values('abc')
DECLARE @ptrval binary(16)
SELECT @ptrval = TEXTPTR(n)
FROM #t
WRITETEXT #t.n @ptrval 'Walcom'
select * from #t
drop table #t
```

الخرج :

```
Walcom
```

الدالة : UPDATETEXT :

تستخدم هذه الدالة لتعديل البيانات في الأعمدة من الأنواع الثلاثة السابقة :

Syntax: UPDATETEXT { table_name.dest_column_name dest_text_ptr } { NULL | insert_offset } { NULL | delete_length } [WITH LOG] [inserted_data | { table_name.src_column_name src_text_ptr }]

مثال :

```
--UPDATETEXT sample insertion only
create table #t (n ntext)
insert #t values('bd')
DECLARE @ptrval binary(16), @i int
SELECT @ptrval = TEXTPTR(n)
FROM #t
UPDATETEXT #t.n @ptrval 0 0 'a'--insert at beginning
select * from #t
UPDATETEXT #t.n @ptrval 2 0 'c'--insert in the middle
select * from #t
set @i=(select DATALENGTH(n) from #t)/2
--/2 only if ntext, 2 bytes/character
print @i
UPDATETEXT #t.n @ptrval @i 0 'e'--insert at the end
select * from #t
drop table #t
```

الخرج :

```
abd
abcd
abcde
```

استخدام الدالة للحذف و الحشر معاً :

```
--UPDATETEXT sample deletion+insertion
create table #t (n ntext)
insert #t values('abxyef')
DECLARE @ptrval binary(16), @i int
SELECT @ptrval = TEXTPTR(n)
FROM #t
UPDATETEXT #t.n @ptrval 2 2 'cd'--insert 2, delete 2
--chars starting at position 2
select * from #t
drop table #t
```

الخرج :

```
abcdef
```

الدالة : READTEXT

تقرأ هذه الدالة كمية محددة من البيانات من الأنواع الثلاثة (Ntext , Text , Image)

Syntax: READTEXT { table.column text_ptr offset size } [HOLDLOCK]

مثال :

```
--READTEXT sample
create table #t (n ntext)
insert #t values('T-sql arabteam2000')
DECLARE @ptrval binary(16)
SELECT @ptrval = TEXTPTR(n) FROM #t
READTEXT #t.n @ptrval 10 8
--read 8 characters starting at position 3
```

```
drop table #t
```

الخرج :

```
team2000
```

الدالة : DATALENGTH

تعيد الحجم (حجم البيانات بالبايت) لأنواع البيانات الثلاثة :

Syntax: DATALENGTH (expression)

```
--DATALENGTH sample
create table #t (n ntext)
insert #t values('1234567890')
DECLARE @i int
set @i=(select DATALENGTH(n) from #t)
--it should return the length in bytes=2*UNICODE length
PRINT @i
drop table #t
```

الخرج :

```
20
```

الدالة : PATINDEX

تعيد مكان تواجد نص ما في حقل من أحد الأنواع الثلاثة و في حال كان النص غير موجود ضمن الحقل الهدف عندها تعيد الدالة قيمة 0 .

Syntax: PATINDEX ('%pattern%' , expression)

مثال :

```
--PATINDEX sample
create table #t (n ntext)
```

```
insert #t values('Hello Tim, long time no see!')
SELECT PATINDEX('%tim%', n) FROM #t
SELECT PATINDEX('%time%', n) FROM #t
drop table #t
```

الخرج :

```
7
17
```

الدالة : CONVERT

دالة التحويل بين أنواع البيانات المعرفة لدى الكثير من مبرمجي T-Sql :

Syntax: CONVERT (data_type [(length)] , expression [, style])

مثال :

```
--CONVERT sample
create table #t (n ntext)
insert #t values('Hello Tim, long time no see!')
DECLARE @c nvarchar(5)
SET @c=(select convert(nvarchar(5),n) from #t)
print @c
drop table #t
```

الخرج :

