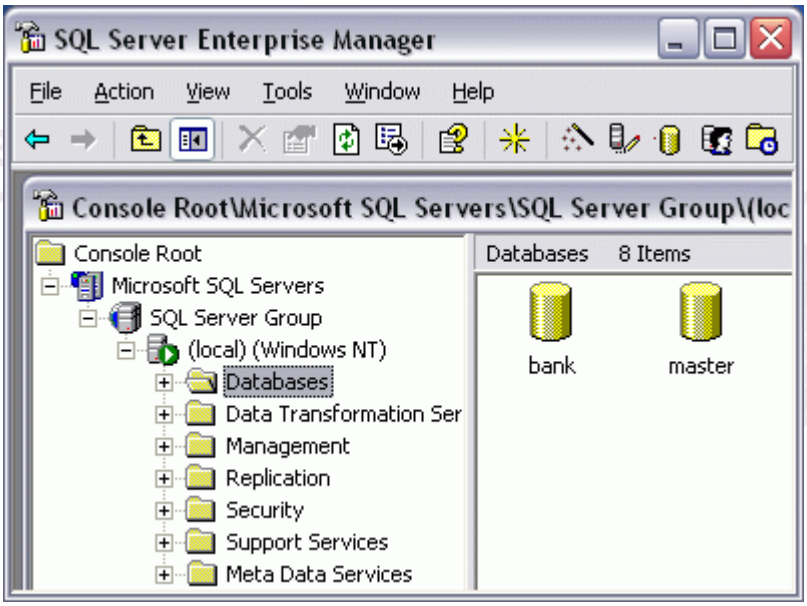
فصل ششم :
آشنایی با زبان SQL و مقدمات SQL-Server
(قسمت اول)



مقدمه :

چون اکثر فصول آتی و عمده ی توانایی ASP.NET در برنامه نویسی دیتابیس خلاصه می شود و بدون آن تقریبا مفهوم خودش را از دست خواهد داد و با توجه به سهولت دسترسی به SQL-Server در ایران ، در فصل جاری فارغ از مباحث ASP.NET مروری خواهیم داشت بر زبان SQL و محیط SQL-Server تا در طی فصول آتی مشکل خاصی (حداقل در مبانی کار) وجود نداشته باشد. اگر با این

مباحث آشنا هستید به راحتی می توانید مطالعه ی فصول آتی را شروع نمایید. پیش فرض این فصل آن است که شما SQL-Server را بر روی سیستم خودتان نصب کرده اید. اگر از ویندوز ۲۰۰۰ Advanced Server استفاده می نمایید در نصب نگارش های مختلف SQL-Server مشکلی نخواهید داشت ولی اگر از win2000 pro و یا XP Pro استفاده می کنید بهتر است از SQL-Server Desktop Edition استفاده نمایید تا بتوان از قابلیت های سروری آن نیز استفاده نمود.

طراحی و ایجاد یک بانک اطلاعاتی :

