عنوان: **افزودن یک DataType جدید برای نگه‌داری تاریخ خورشیدی - 2**
نویسنده: حامد قنادی
تاریخ: ۱۳۹۲/۰۲/۱۰ ۲۱:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, SQL Server 2012, Persian, CLR, SQL

پیش از هرچیز به شما پیش‌نهاد می‌کنم؛ بار دیگر کد سی‌شارپ درس نخست را در پروژه‌ی خود کپی کنید و سپس Publish را بزنید. پس از ارسال آن مطلب، تغییراتی در جهت بهینه‌سازی کد دادم که به نظرم بهتر است شما نیز در پروژه‌ی خود به کار برید.

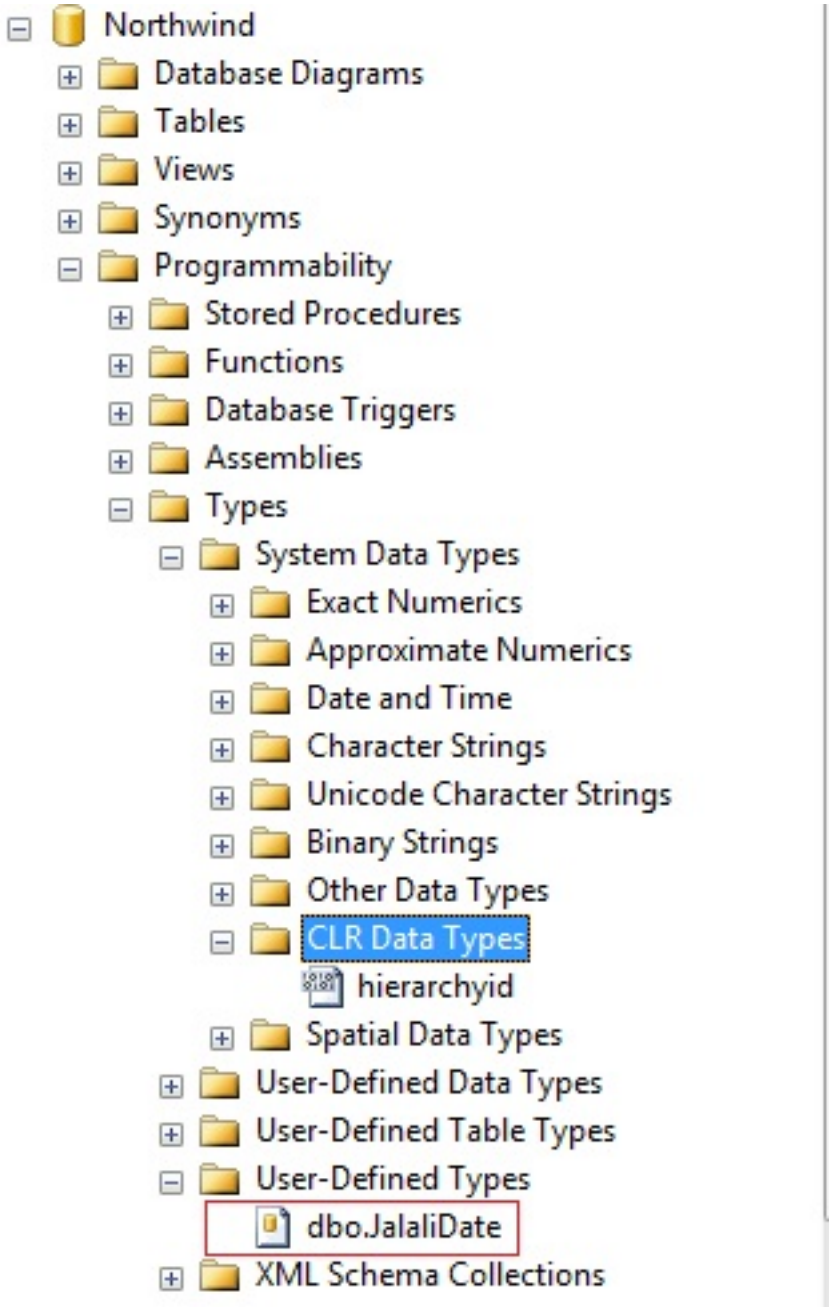
چرا از این نوع داده استفاده کنیم؟

نخستین پرسشی که ممکن است برای شما پیش بیاید این است که چرا بهتر است از این نوع داده استفاده کنیم. برای پاسخ به این پرسش باید راه‌کارهای گذشته را بررسی کنیم. معمولاً طراحان پایگاه داده‌ها برای استفاده از تاریخ خورشیدی، زمان را به صورت میلادی ثبت می‌کنند؛ سپس با یک scalar-valued function زمان درج شده را به خورشیدی تبدیل می‌کنند. در این صورت می‌توان با یک تابع کوچک دیگر بخش مربوط به ساعت را نیز از همان ستون به دست آورد. در این صورت می‌توانیم از کلیه‌ی متدهای مربوط به DateTime در SQL از جمله افزایش و کاهش و تفاضل دو تاریخ بهره برد. برخی دیگر از طراحان، ستونی از نوع char(10) در نظر می‌گیرند و تاریخ خورشیدی را به صورت ده‌کاراکتری در آن ذخیره می‌کنند. این روش هرچند نیاز به تبدیل به خورشیدی را ندارد ولی کلیه‌ی مزایایی که در استفاده از DateTime به آن‌ها دسترسی داریم از دست می‌دهیم. افزون بر این جهت نگه‌داری زمان باید یک فیلد دیگر از نوع کاراکتری و یا در نگارش‌های نوین‌تر از نوع time تعریف کنیم. برخی دیگر از هر دو را در کنار هم استفاده می‌کنند و در واقع جهت سرعت بالاتر نمایش و بررسی داده‌ها از طریق محیط SQL Server از فیلد کاراکتری تاریخ خورشیدی و برای مقایسه و بدست آوردن ساعت از فیلد نوع DateTime استفاده می‌کنند.

از نظر فضای اشغال‌شده نوع DataTime، هشت بایت، smalldatetime (در صورت استفاده) 4 بایت و فیلد 10 کاراکتری تاریخ 10 بایت فضا اشغال می‌کند در صورتی که نوع JalaliDate با درنظر گرفتن همه‌ی مزایای انواع داده‌ی استفاده‌شده برای تاریخ، فقط 8 بایت فضا اشغال می‌کند. با استفاده از این نوع به راحتی داده‌ی تاریخ را بر اساس تقویم ایرانی اعتبارسنجی می‌کنید و بخش‌های مختلف زمان از سال تا ثانیه را با یک متد به دست می‌آورید. می‌توانید به راحتی به تاریخ خود زمانی را بیفزایید یا بکاهید و در گزارش‌ها بدون نگرانی از تبدیل درست استفاده کنید. چون کدباز است می‌توانید با کمی حوصله امکانات دیگر مد نظر خود را به آن بیفزایید و از آن در SQL بهره ببرید.

چگونه این نوع داده را حذف کنم!؟

شما می‌توانید به سادگی نوع داده‌ی ایجادشده توسط CLR را در مسیر زیر بیابید و اقدام به حذف آن نمایید:



همان‌طور که مشاهده می‌شود؛ حتی نوع داده‌ی سیستمی hierarchyid که جهت ساختار سلسله‌مراتبی مانند چارت سازمانی یا درخت تجهیزات استفاده می‌شود؛ نیز یک نوع داده‌ی CLR است.

آیا راه دیگری نیز برای افزودن این نوع داده به SQL به جز Publish کردن وجود دارد؟

مانند بسیاری دیگر از گونه‌های پروژه، در اینجا نیز شما یک فایل DLL خواهید داشت. این فایل برپایه‌ی تنظیماتی که شما در قسمت Properties پروژه‌ی خود انجام می‌دهید ساخته می‌شود. پس از تغییر مسیر فایل DLL در دستور زیر توسط یک New Query از Database خود، آن را اجرا کنید:

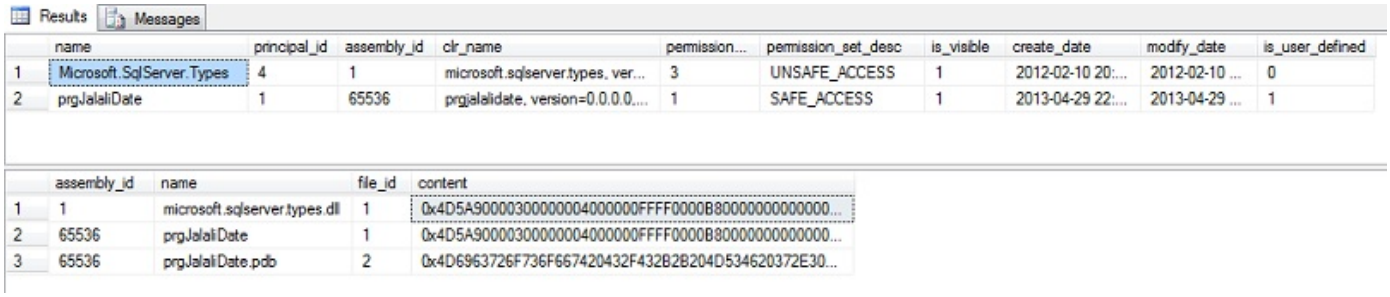
```
CREATE ASSEMBLY JalaliDate
FROM 'F:\prgJalaliDate.dll'
WITH PERMISSION_SET = SAFE;
```

هم‌چنین در صورت ویرایش‌های دوباره پروژه از دستور زیر استفاده کنید:

```
ALTER ASSEMBLY JalaliDate
FROM 'F:\prgJalaliDate.dll'
```

با استفاده از دستورهای زیر می‌توانید از چگونگی درج فایل‌های افزوده شده آگاه شوید:

```
select * from sys.assemblies
select * from sys.assembly_files
```



Results Messages

	name	principal_id	assembly_id	clr_name	permission...	permission_set_desc	is_visible	create_date	modify_date	is_user_defined
1	Microsoft.SqlServer.Types	4	1	microsoft.sqlserver.types, ver...	3	UNSAFE_ACCESS	1	2012-02-10 20:...	2012-02-10 ...	0
2	prgJalaliDate	1	65536	prgjalalidate, version=0.0.0.0,...	1	SAFE_ACCESS	1	2013-04-29 22:...	2013-04-29 ...	1

	assembly_id	name	file_id	content
1	1	microsoft.sqlserver.types.dll	1	0x4D5A90000300000004000000FFFF0000B800000000000000...
2	65536	prgJalaliDate	1	0x4D5A90000300000004000000FFFF0000B800000000000000...
3	65536	prgJalaliDate.pdb	2	0x4D6963726F736F667420432F432B2B204D534620372E30...