```
Hello
```

الدالة : CAST

نفس عمل الدالة السابقة :

Syntax: CAST (expression AS data_type)

مثال :

```
--CAST sample
create table #t (n ntext)
insert #t values('Hello Tim, long time no see!')
DECLARE @c nvarchar(5)
SET @c=(select CAST ( n  AS nvarchar(5) ) from #t)
print @c
drop table #t
```

الخرج :

```
Hello
```

دعونا الآن أيها الإخوة نستعرض قليلاً بعض الاستخدامات العامة لهذه الدوال :

حفظ بيانات حقول من الأنواع الثلاثة إلى ملفات خارجية :

المثال التالي : ينشأ جدول مؤقت لكل من الأنواع text و Ntext و Image و يحشر قيمة نصية ضمن الحقول النصية ومن ثم يستدعي الإجراءات التالية savetext2file لحشر قيم النص الموجودة في حقل

الـ Text و saventext2file من أجل الحقل من نوع Ntext و saveimage2file من أجل

حقل من نوع Image . علماً أن الإجرائيات المذكورة هي من إنشائنا و سأسرد نص كل منها بعد هذه النص الذي يقوم باستدعائها .

مثال :

```
--saveText2file sample
create table ##t (n text)
insert ##t values('Hello Tim, long time no see!')
EXEC saveText2file 'c:\test.txt', '##t','n', ''
drop table ##t
```

```
--saveNtext2file sample
create table ##t (n ntext)
insert ##t values('Hello Tim, long time no see!')
EXEC saveNtext2file 'c:\test.txt', '##t','n', ''
drop table ##t

--saveImage2file sample
exec saveImage2file 'c:\Category1.bak',
'Northwind..Categories', 'Picture',
'where categoryid=1'
```

تعديل قيم الحقول لأنواع البيانات Text و Ntext و Image :

لأن الدوال TEXTPTR, WRITETEXT, and UPDATETEXT لا تسمح بتمرير بارامتراتها على صيغة متحولات من نوع جدول أو عمود (Table Or Column) لذا فإن قراءة محتويات ملف تحتاج إلى جعلنا نستخدم Dynamic Sql .

قمنا بكتابة الدالة readImageFromfile لقراءة نوعي البيانات Varchar و صورة Image من ملفات خارجية ذلك لأنها تتعامل مع البيانات بشكل ثنائي و تكتبها في الحقول الهدف بدون استخدام جداول مؤقتة لهذا الغرض . أما بالنسبة لقراءة البيانات من ملف على شكل محارف Unicode فالدالة readNtextFromfile ستفي بالغرض .

ملاحظة : أخي الكريم تأكد من وجود الملفات التي تتم قرائتها في المثالين التاليين .

مثال 1 :

```
--readImageFromfile sample
--reading a text column from a file
create table ##t (n text)
insert ##t values('Hi Tim, long time no see!')
```

```
EXEC readImageFromfile 'c:\hello.txt', '##t','n', ''
select * from ##t
drop table ##t
```

الخرج :

```
Hello
```

مثال 2 :