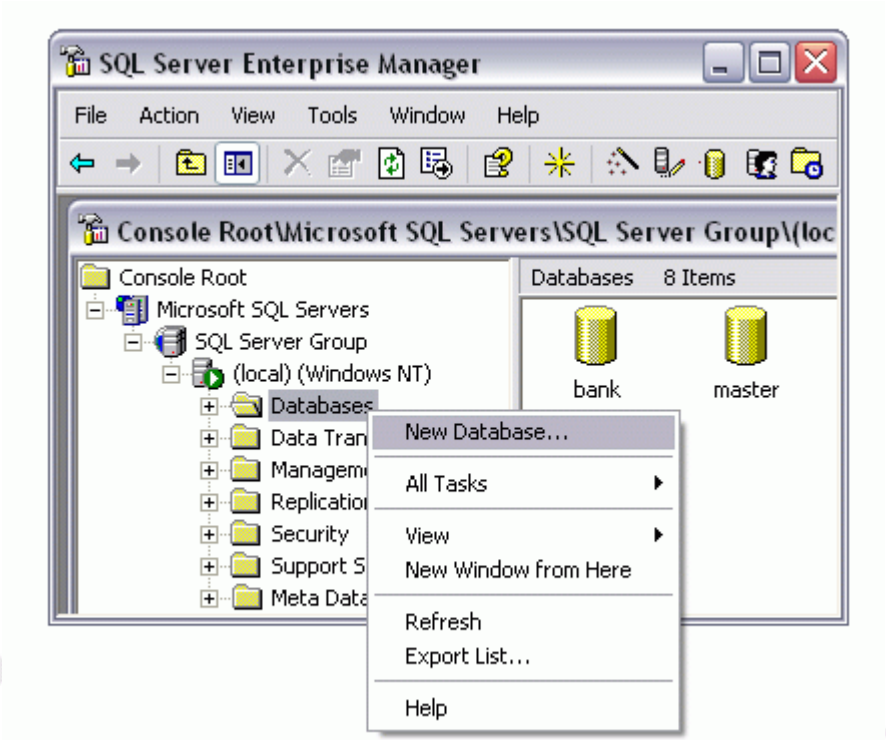
کلید توسعه ی یک وب سایت فعال ، داده ها می باشند. یک وب سایت پویا با بانک اطلاعاتی معنا و مفهوم واقعی خودش را پیدا می کند. برای ایجاد یک بانک اطلاعاتی در SQL-Server ابتدا Enterprise manager آنرا اجرا کنید و سپس روی DataBases آن کلیک راست نمایید (شکل ۱). گزینه ی ایجاد یک بانک اطلاعاتی جدید را انتخاب کنید تا صفحه ی دیالوگ وارد کردن نام دیتا بیس جدید باز شود. نامی دلخواه را وارد و سپس Ok کنید. اکنون یک بانک اطلاعاتی خام به مجموعه ی بانک اطلاعاتی SQL-Server اضافه شده است و نیاز می باشد تا جداول دلخواه را برای مدیریت اطلاعات در آن ایجاد نماییم.

روی دیتابیس جدید کلیک کنید (شکل ۲) و گزینه ی طراحی جدول جدید را انتخاب نمایید تا صفحه ی ایجاد فیلدها ظاهر شود (شکل ۳). در اینجا می توانید به سادگی فیلدها ، نوع ، طول و آیا Null را بپذیرد و یا خیر و موارد دیگر را تنظیم نمایید.

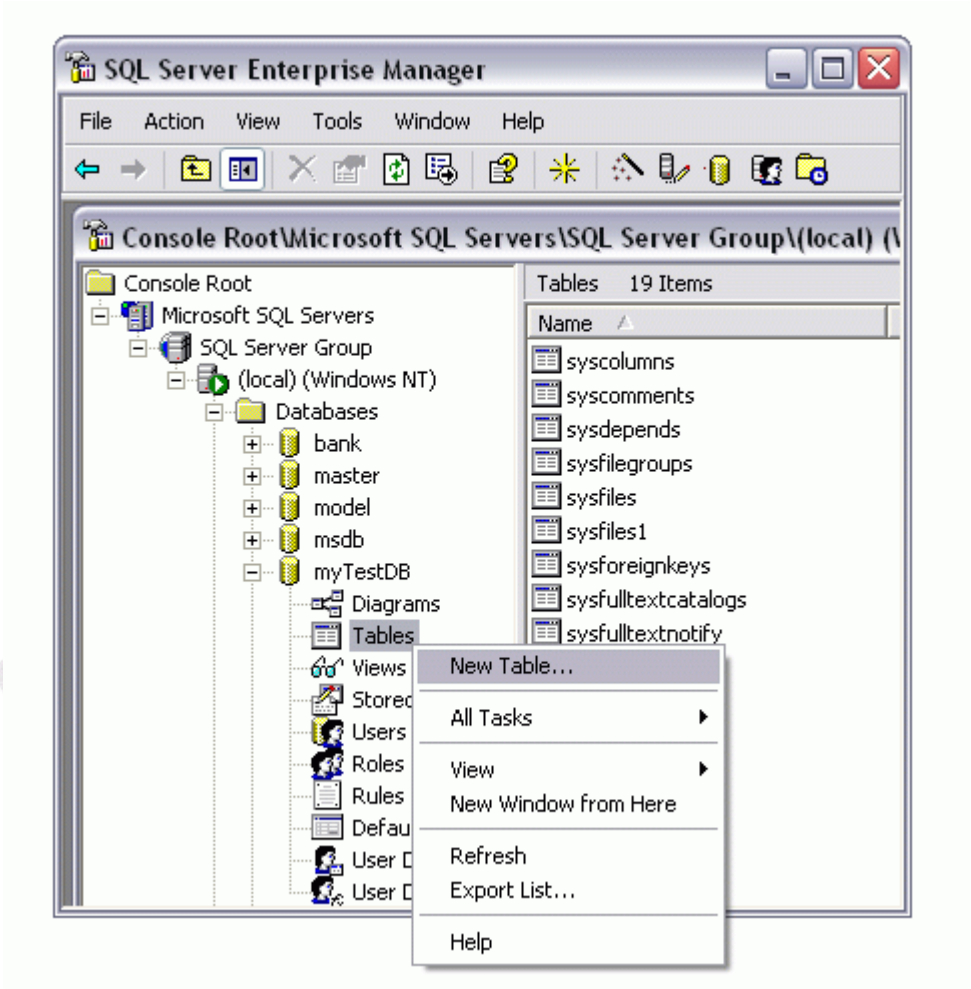
سپس این صفحه را ببندید تا یک صفحه ی دیالوگ دیگر برای وارد کردن نام جدول ظاهر شود (شکل ٤) . نام پیش فرض را بپذیرد.