تا اینجا SQL Server، دی‌ال‌ال مربوط به پروژه را شناخته است. برای تعریف نوع داده از دستور زیر بهره ببرید:

```
CREATE TYPE dbo.JalaliDate
EXTERNAL NAME JalaliDate.[JalaliDate];
```

این کار همانند استفاده از گزینه‌ی Publish در Visual Studio است.
هم‌چنین چنان‌چه در SQL Server 2012 از منوی راست‌کلیک پایگاه داده‌ها روی گزینه Tasks و سپس Generate Scripts را انتخاب کنیم، از مشاهده‌ی سند ساخته شده، درخواهیم یافت که حتی دستورهای مربوط به ساخت اسمبلی CLR با تبدیل فایل به کد در Scripts وجود دارد و با اجرای آن در سروری دیگر، انتقال می‌یابد.

```
GO
/****** Object:  SqlAssembly [prgJalaliDate]    Script Date: 2013/04/30 08:27:00 ب.ظ ******/
CREATE ASSEMBLY [prgJalaliDate]
FROM 0x4D5A90000300000004000000FFFF0000B8000000000000 ..... بقیه‌ی کدها حذف شده
WITH PERMISSION_SET = SAFE

GO
ALTER ASSEMBLY [prgJalaliDate]
ADD FILE FROM 0x4D6963726F736F667420432F432B2B204D534620372E30300D0A1A44530..... بقیه‌ی کدها حذف شده
AS N'prgJalaliDate.pdb'

GO
/****** Object:  UserDefinedType [dbo].[JalaliDate]    Script Date: 2013/04/30 08:27:00 ب.ظ ******/
CREATE TYPE [dbo].[JalaliDate]
EXTERNAL NAME [prgJalaliDate].[JalaliDate]

GO
```

دنباله دارد ...

نظرات خوانندگان

نویسنده: محمد عادل
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱:۴۳

در این حالت ، چطور میتونیم در EF Code First از این DataType استفاده کنیم ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۱ ۸:۳۴

از برنامه Entity Framework Power Tools برای مهندسی معکوس ساختار موجود استفاده کنید و بعد مشاهده کنید که چه کدی رو تولید می‌کنه. کار شما در این حالت code first نیست.

نویسنده: حامد حسین نژاد
تاریخ: ۱۳۹۲/۰۲/۱۱ ۹:۲۰

البته باید این را هم در نظر گرفت که کلا CLR Integration باعث کندی دیتابیس می‌شود و انواع داده CLR بسیار کندتر از بقیه اجرا می‌شوند. این موضوع ممکن است که در دیتابیس‌های کوچک چندان مهم بنظر نیاید ولی در دیتابیس‌های بزرگ باعث بروز مشکل خواهد شد. علاوه بر این استفاده از انواع داده CLR و یا توابع CLR در دیتابیس باعث می‌شود که امکان پارتیشن بندی جداول آن دیتابیس وجود نداشته باشد.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۱:۳۴

مقاله‌ای هست اینجا در مورد کارآیی CLR در SQL Server. به نظر می‌رسه سریعتر است حدود 11 درصد نسبت به T-SQL معمولی. برای پارتشین بندی می‌تونید اینکار رو انجام بدید فقط این نوع خاص قابل انتخاب نیست. مابقی فیلدها رو می‌تونید انتخاب کنید.

نویسنده: حامد حسین نژاد
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۸:۳۱

البته کارایی CLR، بسته به مورد استفاده، متفاوته (این لینک). در مواردی مثل همین مثال، اگه تعداد سطور جدول زیاد باشه، کارایی رو به شدت کاهش میده. مخصوصا اگه بخواین از Data Warehousing هم استفاده کنید.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۹:۲۸

این به شدت رو می‌تونید در موردش عدد و رقم ارائه بدید؟ در همون لینکی که دادید عنوان شده پیاده سازی RegEx روی سطور بالا خیلی سریع‌تر هست با CLR. در هر حال بهتره با عدد و رقم و محاسبات بحث کرد.

نویسنده: حامد حسین نژاد
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۴:۲۸

مثلا فرض کنید جدولی دارید که اطلاعات Taskهای یک شرکت رو نگه میداره که شامل تاریخ شروع و تاریخ پایان هر Task هم هست. اگه این جدول یک میلیون سطر داده داشته باشه و بخوایم Task هایی که مدت زمان انجام اونها کمتر از 5 روز بوده رو انتخاب کنیم تفاوت فاحشی با datetime خواهیم داشت.
البته، همونطور که گفتم بسته به نوع استفاده داره و ممکنه از دیتابیسی به دیتابیس دیگه فرق بکنه.

عنوان: **طراحی پایگاه داده چند زبانه**
نویسنده: محمد پهلوان
تاریخ: ۱۳۹۲/۰۸/۰۹ ۱۲:۰
آدرس: www.dotnettips.info
گروه‌ها: MySQL, database design, SQL

در این مقاله قصد داریم اطلاعات مفیدی را در مورد طراحی دیتابیس‌های چند زبانه، در اختیار شما بگذاریم. مدتی قبل به طراحی دیتابیسی که چند زبانه بودن توضیحات کالا را برای مشتریانی از کشورهای مختلف پشتیبانی می‌کرد، نیاز داشتم . وقتی شروع به پیاده سازی طرح دیتابیس کردم، جواب سرراست نبود. زمانیکه در وب برای بهترین راه جستجو می‌کردم، با نظرات و روش‌های زیادی مواجه شدم. در ادامه بعضی از روش‌های محبوب را بیان می‌کنم.

ستون اضافی : این ساده‌ترین راه است و به ازای هر ستونی که نیاز به ترجمه داشته باشد، ستون اضافی در نظر می‌گیریم.

```
CREATE TABLE app_product (
     Id Int IDENTITY NOT NULL,
     Description_en Text,
     Description_pl Text,
     PRIMARY KEY (Id)
);
```

مزایا :
سادگی

کوئری‌های آسان (بدون نیاز به join)

معایب :
اضافه کردن زبان جدید نیاز به تغییر جداولی که چند زبانه هستند دارد

اگر وارد کردن داده برای همه زبان‌ها الزامی نباشد (بعضی جاها فقط زبان پیش فرض الزامی است) ممکن است داده‌های زیاد و یا فیلدهای خالی در دیتابیس ایجاد شود

نگهداری آن مشکل است

یک جدول ترجمه : این روش تمیزترین راه از دیدگاه ساختار دیتابیس به نظر می‌رسد. شما همه متن هایی را که نیاز به ترجمه دارد، در یک جدول ذخیره می‌کنید.

```
CREATE TABLE ref_language (
         Code Char(2)NOT NULL,
         Name Varchar(20) NOT NULL,
         PRIMARY KEY (Code)
);

CREATE TABLE app_translation (
        Id Int IDENTITY NOT NULL,
        PRIMARY KEY (Id)
);

CREATE TABLE app_translation_entry (
       TranslationId Int NOT NULL,
       LanguageCode Char(2) NOT NULL,
       Text Text NOT NULL,
       FOREIGN KEY (TranslationId) REFERENCES app_translation(Id),
       FOREIGN KEY (LanguageCode) REFERENCES ref_language(Code)
);

CREATE TABLE app_product (
      Id Int IDENTITY NOT NULL,
      Description Int NOT NULL,
      PRIMARY KEY (Id),
      FOREIGN KEY (Description) REFERENCES app_translation(Id)
);
```

مزایا :
اضافه کردن زبان جدید به تغییر طرح دیتابیس نیاز ندارد

به نظر تمیز است و رویکرد رابطه‌ای دارد

همه ترجمه‌ها در یک مکان است (بعضی‌ها می‌گویند این جز معایب است چون امکان خوانایی و نگهداری کمتر است)

معایب :
کوئری‌های پیچیده (به join های چندگانه نیاز دارد تا شرح کالا را به درستی نمایش دهد)

پیچیدگی زیاد
جدول ترجمه اضافی به ازای هر جدول چند زبانه : برای هر جدولی که نیاز به ترجمه دارد یک جدول اضافی ساخته می‌شود. جدول اصلی داده‌های غیر مرتبط به زبان و جدول دوم همه اطلاعات ترجمه شده را ذخیره می‌کند.

```
CREATE TABLE ref_language (
      Code Char(2)NOT NULL,
      Name Varchar(20) NOT NULL,
      PRIMARY KEY (Code)
);

CREATE TABLE app_product (
      Id Int IDENTITY NOT NULL,
      PRIMARY KEY (Id)
);

CREATE TABLE app_product_translation (
      ProductId Int NOT NULL,
      LanguageCode Char(2) NOT NULL,
      Description Text NOT NULL,
      FOREIGN KEY (ProductId) REFERENCES app_product(Id),
      FOREIGN KEY (LanguageCode) REFERENCES ref_language(Code)
);
```

مزایا :
اضافه کردن زبان جدید به تغییر طرح دیتابیس نیاز ندارد

کوئری‌های نسبتا ساده است

معایب :
ممکن است تعداد جداول دو برابر شود

سه مثال نشان داده شده در بالا به ما ایده می‌دهند که چگونه روش‌های مختلف ممکن است استفاده شوند. البته اینها همه گزینه‌های ممکن نیستند، فقط محبوبترین روش‌ها هستند و شما می‌توانید آنها را ویرایش کنید؛ به عنوان مثال با تعریف View های اضافی که join های پیچیده شما را در کدها، ذخیره می‌کنند. راه حلی که شما انتخاب می‌کنید به نیازمندی‌های پروژه وابسته است. اگر شما به سادگی نیاز دارید و مطمئن هستید تعداد زبان‌های پیشتیبانی کم و ثابت است، می‌توانید راه حل اول را انتخاب کنید. اگر به انعطاف پذیری بیشتری نیاز دارید، می‌توانید join های ساده در کوئری‌های چند زبانه را انتخاب کنید. راه حل سوم انتخاب مناسبی است.

منبع :

http://fczaja.blogspot.com/2010/08/multilanguage-database-design.html

نظرات خوانندگان

نویسنده: ناصر فرجی
تاریخ: ۱۳۹۲/۰۸/۰۹ ۱۶:۱۴

روشی که من خودم مدتی استفاده میکردم اضافه کردن یک فیلد language به هر تیبل بود. موقع ثبت دیتا هر زبانی بود همون زبان رو تو این فیلد نگه میداشتم. مثلا برای فارسی fa و انگلیسی en و ... , موقع نمایش هم بر اساس زبان سایت یک کوئری ساده گرفته می‌شد و اطلاعات زبان مورد نظر لود می‌شد.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۲/۰۸/۰۹ ۱۹:۲۹

با سلام و تشکر از مطلب خوبتون
طراحی با یک جدول زبان و نگه داشتن کلید خارجی در جداول مربوطه بهتر میشه
چندید ساله که از این طراحی استفاده میکنیم و جواب داده.
یکی از مزایایی که داره میتونی مدیریت سامانه را نسبت به هر زبان بطور مستقل انجام بدی
و هرجا که نیاز داشتی همزمان چند رکورد را درج کنید.
ساده و روان.
البته استراتژی سیستم استفاده از الگوی مناسب رو توجیه میکنه.

نویسنده: محمد پهلوان
تاریخ: ۱۳۹۲/۰۸/۱۰ ۱۰:۵۰

استفاده از یک فیلد language در هر تیبل باعث می‌شود شما برای یک موجودیت کالا مثلا در سه زبان مختلف 3 رکورد درج کنید که در ارتباط این کالا با دیگر جداول دچار مشکل خواهید شد

نویسنده: محمد پهلوان
تاریخ: ۱۳۹۲/۰۸/۱۰ ۱۱:۳۶

روش دوم واقعا روش تمیز و جمع و جوریه اما کوئری هاش واقعا پیچیده اس و انعطاف نداره. به نظر من مخصوصا برای استفاده از EF روش سوم روش مناسب‌تری باشه. این کوئری‌ها رو با توجه به حجم داده زیاد و سایت پرترافیک بررسی کنید.
خودم بین روش دو و سه مرددم. از دوستان میخوام نظراتشون را با دلائل بیان کنن تا به نتیجه خوبی برسیم

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۱۰ ۱۱:۵۱

مطلب Globalization در ASP.NET MVC - قسمت ششم در همین راستا مفید است.

