معرفی انواع Data type در SQL

وقتی که طراحی داخلی مشخص شده باشد، شما می توانید طراحی دیتابیس خود را در محل ایجاد جدول ها و ستون های آنها آغاز کنید. وقتی فرایند تعریف ستون های یک جدول شروع می شود، انتخاب درست انواع داده و اندازه های آنها بسیار مهم می باشد.

توضیحات

صفحات داده ی SQL Server داده ی ردیف (row data) را در خود دارد و تقریبا در 8060 بایت پوشانده می شود، گرچه یک مکانیزم سرریز ردیف وجود دارد که می توانید مشارکت داشته باشد. به طور کل شما باسد اندازه های نوع داده ی خود را تا جایی که ممکن است فشرده نگه دارید. دلیل آن نیز این می باشد که هرچه ردیف ها فشرده تر باشند، ردیف های بیشتری می توانند روی یک صفحه داده قرار بگیرند، در نتیجه I/Os کمتری لازم است تا داده مورد نیاز را برای درخواست بررسی ارائه دهد. علاوه بر این حافظه ی پنهان (cache) داده در SQL Server فضای بیشتری برای ذخیره ی داده خواهد داشت. من معمولا دیدگاه مینیمالیستی به تعاریف اندازه ی نوع داده دارم و از کوچکترین اندازه های نوع داده استفاده می کنم تا یک ستون را به طور منطقی تعریف کنم.

انواع داده نیز رسیدگی دقیق لازم دارند. آیا برنامه ی شما چند ملیتی است یا اینکه آیا شانس چند ملیتی شدن را دارد؟ در بهترین حالت ممکن است انواع یونیکد (Unicode) به شما ارائه شود (مانند nvarchar, nchar) که در مقابل انواع varchar و char می باشند. انواع یونیکد دوبرابر فضای ذخیره سازی را می گیرند، اما اجازه ی ذخیره ی کاراکترهای بین المللی را نیز می دهند، مانند Kanji ژاپنی. که گفته می شود این دارای تاثیر منفی محدودیت اندازه ی 8k می باشد.

در هنگام تعریف کاراکتر براساس ستون ها یک آیتم مطرح می شود، من معمولا از char/nchar استفاده نمی کنم، مگر اینکه اندازه ی داده درستون به یک فرمت ثابت و شناخته شده باشد. دلیل آن این است که زمانی که یک ستون با عنوان char/nchar تعریف می شود، کل اندازه ی ستون اختصاص داده می شود و در مقابل محدودیت صفحه ی 8 k قرار می گیرد. برای مثال اگر قرار بود ستونی با نام FirstName را به عنوان char(50) تعریف کنید، تمام 50 کاراکتر اختصاص داده می شوند، حتی اگر اولین نام که وارد می شود کمتر از 50 کاراکتر باشد. البته این در صفحه ی داده ی شما یک فضای تلف شده می باشد و باعث می شود ردیف های کمتری مناسب باشند و درنتیجه I/Os بیشتری بررسی ها را به نتیجه می رسانند.

به مثال زیر توجه کنید که در آن یک جدول با یک ستون با عنوان char(2000) ایجاد شده و با 1000 ردیف تغذیه می شود (seed).

```
use tempdb
go
if object_id('SizeTest') is not null
  drop table SizeTest
go
create table SizeTest (id int identity primary key clustered, value char(2000) default 'x')
```

```
go
insert into SizeTest default values
go 1000
```

با امتحان کردن تعداد صفحه مشاهده می کنیم که یک مجموعه ی کلی از 250 صفحه اختصاص داده شده اند.

```
use tempdb
go
select page_count
from sys.dm_db_index_physical_stats(DB_ID('tempdb'),OBJECT_ID('SizeTest'),NULL,NULL,NULL)
go
```

در مقابل اجازه بدهید جدول را بازسازی کرده و مقدار ستون را با عنوان varchar(2000) تعریف کنیم.

```
use tempdb
go
if object_id('SizeTest') is not null
  drop table SizeTest
go
create table SizeTest (id int identity primary key clustered, value varchar(2000) default 'x')
go
insert into SizeTest default values
go 1000
```

اجرای مجدد query در مقابل sys.dm_db_index_physical_stats یک کاهش بزرگ در ذخیره ی ردیف های لازم نشان می دهد.

همچنین در نظر داشته باشید از انواع داده های منسوخ از قبیل متن، ntext و تصویر اجتناب کنید، چرا که آنها در نهایت پشتیبانی نمی شوند. در عوض varchar(max), nvarchar(max) و varbinary را تغییر داده و شروع به استفاده از آنها کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد7

88146323 - 88446780 - 88146330

http://www.tahlildadeh.com/