```
--readNtextfromfile sample
create table ##t (n ntext)
insert ##t values('Hello Tim, long time no see!')
exec readNtextFromfile 'c:\t.txt', '##t','n', ''
select * from ##t
drop table ##t
```

نفس الكود تماماً مع ملف صورة يمكن استعماله لسحب صورة من ملف إلى حقل من نوع Image .

ولكن عليك أخي الكريم أن تنتبه أخي الكريم للملاحظة التالية :

لا يجب أن يتلقى الحقل من نوع Ntext بياناته من ملف بصيغة الـ Ascii .

وألا يتلقى الحقل من نوع Text بياناته من ملف بصيغة الـ Unicode .

وذلك لأن الدوال المقدمة سيتم إدخالها على حالتها الطبيعية أي بدون أي تحويل عليها .

ملاحظة 2 : الدوال المقدمة في الأمثلة تقوم بإعادة كتابة القيمة في الحقل الذي تذكره في الإستدعاء ، لذلك فإن الدوال التالية ستقوم بالإضافة على القيمة الموجودة في الحقول .. وهي :

الدالة : readImageFromfile2 و أترك لك إنشاء دالة مشابهه لنوع البيانات Nvarchar أو Unicode كتمرين ..

ملاحظة 3 : إن عملية (الإضافة) Appending على الحقول من الأنواع السابقة تكون أسهل من (التعديل) Updating و ذلك لأن :

- عملية الإضافة تقوم باستبدال أول كتلة من الملف بالـمحتويات الأصلية للحقل و من ثم تتابع بشكل متسلسل عملية الإضافة كتابة الكتل .. الكتلة الثانية فالثالثة .. وهكذا .
- بينما عملية التعديل لا تحتاج أصلاً إلى الخطوة الأولى (استبدال البيانات السابقة) فتبدأ بحشر الكتل الأولى فالثانية في الحقل الهدف .

حفظ نص كائن من قاعدة البيانات إلى ملف :

و للقيام بذلك سنعتمد على الجداول المؤقتة تملك حقولاً من نوع Ntext من الممكن حفظ نص كل من a rule, default, unencrypted stored proc, UDF, trigger, or view إلى ملفات خارجية

حيث سينشأ الجدول المؤقت و بعدها سيتم حشر سجلات فيه باستخدام تقنية INSERT EXEC .

هذا السجل سوف يحتوي على نص الكائن المعاد من جدول الـنظام sysobjects بواسطة الإجراء sp_helptext و سنستخدم الإجراء التي أنشأناها و هي saveobj كما يلي :

```
create table #temp(s nvarchar(4000))

insert #temp

exec sp_helptext @object
```

الإجراء الذي قمنا بإنشائه (saveobj) يمكنه أن يقوم بقراءة نص أي من الكائنات التالية :

rule, default, unencrypted stored proc, UDF, trigger, or view إلى ملف نصي خارجي .

و مثل هذا الأمر طبعاً كما تعلمون أيها الأخوة مفيد جداً و خاصة من أجل أغراض النسخ الإحتياطي و توثيق القاعدة .

```
--saveobj sample, saving object hello

exec saveobj 'c:\hello.sql', 'hello'
```

الآن دعونا نتحدث عن الخطوة المعاكسة أقصد :

قراءة نص كائن من ملف :

في الحقيقة قراءة ملف حجمه أكبر من 8000 بايت يتطلب عبارات استعلام ديناميكية (Dynamic SQL) ذلك لأن العديد من الـ Buffers سوف تمتلئ طبقاً لحجم الملف الكبير . لذا قمنا بإنشاء إجراء مخزن اسمه readobj يستخدم حلقة لكي ينشأ استعلام ديناميكياً ، علماً أن الحلقة سوف تقوم بتحديد عدد المتحولات المؤقتة التي سوف تنشئها . كل متحول مؤقت سوف يمسك 8000 بايت من البيانات ، وباستخدام أسماء المتحولات المؤقتة مع محرف و زيادة الأعداد لتمييزها عن بعضها سوف يعود بالنتيجة و يحقق الهدف .

```
--readobj sample, reading object hello

exec readobj 'c:\hello.sql'
```

هذا الإجراء المخزن سوف يمنحك إمكانية قراءة كائن قاعدة بيانات من ملف Script بحجم أكبر 8000 بايت و يمكننا استخدام الإجراء من أجل:

عبارات INSERT or UPDATE طويلة أو أي من أنواع الكائنات التالية :

rule, default, unencrypted stored procedure, UDF, trigger, or view و إنشائها بعد قراءتها من الملف النصي .

وفي حال كان لديك عبارات T-SQL ذات حجم كبير فإن إجرائية مثل الإجرائية التالية يمكنها أن تنفذها :