نویسنده: بختیاری
تاریخ: ۱۳۹۲/۰۸/۱۰ ۱۵:۱۶

سلام
من روش آقای موسوی را منطقی می‌دانم خودم هم با این روش یک سایت طراحی کردم که الان درست و بدون مشکل کار می‌کند

عنوان: آموزش MDX Query - قسمت اول
نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۰۹/۱۹ ۲۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: SQL, OLAP, MDX, SSAS

در طول این سری آموزش‌های MDX (البته هنوز نمی‌دانم چند قسمت خواهد بود) تلاش خواهم کرد تمامی موارد موجود در MDXها را به طور کامل با شرح و توضیح مناسب پوشش دهم.

امیدوارم شما دوستان عزیز پس از مطالعه‌ی این مجموعه مقالات به دانش کافی در خصوص MDX Queryها دست پیدا کنید.

در قسمت اول این آموزش‌ها در نظر دارم در ابتدا مفاهیم اولیه OLAP و همچنین مفاهیم مورد نیاز در Multi Dimentional Data Base ها برای شما عزیزان توضیح دهم و در قسمت‌های بعدی این مجموعه در خصوص MDX Queryها صحبت خواهم کرد.

انباره داده (Data Warehouse)

عملا یک یا چند پایگاه داده می‌باشد که اطلاعات تجمیع شده از دیگر پایگاه‌های داده را درخود نگه داری می‌کند. برای ارایه گزارشاتی که از پایگاه داده‌های OLTP نمی‌توانیم به راحتی بگیریم.

OLTP (Online transaction processing)

سیستم پردازش تراکنش برخط می‌باشند . که عملا همان سیستم هایی می‌باشند که در طول روز دارای تغییرات بسیار زیادی می‌باشند (مانند سیستم‌های حسابداری، انبار داری و ... که در طول روز دایما دارای تغییرات در سطح داده می‌باشند.)

OLAP (OnLine Analysis Processing)

این سیستم‌ها خدماتی در نقش تحلیل‌گر داده و تصمیم گیرنده ارائه می‌کند. چنین سیستمهایی می‌توانند، داده را در قالبهای مختلف برای هماهنگ کردن نیازهای مختلف کاربران مختلف، سازماندهی کنند.

تفاوت انبار داده (Data Warehouse) و پایگاه داده(Data Base)

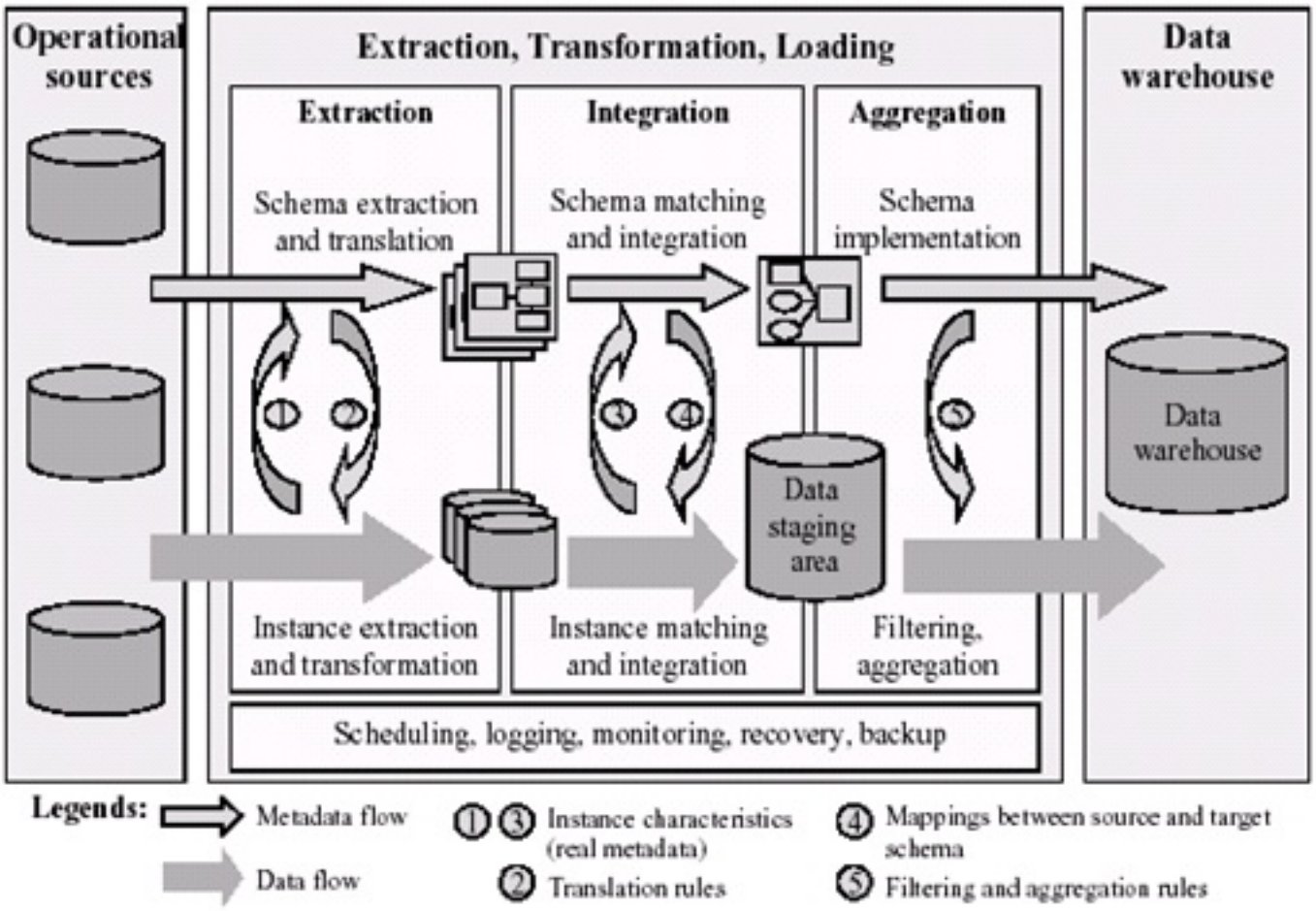
وظیفه اصلی سیستم‌های پایگاه‌داده کاربردی OnLine ،پشتیبانی از تراکنش‌های برخط و پردازش کوئری است. این سیستم‌ها، سیستم پردازش تراکنش برخط(OLTP) نامیده می‌شوند و بیشتر عملیات روزمره یک سازمان را پوشش می‌دهند. از سوی دیگر انبارداده، خدماتی در نقش تحلیل‌گر داده و تصمیم گیرنده ارائه می‌کند. چنین سیستمهایی می‌توانند داده را در قالبهای مختلف برای هماهنگ کردن نیازهای مختلف کاربران مختلف، سازماندهی و ارائه می‌کند. این سیستم‌ها با نام سیستم‌های پردازش تحلیلی برخط (OLAP) شناخته‌می‌شوند.

موارد تفاوت انبار داده (Data Warehouse) و پایگاه داده(Data Base)

- **از لحاظ مدل‌های داده:** پایگاه‌های داده برای مدل OLTP بهینه سازی شده‌است. که بر اساس مدل داده رابطه‌ای امکان پردازش تعداد زیادی تراکنش همروند، که اغلب حاوی رکوردهای اندکی هستند را دارد. اما در انبارهای داده که برای پردازش تحلیلی بر خط، طراحی شده‌اند امکان پردازش تعداد کمی کوئری پیچیده بر روی تعداد بسیار زیادی رکورد داده فراهم می‌شود. سرورهای OLAP می‌توانند از دو نوع رابطه‌ای (ROLAP) یا چندبعدی باشند (MOLAP).
- **از لحاظ کاربران:** کاربران پایگاه‌داده کارمندان دفتری و مسؤولان هستند در حالی‌که کاربران انبارداده مدیران و تصمیم‌گیرنده‌ها هستند.
- **از لحاظ عملیات قابل اجرا بر روی آن‌ها:** عملیات انجام شده برروی پایگاه‌های داده عمدتا عملیات (Select/Insert/Update/Delete) می‌باشد ، در حالی که عملیات روی انبار داده عمدتا Select ها می‌باشند.

• **از لحاظ مقدار داده‌ها:** مقدار داده‌های یک پایگاه‌داده در حدود چند مگابایت تا چند گیگابایت است در حالی که این مقدار در انبار داده در حدود چند گیگابایت تا چند ترابایت است.
• **از لحاظ زمان پرس و جو :** به طور کلی سرعت پرس و جو ها روی انباره‌ی داده بسیار بالاتر از کوئری مشابه آن روی پایگاه داده می‌باشد.

مراحل ساخت یک انباره‌ی داده (Data WareHouse) به شرح زیر می‌باشد



• **پاکسازی داده (Data Cleansing)**

پاکسازی داده‌ها عبارت است از شناسایی و حذف خطاها و ناسازگاریهای داده ای به منظور دستیابی به داده‌هایی با کیفیت بالاتر.

اگر داده‌ها از منابع یکسان مثل فایل‌ها یا پایگاه‌های داده ای گرفته شوند خطاهایی از قبیل اشتباهات تایپی، داده‌های نادرست و فیلدهای بدون مقدار را خواهیم داشت و چنانچه داده‌ها از منابع مختلف مثل پایگاه داده‌های مختلف یا سیستم اطلاعاتی مبتنی بر وب گرفته شوند .با توجه به نمایش‌های دادهای مختلف خطاها بیشتر بوده و پاکسازی داده‌ها اهمیت بیشتری پیدا خواهد کرد.
برای دستیابی به دادههای دقیق و سازگار، بایستی داده‌ها را یکپارچه نموده و تکرارهای آنها را حذف نمود.

وجود خطاهای نویزی، ناسازگاری در داده‌های انبار داده و ناقص بودن داده‌ها امری طبیعی است. فیلدهای یک جدول ممکن است خالی باشند و یا دارای داده‌های خطا دار و ناسازگار باشند. برای هر کدام از این حالت‌ها روشهایی جهت پاکسازی و اصلاح داده‌ها ارایه می‌شود.

در این بخش عملیات مختلفی برای پاکسازی داده‌ها قابل انجام است:

• نادیده گرفتن تاپل‌های نادرست

• پرکردن فیلدهای نادرست به صورت دستی

• پرکردن فیلدهای نادرست با یک مقدار مشخص

• پرکردن فیلدها با توجه به نوع فیلد و داده‌ها ی موجود

• پرکردن فیلدها با نزدیکترین مقدار ممکن (مثلا میانگین فیلد تاپل‌های دیگر می‌تواند به عنوان یک مقدار مناسب در نظر گرفته شود) • **یکپارچه‌سازی (Integration)**

این فاز شامل ترکیب داده‌های دریافتی از منابع اطلاعاتی مختلف، استفاده از متاداده‌ها برای شناسایی و حذف افزونگی داده ها، تشخیص و رفع برخوردهای داده ای می‌باشد.