برای مثال این دیتابیس را ایجاد کنید:

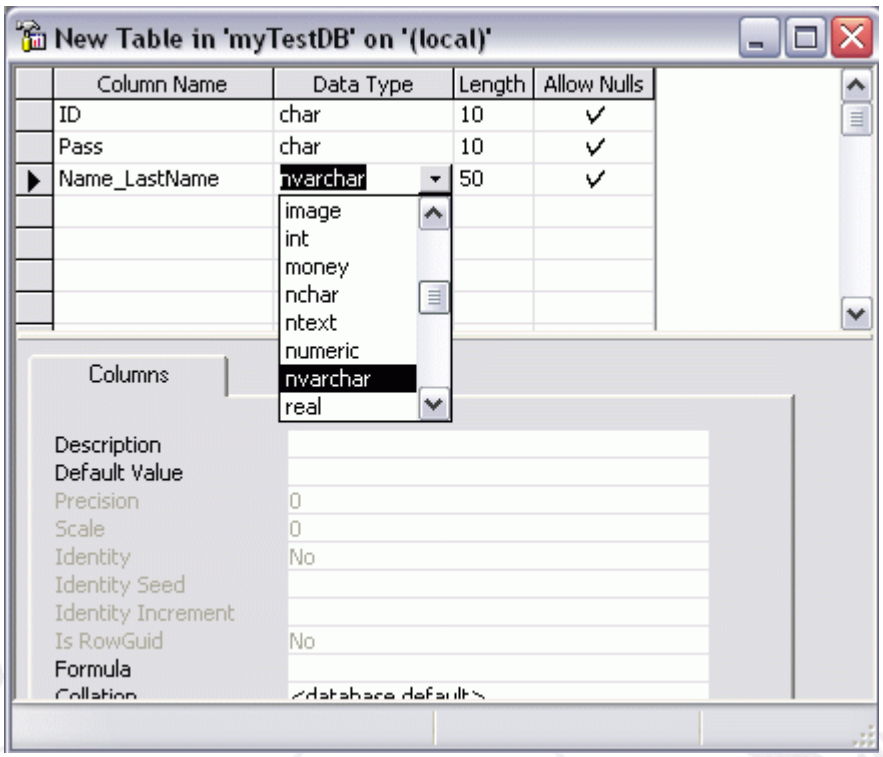
بانک اطلاعاتی با سه فیلد ID, Pass, Name_LastName . سایر مشخصات آنها را هم می توانید در تصاویر مشاهده نمایید.



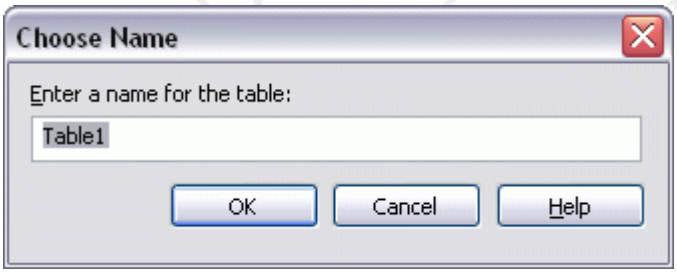
شکل ۱- اولین قدم برای ایجاد یک بانک اطلاعاتی جدید در SQL-Server .



شکل ۲- اولین قدم برای ایجاد جدولی جدید در SQL-Server .



شکل ۳- طراح دیتا بیس در SQL-Server .