```
CREATE PROCEDURE exec_ntext (@SQL ntext)

--Execute a gigantic SQL statement

AS

EXEC (@SQL)
```

مقارنة محتويات الحقول من أنواع البيانات السابقة :

سنستخدم لهذه الغاية الدوال (User-Defined Functions) لمقارنة الأنواع text, ntext, or image .

الطريقة الأسهل للمقارنة هي مقارنة أطوال البيانات مع بعضها ولكن هذه الطريقة ليست مضمونة النتائج خاصة في حال تساوي الأطوال لكلا محتويات العمودين اللذان تتم مقارنتهما :

```
CREATE FUNCTION  testlength (@a ntext, @b ntext)

--returns true if both inputs have the same length

RETURNS bit AS

BEGIN

declare @temp_bit bit
```

```
if datalength(@a)=datalength(@b)
  set @temp_bit= 1
else
  set @temp_bit=0
return @temp_bit
END
```

ملاحظة : المثال السابق ليس فقط لنوع البيانات ntext و إنما يمكن تطبيقه على نوعي البيانات text أو image .

الطريقة المثلى للمقارنة هي مقارنة كامل محتويات الحقلين مع بعضهما أو مجموعتين فرعيتين منهما كما يلي :

```
CREATE FUNCTION  testequality (@a ntext, @b ntext)
--returns 1 if the comparison result is true
RETURNS bit AS
BEGIN
declare @temp_bit bit
if  @a like @b
  set @temp_bit= 1
else
  set @temp_bit=0
return @temp_bit
END
```

و لزيادة قوة المقارنة السابقة يمكنك استخدام أحرف المقارنة و المطابقة (wildcard characters) لجعل الدالة بإمكانيات أعلى و إذا لم تكن تعرف أخي الكريم ما هي أحرف المطابقة المستخدمة في SQL Server

فهي :

1. % سلسلة من أي طول حتى لو كان صفري .
2. _ سلسلة من محرف واحد .
3. [] محرف واحد من مجموعة محارف

1. عشوائية [wewe]
2. أو مجموعة من محارف متسلسلة الترتيب [k-t] .
3. محرف واحد لا ينتمي إلى مجموعة محارف [^wewe] أو [^k-t] .

تحديد نوع الصورة :

هناك طريقة سريعة لكي تكشف نوع الصورة هل هي : bitmap, JPEG, PNG, or TIFF هي أن تتفحص البايتات الأولى من الصورة لتحدد فيما إذا كانت تماثل مواصفات ترويسات تلك الأنواع من الصور من حيث الهيئة .

هذه التقنية ليست سيئة و لكنها ليست ممتازة و لكنها الخطوة الأولى لكتابة دالة أكثر إحكاماً للمطابقة .

```
CREATE FUNCTION  getImageType (@a image)
--returns the type of image format
RETURNS varchar(4) AS
BEGIN
declare @out varchar(4), @temp varbinary(8)
SET @temp=convert(varbinary(8), @a)
SET @out=CASE WHEN LEFT(@temp,2)=0x424D THEN 'BMP'
WHEN LEFT(@temp,2)=0xFFD8 THEN 'JPG'
WHEN LEFT(@temp,8)=0x89504E470D0A1A0A THEN 'PNG'
WHEN (LEFT(@temp,2)=0x4949)OR(LEFT(@temp,2)=0x4D4D)
THEN 'TIFF'
ELSE '' END
return @out
END
```

كلمة أخيرة :

وبذلك نكون قد أنهينا بعون الله تعالى القسم الأول من التعامل مع أنواع البيانات في SqlServer أدعو الله كي يجعل فيه الفائدة لجميع الإخوة .

لا إله إلا الله محمد رسول الله .. لبيك يا رسول الله

نعيب زماننا و العيب فينا و ما لزماننا عيب سوانا