یکپارچه سازی داده‌ها از سه فاز کلی تشکیل شده است:

• **شناسایی فیلدهای یکسان:** فیلدهای یکسان که در جدول‌ها ی مختلف دارای نامهای مختلف میباشند.

• **شناسایی افزونگی‌ها ی موجود در داده‌ها ی ورودی:** داده‌های ورودی گاهی دارای افزونگی است. مثلا بخشی از رکورد در جداول مختلف وجود دارد.

• **مشخص کردن برخوردهای داده ای:** مثالی از برخوردهای داده ای یکسان نبودن واحدهای نمایش داده ای است. مثلا فیلد وزن در یک جدول بر حسب کیلوگرم و در جدولی دیگر بر حسب گرم ذخیره شده است.

• **تبدیل داده‌ها(Data Transformation)**

در این فاز، داده‌های ورودی طی مراحل زیر به شکلی که مناسب عمل داده کاوی باشند، در می‌آیند:

• از بین بردن نویز داده¬ها(Smoothing)

• تجمیع داده¬ها(Aggregation)

• کلی¬سازی(Generalization)

• نرمال¬سازی(Normalization)

• افزودن فیلدهای جدید

در ادامه به شرح هر یک می‌پردازیم: **1. از بین بردن نویزهای داده ای(Smoothing):** منظور از داده‌های نویزی، داده هایی هستند که در خارج از بازه مورد نظر قرار می‌گیرند. مثلا اگر بازه حقوقی کارمندان بین یک صد هزار تومان و یک میلیون تومان باشد، داده‌های خارج از این بازه به عنوان داده‌های نویزی شناخته شده و در این مرحله اصلاح می‌گردند. برای اصلاح داده‌های نویزی از روشهای زیر استفاده می‌شود:

• استفاده از مقادیر مجاور برای تعیین یک مقدار مناسب برای فیلدهای دارای نویز

• دسته بندی داده‌های موجود و مقداردهی فیلد دارای داده نویزی با استفاده از دسته نزدیکتر

• ترکیب روشهای فوق با ملاحظات انسانی، در این روش، اصلاح مقادیر نویزی با استفاده از یکی از روشهای فوق انجام می‌گیرد اما افرادی برای بررسی و اصلاح نیز وجود دارند

2. تجمیع داده ها(Aggregation): تجمیع داده‌ها به معنی بدست آوردن اطلاعات جدید از ترکیب داده‌های موجود می‌باشد. به عنوان مثال بدست فروش ماهانه از حساب فروش‌های روزانه. **3. کلی سازی(Generalization):** کلی سازی به معنی دسته بندی داده‌های موجود براساس ماهیت و نوع آنها است. به عنوان مثال می‌توان اطلاع رده‌های سنی خاص (جوان، بزرگسال، سالخورده) را از فیلد سن استخراج کرد.

4. نرمال سازی(Normalization): منظور از نرمال سازی، تغییر مقیاس داده‌ها است. به عنوان مثالی از نرمال سازی، می‌توان به تغییر بازه یک فیلد از مقادیر موجود به بازه 0 تا 1 اشاره کرد.

5. افزودن فیلدهای جدید: گاهی اوقات برای سهولت عمل داده کاوی می‌توان فیلدهایی به مجموعه فیلدهای موجود اضافه کرد. مثلا می‌توان فیلد میانگین حقوق کارمندان یک شعبه را به مجموعه فیلدهای موجود اضافه نمود.

• کاهش داده‌ها(Reduction)

در این مرحله، عملیات کاهش داده‌ها انجام می‌گیرد که شامل تکنیکهایی برای نمایش کمینه اطلاعات موجود است

. این فاز از سه بخش تشکیل می‌شود:

• کاهش دامنه و بعد: فیلدهای نامربوط، نامناسب و تکراری حذف می‌شوند. برای تشخیص فیلدهای اضافی، روشهای آماری و تجربی وجود دارند ؛ یعنی با اعمال الگوریتمهای آماری و یا تجربی بر روی داده‌های موجود در یک بازه زمانی مشخص، به این نتیجه می‌رسیم که فیلد یا فیلدهای خاصی کاربردی در انباره داده ای و داده کاوی نداشته و آنها را حذف می‌کنیم.

• فشرده سازی داده ها: از تکنیکهای فشرده سازی برای کاهش اندازه داده‌ها استفاده می‌شود.

• کدکردن داده ها: داده‌ها در صورت امکان با پارامترها و اطلاعات کوچکتر جایگزین می‌شوند.

مدل داده‌ای رابطه‌ای (Relational) وچند بعدی (Multidimensional) :

1. مدل داده رابطه‌ای (Relational data modeling) بر اساس دو مفهوم اساسی موجودیت (entity) و رابطه (relation) بنا نهاده شده است. از این رو آن را با نام مدل ER نیز می‌شناسند.

• موجودیت (entity) : نمایانگر همه چیزهایی که در پایگاه داده وجود خارجی دارند یا به تصور در می‌آیند. پدیده‌ها دارای مشخصاتی هستندکه به آن‌ها صفت (attribute) گفته می‌شود.

• رابطه (relation) : پدیده‌ها را به هم می‌پیوندد و چگونگی در ارتباط قرار گرفتن آن‌ها با یکدیگر را مشخص می‌کند.

2. مدل داده چندبعدی (Multidimensional modeling) یا MD بر پایه دو ساختار جدولی اصلی بنا نهاده شده است:

• جدول حقایق (Fact Table)

• جداول ابعاد (Dimension Table)

این ساختار امکان داشتن یک نگرش مدیریتی و تصمیم‌گیری به داده‌های موجود در پایگاه داده را تسهیل می‌کند.

جدول حقایق : قلب حجم داده‌ای ما را تشکیل می‌دهد و شامل دو سری فیلد است : کلیدهای خارجی به ابعاد و شاخص‌ها (Measure).

شاخص‌ها (Measure) : معیارهایی هستند که بر روی آن‌ها تحلیل انجام می‌گیرد و درون جدول حقایق قرار دارند. شاخص‌ها قبل از شکل‌گیری انبار داده توسط مدیران و تحلیل‌گران به دقت مشخص می‌شوند. چون در مرحله کار با انبار اطلاعات اساسی هر تحلیل بر اساس همین شاخص‌ها شکل می‌گیرد. شاخص‌ها تقریباً همیشه مقادیر عددی را شامل می‌شوند. مثلا برای یک فروشگاه

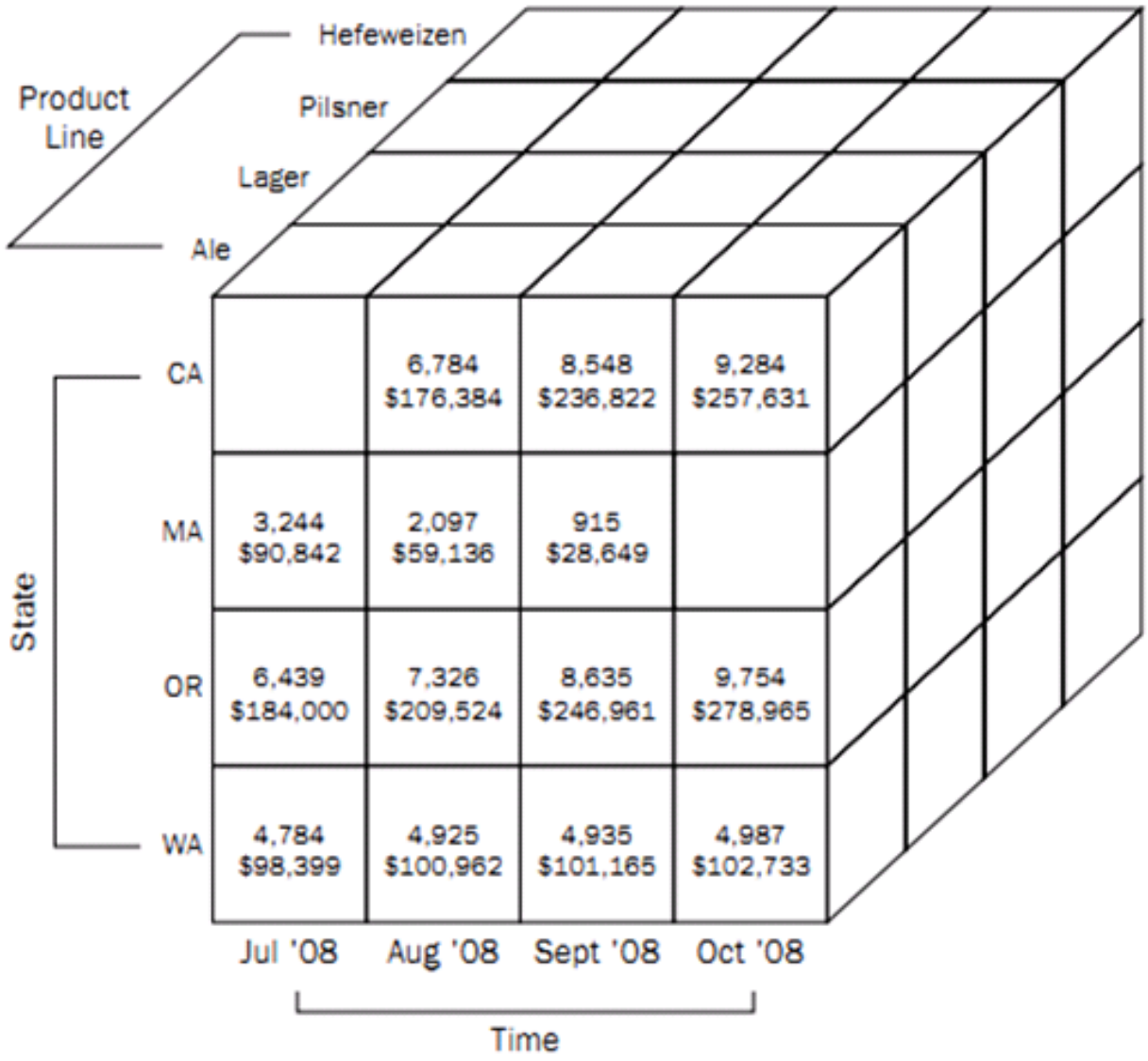
زنجیره‌ای این شاخص‌ها می‌توانند واحدهای فروخته‌شده کالاها و مبلغ فروش به تومان باشند.

بعد (Dimension) : هر موجودیت در این مدل می‌تواند با یک بعد تعریف شود. ولی بعدها با موجودیت‌های مدل ER متفاوتند زیرا آن‌ها سازمان شاخص‌ها را تعیین می‌کنند. علاوه بر این دارای یک ساختار سلسله مراتبی هستند و به طور کلی برای حمایت از سیستم‌های تصمیم گیری سازمان‌دهی شده‌اند.

اجزای بعدها member نام دارند و تقریباً همه بعدها، memberهای خود را در یک یا چند سطح سلسله مراتبی (hierarchies) سازمان‌دهی می‌نمایند، که این سلسله مراتب نمایانگر مسیر تجمیع (integration) و ارتباط بین سطوح پایین‌تر (مثل روز) و سطوح بالاتر (مثل ماه و سال) است. وقتی یک دسته از memberهای خاص با هم مفهوم جدیدی را ایجاد می‌کنند، به آنها یک سطح (Level) می‌گوییم. (مثلاً هر سی روز را ماه می‌گوییم. در این حالت ماه یک سطح است.)