شکل ٤- ورود نامی برای جدول جدید ایجاد شده.

چند نکته :

۱- اگر می خواهید داده های یونیکد مانند متن فارسی را در بانک اطلاعاتی وارد کنید بهتر است نوع فیلدهایی را انتخاب نمایید که با n شروع می شوند. n در اینجا به معنای national است.

۲- اگر در هنگام برنامه نویسی ، یک فیلد (مثلا توضیح) اهمیت آنچنانی برای ورود اطلاعات ندارد می توانید آنرا در هنگام طراحی دیتابیس Allow Null نمایید. درغیر اینصورت اگر فیلدی اینگونه نباشد حتما باید با داده ای پر شود و گرنه با یک خطا مواجه خواهید شد. برای اطمینان حاصل کردن از این موضوع می توان به سادگی از کنترل های تعیین اعتبار که در مورد آنها توضیح داده شد استفاده نمود.

۳- فرض کنید فیلدی به نام Name به طول ۵۰ ایجاد کرده اید. بهتر است در TextBox ایی که می خواهد این فیلد را از کاربر دریافت کند خاصیت MaxLength را مساوی ۵۰ قرار دهید تا بازهم برنامه دچار خطای زمان اجرا نشود. در غیر اینصورت برای مثال اگر کاربر ۵۱ کاراکتر وارد کند حتما برنامه با یک خطا متوقف می شود و متاسفانه این نوع خطایابی و رفع مشکلات آن بسیار مشکل می باشد. پس بهتر است علاج واقعه قبل از وقوع شود.

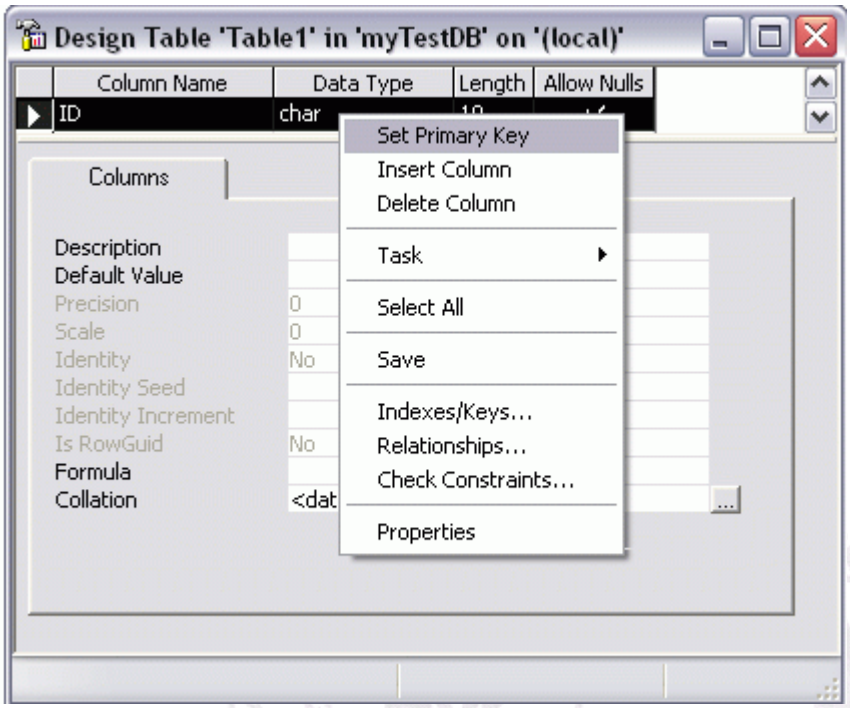
٤- تجربه نشان داده است که برای وارد کردن تاریخ فارسی بهتر است از نوع کاراکتر استفاده شود و قرار دادن یک صفر قبل از اعداد یک رقمی را فراموش نکنید تا بتوان به راحتی آمارهای از تاریخ تا تاریخ را بدست آورد. برای مثال بجای ۸۲/۱/۱ می توان نوشت ۸۲/۰۱/۰۱ .

٥- هنگام نامگذاری یک جدول در SQL-Server بهتر است یک tbl ویا t_ به قبل از نام جدول اضافه نمایید. این کار هنگامیکه تعداد جداول بانک شما زیاد می شود اهمیت خودش را نشان می دهد و جداول شما لابلای جداول سیستمی SQL-Server قرار نخواهد گرفت (در لابلای لیست آنها).