حجم‌های داده‌ای (Data Cube)

حجم‌های داده‌ای یا Cube از ارتباط تعدادی بعد با تعدادی شاخص تعریف می‌شود. ترکیب memberهای هر بعد از حجم داده‌ای فضای منطقی را تعریف می‌کند که در آن مقادیر شاخص‌ها ظاهر می‌شوند. هر بخش مجزا که شامل یکی از memberهای بعد در حجم داده‌ای است ، سلول (cell) نامیده‌می‌شود. سلول‌ها شاخص‌های مربوط به تجمیع‌های مختلف را در خود نگهداری می‌نمایند. در واقع مقادیر مربوط به حقایق (Fact) که در جدول حقایق (Fact) تعریف می‌شوند در حجم داده‌ای (Data Cube) در سلول‌ها (Cell) نمایان می‌گردند.



شماهای داده‌ای (Data Schema) : سه نوع Schema در طراحی Data Warehouse وجود دارد

1. Stare

2. Snowflake

3. Galaxy

1. شمای ستاره‌ای (Star Schema) : متداولترین شما، همین شمای‌ستاره‌ای است. که در آن انبارداده با استفاده از اجزای زیر تعریف می‌شود:

• یک جدول مرکزی بزرگ به نام جدول حقایق که شامل حجم زیادی از داده‌های بدون تکرار است.

• مجموعه‌ای از جدول‌های کمکی کوچک‌تر به نام جدول بعد ، که به ازای هر بعد یکی از این جداول موجود خواهد بود.

• شکل این شما به صورت یک ستاره است که جدول حقایق در مرکز آن قرار گرفته و هر یک از جداول بعد به وسیله شعاع‌هایی به آن مربوط هستند.

مشکل این مدل احتمال پیشامد افزونگی در آن است.

2. شمای دانه‌برفی (Snowflake Schema) : در واقع شمای دانه‌برفی، نوعی از شمای ستاره‌ای است که در آن بعضی از جداول بعد نرمال شده‌اند. و به همین خاطر دارای تقسیمات بیشتری به شکل جداول اضافی می‌باشد که از جداول بعد جدا شده‌اند.

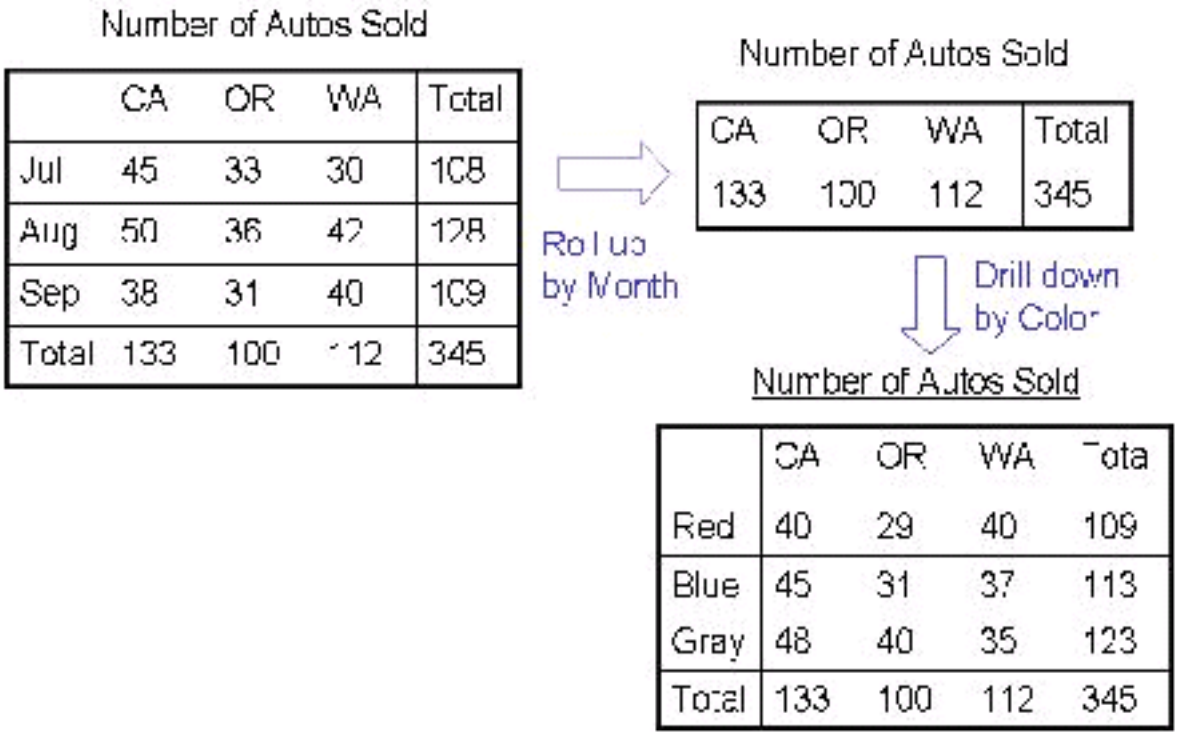
تفاوت این دو شما در این است که جداول شمای دانه برف نرمال هستند و افزونگی در آن‌ها کاهش یافته است. که این برای کار کردن با داده‌ها و از لحاظ فضای ذخیره‌سازی مفید است. ولی در عوض کارایی را پایین می‌آورد، زیرا در محاسبه کوئری‌ها به joinهای بیشتری نیاز داریم.

3. شمای کهکشانی (galaxy schema) : در کاربردهای پیچیده برای به اشتراک گذاشتن ابعاد نیاز به جداول حقایق چندگانه احساس می‌شود که یک یا چند جدول بعد را در بین خود به اشتراک می‌گذارند. این نوع شما به صورت مجموعه‌ای از شماهای ستاره‌ای است و به همین دلیل شمای کهکشان یا شمای منظومه‌ای نامیده‌می‌شود. این شما به ما این امکان را می‌دهد که جداول بعد بین جداول حقایق مختلف به اشتراک گذاشته شوند.

عملیات بر روی حجم‌های داده‌ای :

• **Roll Up (یا Drill-up) :** با بالا رفتن در ساختار سلسله مراتبی مفهومی یک حجم داده‌ای، یا با کاهش دادن بعد، یک مجموعه با جزئیات کمتر (خلاصه شده) ایجاد می‌نماید. بالا رفتن در ساختار سلسله مراتبی به معنای حذف قسمتی از جزئیات است. برای مثال اگر قبلاً بعد مکان بر حسب شهر بوده آن را با بالا رفتن در ساختار سلسله مراتبی بر حسب کشور درمی‌آوریم. ولی وقتی با کاهش دادن بعد سروکار داریم منظور حذف یکی از ابعاد و جایگزین کردن مقادیر کل است. در واقع همان عمل تجمیع (aggregation) است.

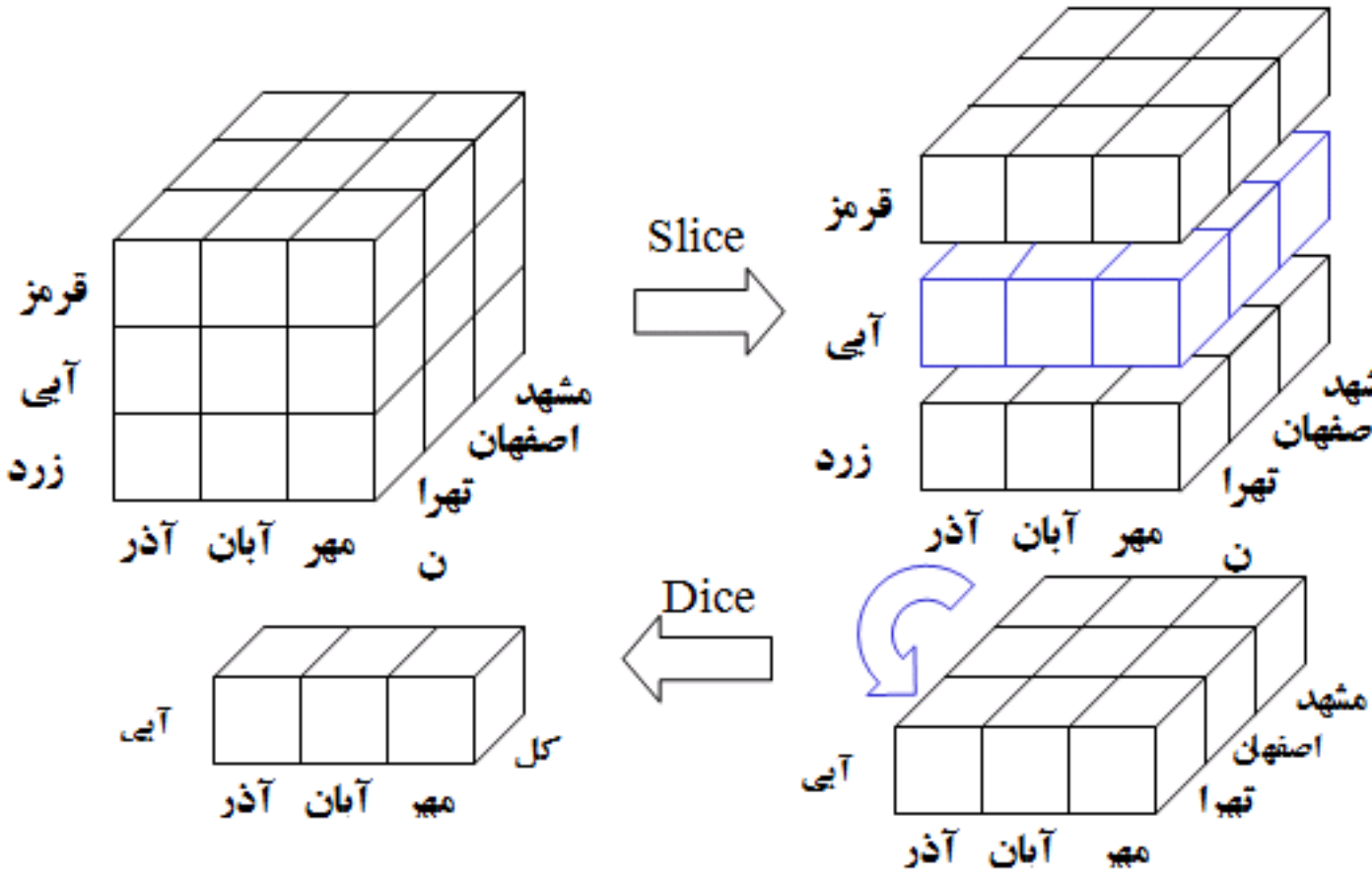
• **Drill Down :** بر عکس عملRoll-up است و از موقعیتی با جزئیات داده‌ای کم به جزئیات زیاد می‌رود. این کار با پایین آمدن در ساختار سلسله مراتبی(به سمت جزئیات بیشتر) یا با ایجاد ابعاد اضافی انجام می‌گیرد.



نمونه‌ای از عملیات Drill Down و Roll Up

• **Slice :** با انتخاب و اعمال شرط بر روی یکی از ابعاد یک subcube به شکل یک برش دو بعدی ایجاد می‌کند. در واقع همان عمل انتخاب (select) است.

• **Dice :** با انتخاب قسمتی از ساختار سلسله مراتبی بر روی دو یا چند بعد یک subcube ایجاد می‌نماید.



نمونه‌ای از عملیات Dice و Slice

• **Pivot (یا Rotate) :** این عملیات بردارهای بعد را در ظاهر می‌چرخاند.

Region	Sales variance
Africa	105%
Asia	57%
Europe	122%
North America	97%
Pacific	85%
South America	163%

Nation	Sales variance
China	123%
Japan	52%
India	87%
Singapore	95%

نمونه‌ای از عملیات pivot

• **Drill-across :** نتیجه اجرای کوئری‌هایی که نتیجه اجرای آنها حجم‌های داده‌ایهای مرکب با بیش از یک fact-table است.

• **Ranking :** سلول‌هایی را باز می‌گرداند که در بالا یا پایین شرط خاصی واقع هستند. مثلاً ده محصولی که بهترین فروش را داشته‌اند.

سرورهای OLAP :

در تکنولوژیOALP داده‌ها به دو صورت چندبعدی (MOLAP) (Multidimensional OLAP) و رابطه‌ای (ROLAP) (Relational OLAP) ذخیره می‌شوند. OLAP پیوندی(HOLAP) تکنولوژیی است که دو نوع قبل را با هم ترکیب می‌کند.

MOLAP : روشی است که معمولاً برای تحلیل‌های OLAP در تجارت مورد استفاده قرار می‌گیرد. در MOLAP، داده‌ها با ساختار یک حجم داده‌ای (Data Cube) چند بعدی ذخیره می‌شوند. ذخیره‌سازی در پایگاه‌داده‌های رابطه‌ای انجام نمی‌گیرد، بلکه با یک فرمت خاص انجام می‌شود. اغلب محصولات موفق MOLAP از یک روش چندبعدی استفاده می‌نمایند که در آن یک سری حجم‌های داده‌ای کوچک، انبوه و از پیش محاسبه‌شده، یک حجم داده‌ای بزرگ (hypercube) را می‌سازند.

علاوه براین MOLAP به شما امکان می‌دهد داده‌های دیدهای (View) تحلیل‌گران را دسته بندی کنید، که این در حذف اشتباهات و برخورد با ترجمه‌های پرغلط کمک بزرگی است.

گذشته از همه این‌ها از آن‌جا که داده‌ها به طور فیزیکی در حجم‌های داده‌ای بزرگ چندبعدی ذخیره می‌شوند، سرعت انجام فعالیت‌ها بسیار زیاد خواهد بود.

از آنجا که یک کپی از داده‌های منبع در کامپیوتر Analysis server ذخیره‌می‌شود، کوئری‌ها می‌توانند بدون مراجعه به منابع مجدداً

محاسبه شوند. کامپیوتر Analysis server ممکن است کامپیوترسرور که تقسیم بندی‌ها در آن انجام شده یا کامپیوتر دیگری باشد. این امر بستگی به این دارد که تقسیم‌بندی‌ها در کجا تعریف شده‌اند. حتی اگر پاسخ کوئری‌ها از روی تقسیمات تجمیع (integration) شده قابل دستیابی نباشند، MOLAP سریع‌ترین پاسخ را فراهم می‌کند. سرعت انجام این کار به طراحی و درصد تجمیع تقسیم‌بندی‌ها بستگی دارد.

مزایا : کارایی عالی- حجم‌های داده‌ای MOLAP برای بازیابی سریع داده‌ها ساخته شده‌اند و در فعالیت‌های slice و dice به صورت بهینه پاسخ می‌دهند. ترکیب سادگی و سرعت مزیت اصلی MOLAP است.

در ضمنMOLAP قابلیت محاسبه محاسبات پیچیده را فراهم می‌کند. همه محاسبات از پیش وقتی که حجم‌های داده‌ای ساخته می‌شود، ایجاد می‌شوند. بنابراین نه تنها محاسبات پیچیده انجام شدنی هستند بلکه بسیار سریع هم پاسخ می‌دهند.

معایب : عیب این روش این است که تنها برای داده‌هایی با مقدار محدود کارکرد خوبی دارد. از آنجا که همه محاسبات زمانی که حجم‌های داده‌ای ساخته می‌شود، محاسبه می‌گردند، امکان این که حجم‌های داده‌ای مقدار زیادی از داده‌ها را در خود جای دهد، وجود ندارد. ولی این به این معنا نیست که داده‌های حجم‌های داده‌ای نمی‌توانند از مقدار زیادی داده مشتق شده باشند. داده‌ها می‌توانند از مقدار زیادی داده مشتق شده‌باشند. اما در این صورت، فقط اطلاعات level خلاصه (level ای که دارای کمترین جزئیات است یعنی سطوح بالاتر) می‌توانند در حجم‌های داده‌ای موجود باشند.

ROLAP : محدودیت MOLAP در حجم داده‌های قابل پرس‌وجو و نیاز به روشی که از داده‌های ذخیره‌شده به روش رابطه‌ای حمایت کند، موجب پیشرفت ROLAP شد.

مبنای این روش کارکردن با داده‌هایی که در پایگاه‌داده‌های رابطه‌ای ذخیره‌شده‌اند، برای انجام اعمال slicing و dicing معمولی است. با استفاده از این مدل ذخیره‌سازی می‌توان داده‌ها را بدون ایجاد واقعی تجمیع در پایگاه‌داده‌های رابطه‌ای به هم مربوط کرد.

مزایا : با این روش می‌توان به حجم زیادی از داده‌ها را رسیدگی کرد. محدودیت حجم داده در تکنولوژی ROLAP مربوط به محدودیت حجم داده‌های قابل ذخیره‌سازی در پایگاه‌داده‌های رابطه‌ای است. به بیان دیگر، خود ROLAP هیچ محدودیتی بر روی حجم داده‌ها اعمال نمی‌کند.

معایب : ممکن است کارایی پایین بیاید. زیرا هر گزارش ROLAP در واقع یک کوِاری SQL (یا چند کوِاری SQL)در پایگاه داده‌های رابطه‌ای است و اگر حجم داده‌ها زیاد باشد ممکن است زمان پاسخ کوِاری طولانی شود. در مجموع ROLAP سنگین است، نگهداری آن سخت است و کند هم هست. بخصوص زمانی که نیاز به آدرس دهی جدول‌های ذخیره شده در سیستم چند بعدی داریم.

این محدودیت ناشی از عملکرد SQL است. زیرا تکنولوژی ROLAP بر پایه عبارات مولد SQL برای پرسش و پاسخ بر روی پایگاه داده رابطه‌ای است و عبارات SQL به همه نیازها پاسخ نمی‌دهند (مثلاً محاسبه حساب‌های پیچیده در SQL مشکل است)، بنابراین فعالیت‌های ROLAP به آن چه SQL قادر به انجام آن است محدود می‌گردد.

تفاوت ROALP و MOLAP : تفاوت اصلی این دو در معماری آن‌ها است. محصولات MOLAP داده‌های مورد نیاز را در یک حافظه نهان (cache) مخصوص می‌گذارد. ولی ROLAP تحلیل‌های خود را بدون استفاده از یک حافظه میانی انجام می‌دهد، بدون آن که از یک مرحله میانی برای گذاشتن داده‌ها در یک سرور خاص استفاده کند.

با توجه به کند بودن ROLAP در مقایسه باMOLAP ، باید توجه داشت که کاربرد این روش بیشتر در پایگاه داده‌های بسیار بزرگی است که گاه‌گاهی پرس و جویی بر روی آن‌ها شکل می‌گیرد، مثل داده‌های تاریخی و کمتر جدید سال‌های گذشته.

نکته: اگر از Analysis Services که به وسیله Microsoft OLE DB Provider مهیا شده استفاده می‌کنید، تجمیع‌ها نمی‌توانند برای تقسیم‌بندی از روش ROLAP استفاده نمایند.

HOLAP : با توجه به نیاز رو به رشدی که برای کارکردن با داده‌های بلادرنگ (real time) در بخش‌های مختلف در صنعت و تجارت

احساس می‌شود، مدیران تجاری انتظار دارند بتوانند با دامنه وسیعی از اطلاعات که فوراً و بدون حتی لحظه‌ای تأخیر در دسترس باشند، کار کنند. در حال حاضر شبکه اینترنت و سایر کاربردها یی که به داده‌هایی از منابع مختلف مراجعه دارند و نیاز به فعالیت با یک سیستم بلادرنگ هم دارند، همگی از سیستم HOLAP بهره می‌گیرند.

named set :

Named Set مجموعه‌ای از memberهای بعد یا مجموعه‌ای از عبارات است که برای استفاده مجدد ایجاد می‌شود.

Calculated member

Calculated Memberها memberهایی هستند که بر اساس داده‌ها نیستند بلکه بر اساس عبارات ارزیابی MDX هستند. آنها دقیقاً به سبک سایر memberهای معمولی هستند. MDX یک مجموعه قوی از عملیاتی را تامین میکند که میتوانند برای ساختCalculated Memberها مورد استفاده قرار گیرند به طوری که به شما امکان داشتن انعطاف زیاد در کار کردن با داده‌های چند بعدی را بدهد.

امیدوارم در این قسمت با مفاهیم نخستین OLAP آشنا شده باشید.

تلاش خواهم کرد در قسمت بعدی در خصوص نصب SQL Server Analysis Services و نصب پایگاه داده‌ی Adventure Work DW 2008 شرح کاملی را ارایه کنم.

نظرات خوانندگان

نویسنده: afshin
تاریخ: ۱۳۹۲/۰۹/۲۰ ۱۷:۲۶

خیلی ممنون ... لطفا ادامه بدید ... به بحث عملی هم میپردازید ؟ کلا بحث عملی بر پایه SQL Server هست یا از نرم افزارهای دیگه هم استفاده میشه ؟ در مورد ETS ها هم اگه میشه توضیح بدید .... باز هم تشکر

نویسنده: vahid
تاریخ: ۱۳۹۲/۰۹/۲۰ ۲۱:۱۶

ممنون از مطلب مفیدتون لطفا در مورد اینکه چگونه datawarehouse را پیاده سازی کنیم توضیح دهید ممنون

نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۰۹/۲۰ ۲۳:۸

به طور کلی زبان MDX بین DBMS یک سان می‌باشد (تقریبا) و البته من در این مجموعه تلاش دارم در خصوص MDX Queryها آنچه را بلد هستم با شما در میان بگذارم. اگر عمری بود مطالبی در خصوص UI هم اضافه خواهم کرد.

نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۰۹/۲۰ ۲۳:۹

تلاشم را می‌کنم بعد از آموزش MDXها یک مطلب هم در این خصوص برای شما بگذارم.

نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۲/۰۹/۲۱ ۱۴:۳

سلام، لطفا تعریفی از Datamart و ارتباطش با Datawarehouse ارائه بدید. تشکر

نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۰۹/۲۱ ۱۵:۴۹

اگر اجازه دهید تعاریف DataMart و ارتباطش با DW را در یک مقاله کوتاه انتشار می‌دهم . اینجا در کامنت بگذارم تعداد کمی بهش دسترسی پیدا خواهد کرد.