٦- هنگام طراحی جدول حتما سعی کنید یک فیلد منحصر بفرد ایجاد کنید به نام Primary key (شکلهای ٥ و٦) . برای مثال حداقل یک فیلد ردیف که با اعداد متوالی پر می شود ایجاد نمایید. SQL-Server در مورد یکتا بودن داده های وارد شده در این فیلد مراقبت های لازم را انجام خواهد داد!

۷- یکی دیگر از روش های مقید سازی داده ها و اطمینان حاصل کردن از یکپارچگی دیتابیس استفاده از Foreign key می باشد. یک Foreign key برای ایجاد ریفرنس Primary key جدول دیگر

بکار برده می شود و بدین وسیله اطمینان حاصل خواهد شد که داده ها در ستون کلیدخارجی جدول ارجاع داده شده حضور خواهند داشت.



شکل ٥- نحوه ی ایجاد Primary Key برای جدول (ابتدا فیلد را انتخاب کنید و سپس روی آن کلیک راست نمایید) .

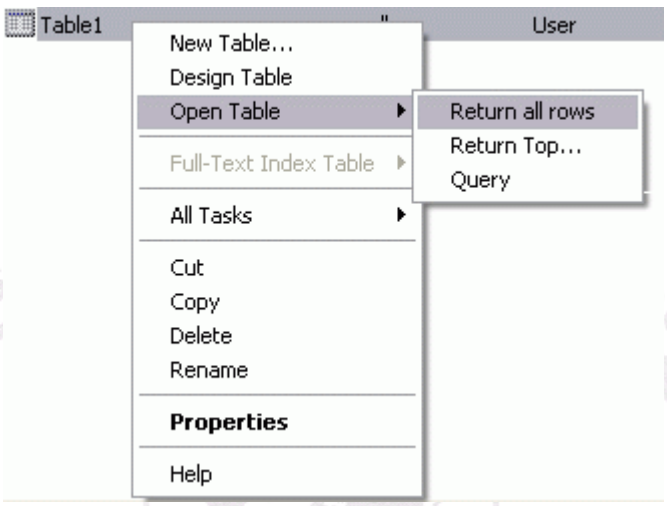
	Column Name	Data Type	Length	Allow Nulls
▶🔑	ID	char	10	

شکل ٦- شکل جدول علامت کلید در کنار فیلدی که Primary key شده است .

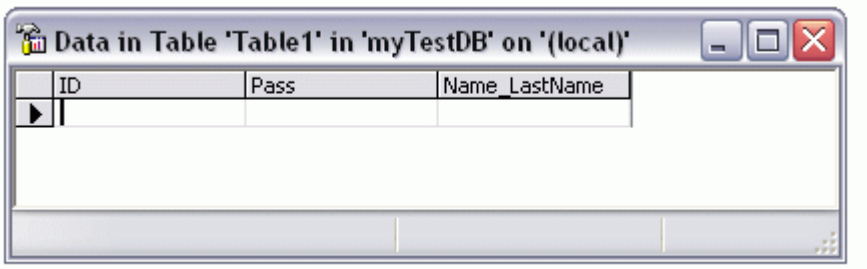
برای گرفتن کوئری و یا پرسجو روی جدول در SQL-Server راه حلهای زیادی وجود دارد. برای مثال روی جدول کلیک راست نموده و از گزینه ی Open Table آن گزینه ی Return all rows را انتخاب

کنید (شکل ۷). صفحه ای باز خواهد شد (شکل ۸) که تمام رکوردهای جدول را نمایش می دهد. در اینجا می توان داده ها را به صورت دستی هم وارد کرد.

اگر به حفظ کردن دستورات SQL علاقه ای ندارید می توان به صورت ویژوال این دستورات را تولید نمود و آزمایش کرد. بر روی نام جدول کلیک راست کنید واز گزینه ی Open Table گزینه ی Query را انتخاب نمایید (شکل ۹).



شکل ۷- نحوه ی گرفتن یک کوئری ساده بر روی جدول .



شکل ۸- پس از کلیک کردن روی Return all rows رکوردهای جدول نمایش داده می شود.