نویسنده: علاقه مند
تاریخ: ۱۳۹۲/۱۰/۱۰ ۱۱:۱۰

سلام،
آیا کتاب فارسی تالیفی (یا ترجمه بصورت مناسب و کامل) با موضوع انبار داده‌ها و OLAP در بازار سراغ دارید؟

نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۱۰/۱۰ ۱۶:۱۷

نه من از منبع فارسی مطالعه نکردم و اطلاعی هم در این خصوص ندارم . البته انتشارات Wrox دو تا کتاب خود در خصوص SSAS دارد منبع اطلاعاتی هم که بنده اینجا ارایه می‌کنم همین مرجع‌ها می‌باشد.

عنوان: **آموزش MDX Query - قسمت دوم - نصب و راه اندازی SSAS**
نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۰۹/۲۳ ۸:۴۰
آدرس: www.dotnettips.info
گروه‌ها: SQL, OLAP, MDX, SSAS

در این قسمت در خصوص نحوه‌ی نصب SSAS صحبت خواهم کرد .

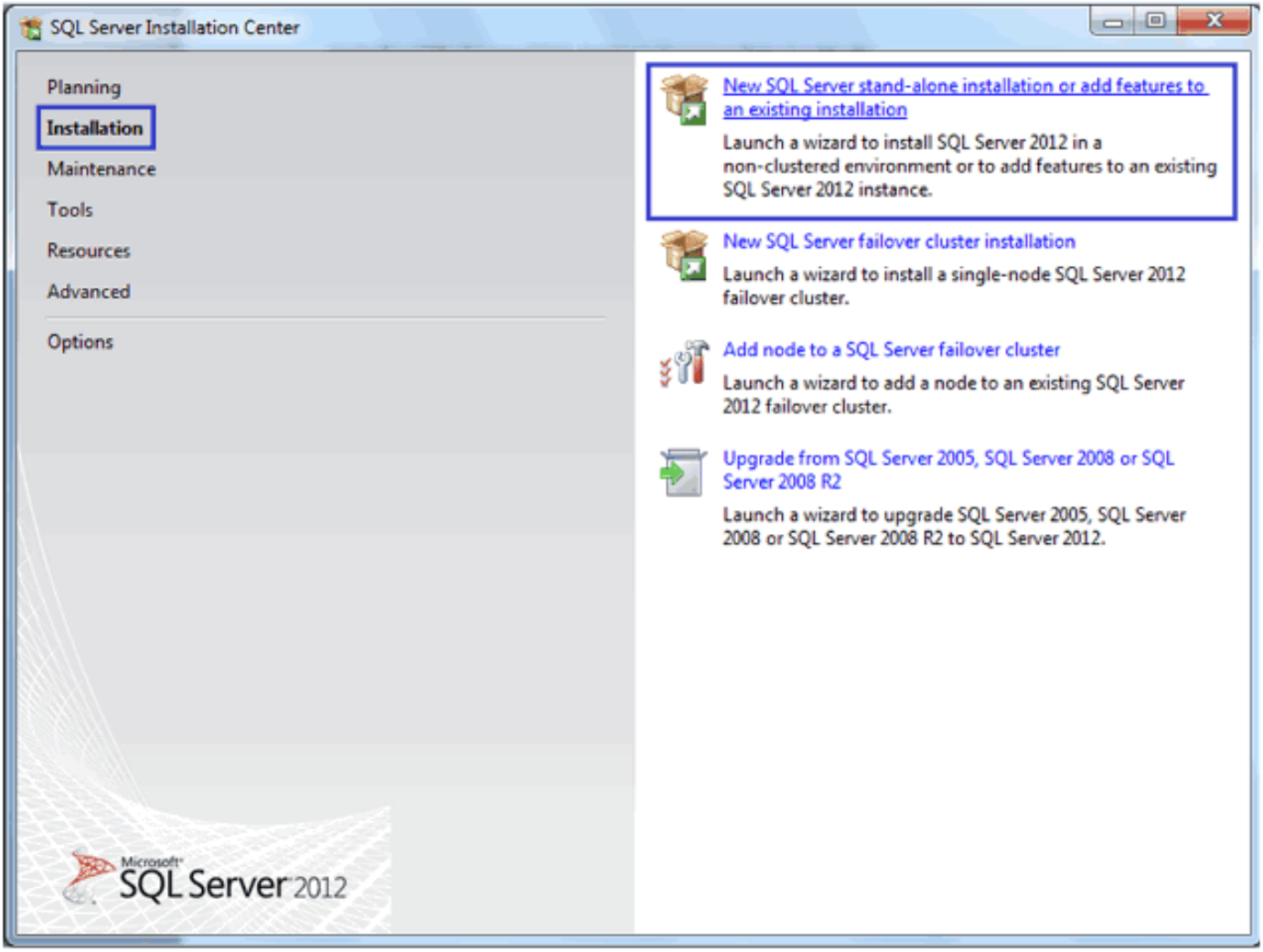
تصور می‌کنم بهتر است در خصوص آنچه در هنگام نصب SQL Server انتخاب می‌کنیم، دقت بیشتری کنیم. بسیار دیده ام که برخی از دوستان و همکاران در هنگام نصب SQL Server در قسمت انتخاب Feature ها تمامی آنها را انتخاب کرده در صورتی که تنها به Database Engine نیاز دارند و عملا با این کار ، کارایی Database Server خود را پایین می‌آورند .

بنابر این توصیه می‌کنم در پنجره ی Feature Selection فقط آنچه را که نیاز دارید نصب نمایید .

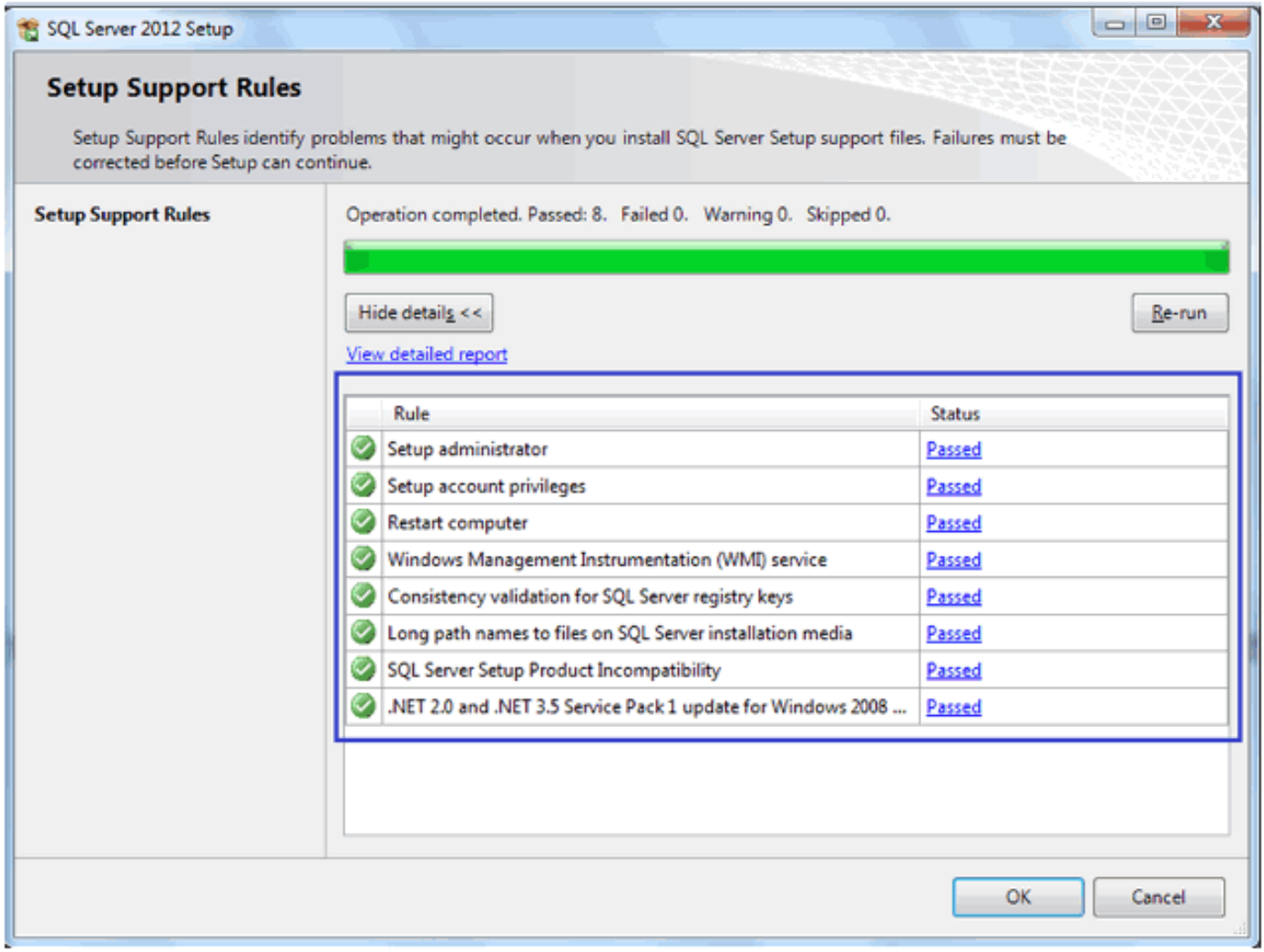
بنابر این در صورتی که شما جزو آن دسته دوستانی می‌باشید که در پنجره ی Feature Selection تمامی گزینه‌ها را انتخاب نموده اید، خوب نیازی به نصب مجدد SSAS ندارید و شما ناخواسته این سرویس را برروی Database Server خود نصب نموده اید .

در صورتی که شما قبلا این سرویس را برروی سرور خود نصب نکرده اید و فقط Database Engine را نصب نموده اید مراحل زیر را طی نمایید.