نرمال سازی داده ها :

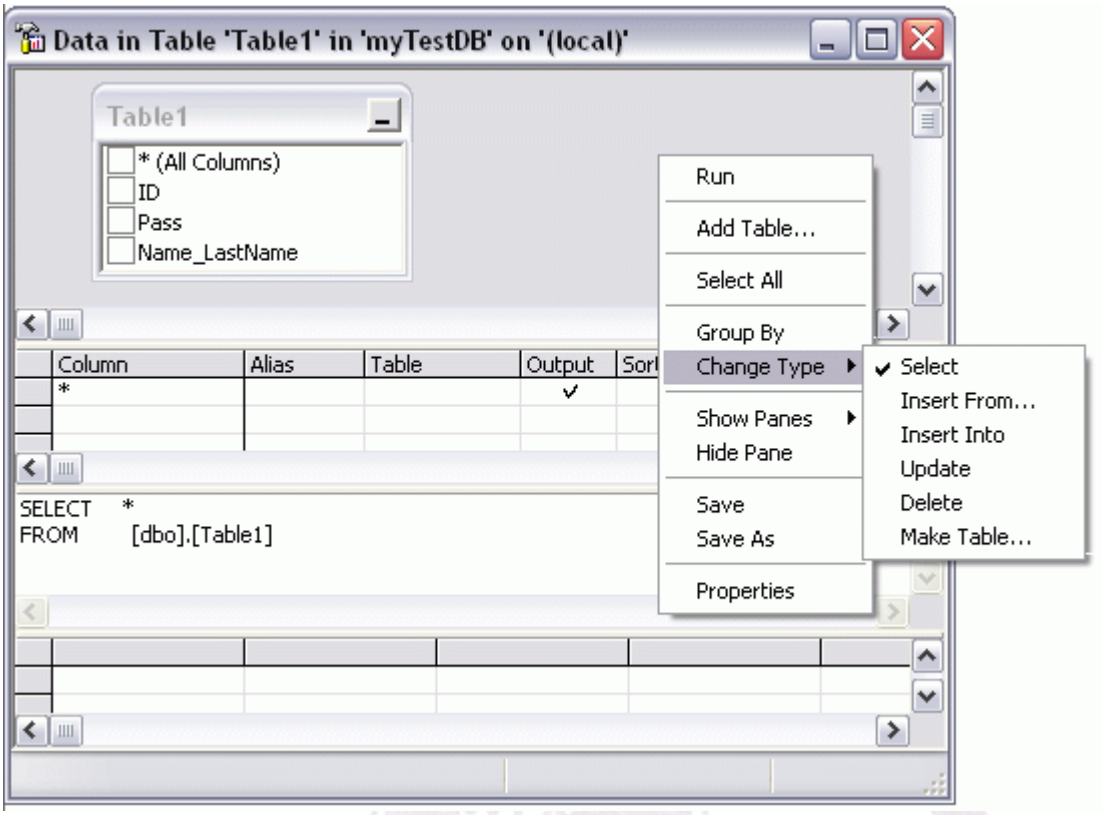
برای توضیح دادن این قسمت می توان یک دیتابیس با مشخصات زیر ایجاد کرد (نام جدول آن t_band است) :

```
[band_id] [int] IDENTITY (1, 1) NOT NULL
[band_title] [varchar] (100) NOT NULL
[music_type_id] [int] NOT NULL
[record_company_id] [int] NOT NULL
```

عموما برای ارزیابی طراحی یک جدول ، اطمینان حاصل می شود که آیا Normalized شده است یا خیر. Normalization پروسه ای است که در آن داده ها در جداول مرتبط قرار می گیرند و بدین وسیله داده های زاید و تکراری حذف می گردند. قوانین متعددی برای انجام اینکار وجود دارد که به آنها Normalization forms گفته می شود. سه فرم ابتدایی آن به صورت زیر هستند:

فرم اول نرمال (FNF) :

مطابق با این قاعده یک ستون در جدول نمی تواند حاوی داده های چندگانه باشد. برای مثال در جدول فوق یک کاربر می تواند چندیدن آلبوم را بخرد و برعکس. پس بهتر است کاربر و آلبوم تجزیه شوند.



شکل ۹- محیط طراحی کوئری در SQL-Server .

فرم دوم نرمال (SNF) :

هر ستونی که کلید نیست باید وابسته باشد به entire key و نه فقط Primary key . کـه در ایـن جـدول رعایت شده است.

فرم سوم نرمال (TNF):

ستون هایی که کلید نیستند نباید به سایر ستون هایی که آنها هم کلید نیستند وابسته باشند.

ایده ی بهتر در مورد این جدول بدین صورت است که آنها را به جداول کوچکتر تقسـیم کنـیم و از طریـق Foreign key آنها را به هم ارتباط دهیم (توضیحات بیشتر برای سنگین تر نشدن این فصل در طی فصل آتی که ادامه ی فصل جاری می باشد ارائه خواهد شد) .

مطلبی را که باید بخاطر داشت این است که تا حد امکان باید از تعداد جداول زیاد پرهیز کرد چون کارآیی را کاهش می دهد و نیاز به Join های زیادی در ادامه وجود خواهد داشت . در هر حال طراحی یک بانک اطلاعاتی بیشتر یک هنر است تا علم!

Query Analyzer :

برای ایجاد یک بانک و جدول همانطور که گفته شد یا می توان از Enterprise manager استفاده کرد و یا با استفاده از برنامه Query Analyzer که همراه SQL-Server نصب می شود دستورات T-SQL را نوشت و کار ایجاد بانک و جدول و موارد دیگر را انجام داد.

هنگامیکه Query Analyzer را اجرا می کنید ابتدا صفحه ی اتصال به SQL-Server ظاهر می شود (شکل ۱۰). به طور معمول هنگام نصب SQL-Server کاربری با LoginName=sa و پسورد خالی ایجاد می شود (یکی از مواردی که باید هنگام قرار دادن داده ها روی شبکه به آن دقت شود تا هکرها از آن سوء استفاده نکنند) . یک راه دیگر هم برای اجرای Query Analyzer وجود دارد. در محیط Enterprise manager از منوی Tools که در بالای صفحه قرار دارد Query Analyzer را انتخاب نمایید.

در این محیط بهتر است قبل از هرکاری دیتابیسی را که می خواهید روی آن کار کنید انتخاب نمایید تا دستورات شما روی دیتابیس دیگری اجرا نشود (شکل ۱۱).

در محیط آن با فشردن دکمه F5 دستورات اجرا می شوند .

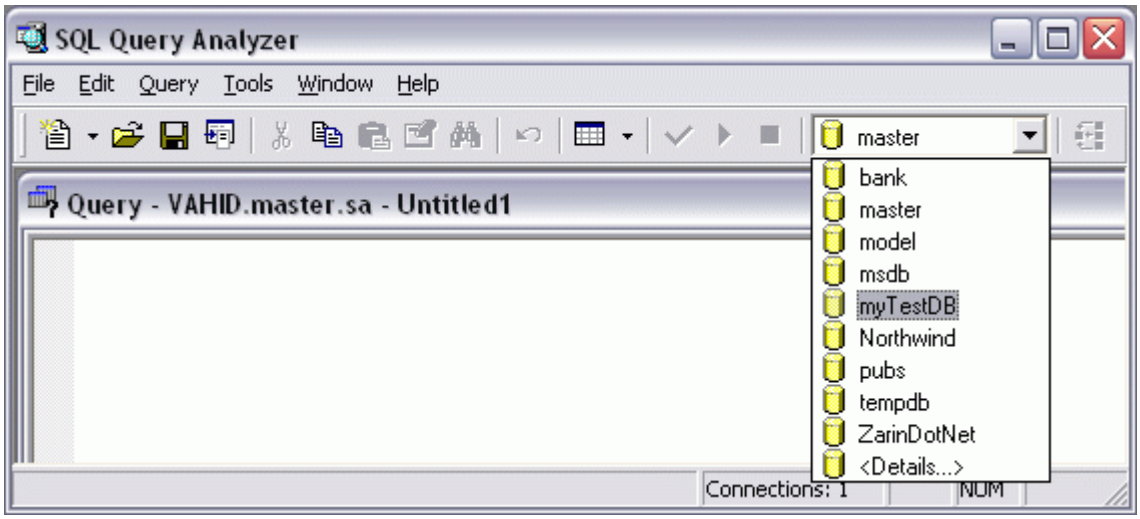
برای ایجاد دیتا بیس و تولید جدول از دستورات زیر هم می توان استفاده کرد :

```
USE master
GO
CREATE DATABASE Music ON PRIMARY
( NAME = MusicData,
FILENAME = 'C:\MSSQL7\data\MusicData.mdf'
)
```

و در ادامه :

```
USE Music
GO
CREATE TABLE [dbo].[t_bands] (
[band_id] [int] IDENTITY (1, 1) NOT NULL ,
[band_title] [varchar] (100) NOT NULL ,
[music_type_id] [int] NOT NULL ,
[record_company_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[t_bands] WITH NOCHECK ADD
CONSTRAINT [PK_t_bands] PRIMARY KEY NONCLUSTERED
(
[band_id]
) ON [PRIMARY] ,
CONSTRAINT [IX_bands_title] UNIQUE NONCLUSTERED
(
[band_title]
) ON [PRIMARY]
GO
```

شکل ۱۰- صفحه ی اتصال به SQL-Server برای اجرای Query Analyzer .



شکل ۱۱ – محیط Query Analyzer . قبل از هر چیز دیتابیسی را که می خواهید روی آن کار کنید را مشخص نمایید.

ایجاد View :

یک View اساسا یک شیء SQL-Server است و تعیین می کند که یک کاربر داده ها را به چه شکلی می تواند ببیند. View یک پرسجوی (Query) ذخیره شده است. View ها برای موارد امنیتی هم مفید بوده (در این حالت کاربر شی های View را می بیند و نه جداول را) و همچنین برای ساده سازی Query هایی که زیاد کاربرد دارند. ایجاد View با هر دوی Query analyzer و Enterprise Manager امکان پذیر است .

برای مثال :

```
CREATE VIEW [owner.]view_name
AS select_statement
```

برای مثال فرض کنید که می خواهید View ایی را درست کنید که نام تمام bands را در جدول t_babds ارائه دهد.

```
USE Music
GO
CREATE VIEW all_bands
AS
SELECT * FROM t_bands
```

ایجاد Stored procedures :

یک رویه یا دستور العمل ذخیره شده ، یک جمله ی T-SQL پیش کامپایل شده است. بدلیل این پیش کامپایل شدن ، آنها کارآیی بهتری را نسبت به View و سایر انواع کوئری ها ارائه می دهند. بعلاوه امکان پاس کردن متغیر به و یا از آنها وجود دارد.

روش ایجاد :

```
CREATE PROCEDURE procedure_name
[{@parameter_name data_type} [VARYING] [= default] [OUTPUT]]
[, ...n]
AS
sql_statement
```

مثال :

```
CREATE PROCEDURE pr_albums
AS
SELECT album_title FROM t_albums ORDER BY album_title
```

برای اجرای یک رویه ذخیره شده شما می توانید در Query analyzer ابتدا بنویسید Exec (مخفف Execute) و سپس نام رویه ذخیره شده و سپس دکمه ی F5 را فشار دهید.

مثال زیر یک آرگومان را هم می پذیرد :

```
CREATE PROCEDURE pr_albums2
@iBandID INT
AS
SELECT album_title
FROM t_albums
WHERE band_id = @iBandID
ORDER BY album_title
```

ایجاد Triggers :

Trigger یک نوع رویه ذخیره شده ی خاص است که به صورت اتوماتیک invoked می شود تا از تغییرات و اصلاحات ناخواسته جلوگیری کند. Triggers کمک می کنند تا از یکپارچگی داده ها اطمینان حاصل شده و از تغییراتی که این مورد را به خطر می اندازد جلوگیری شود. برای مثال می توان اطمینان حاصل کرد که یک رکورد را نمی توان هنگامیکه در جدول دیگر از ریفرنس آن دارد استفاده می شود delete کرد .

Trigger ها پارمتر ندارند و به صورت صریح قابل اجرا نیستند. آنها هنگامی اجرا می شوند که شما سعی در Insert ، Update یا Delete داده ها به / از جدول داشته باشید.

نحوه ی ایجاد :

```
CREATE TRIGGER trigger_name
ON table_name
FOR {INSERT | UPDATE | DELETE}
AS sql_statement
```

و برای مثال :

```
CREATE TRIGGER trg_DeleteBand
ON t_bands
FOR DELETE
AS
IF EXISTS(SELECT album_id FROM t_albums, deleted WHERE t_albums.band_id
=
deleted.band_id)
BEGIN
RAISERROR(Band has albums!',16,1)
END
```

تمرین :

۱- جدول t_band را ایجاد نمایید و تمام مواردی را که در مورد آن عنوان شد بعنوان تمرین در Query analyzer آزمایش نمایید. در صورت نیاز به اطلاعات وارد شده در بانک اطلاعاتی روش وارد کردن اطلاعات به صورت دستی هم درمتن توضیح داده شد.