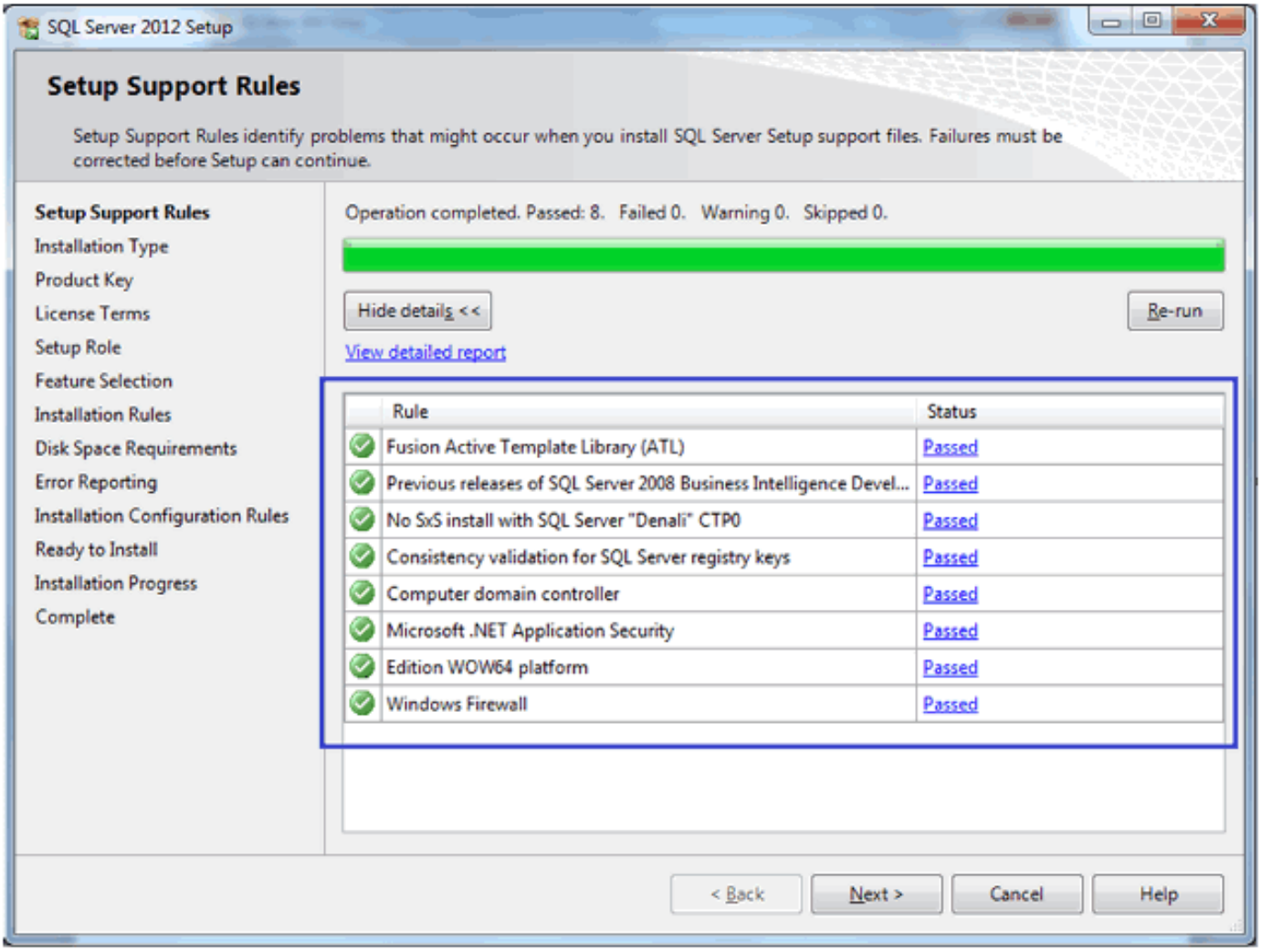
1. در ابتدا Set Up مربوط به SQL Server 2012 را اجرا نمایید. و در صفحه‌ی ابتدایی برنامه‌ی نصب ، مطابق شکل زیر روی Installation کلیک کنید. و در قسمت سمت راست گزینه‌ی New SQL Server stand-alone installation or add features to an existing installation را انتخاب نمایید.



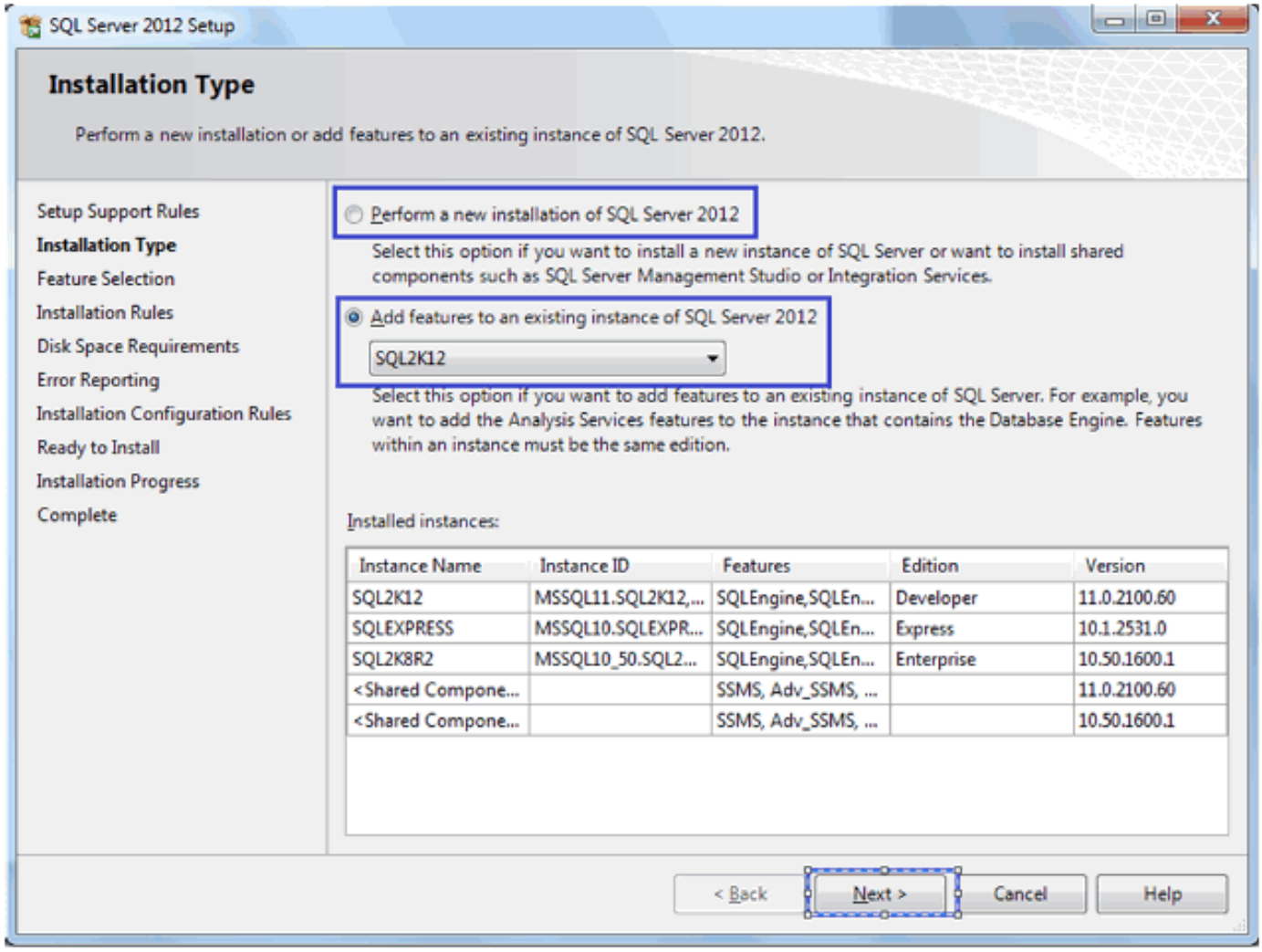
2. د رپنجره‌ی Setup Support Rules مطمئن شوید که تمامی پیش شرایط نصب را دارید (در صورتی که Warning داشته باشید احتمالا در مراحل بعدی ، نصب برنامه با مشکل روبرو خواهد شد یا بعد از نصب برخی قسمت‌های برنامه به درستی کار نمی‌کنند.) در صورتی که در قسمتی با Warning روبرو شدید بعد از برطرف کردن مشکل دکمه‌ی Rerun را بزنید به عبارت دیگر نیازی نمی‌باشد مراحل نصب را از ابتدا ادامه دهید. سپس دکمه‌ی OK را بفشارید .



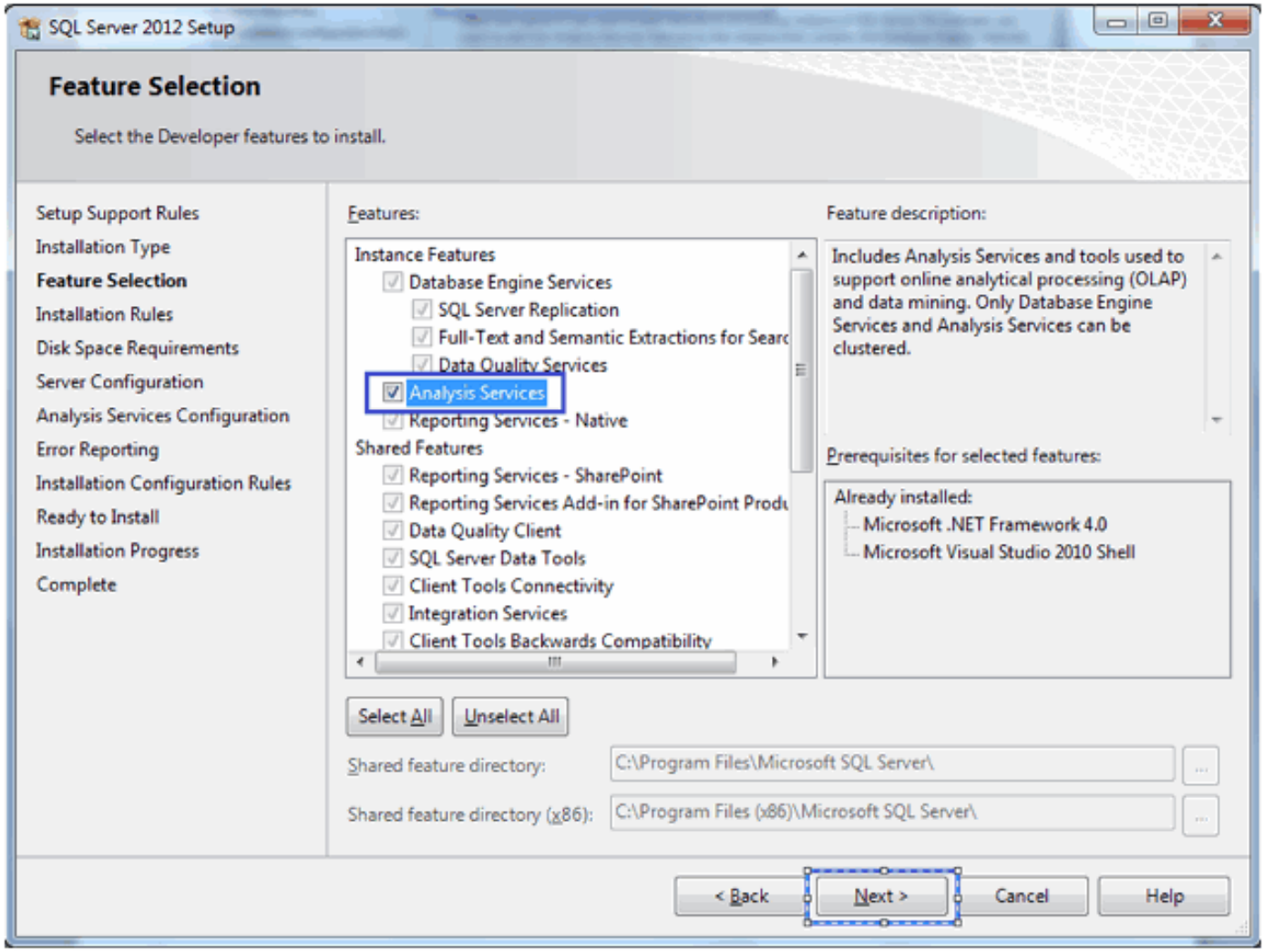
3. در پنجره‌ی بعدی دکمه‌ی Install را بزنید. سپس دوباره صفحه‌ی Setup Support Rules را خواهید داشت. مطمئن شوید تمامی پیش شرایط Passed شده باشند. سپس دکمه‌ی Next را بزنید.



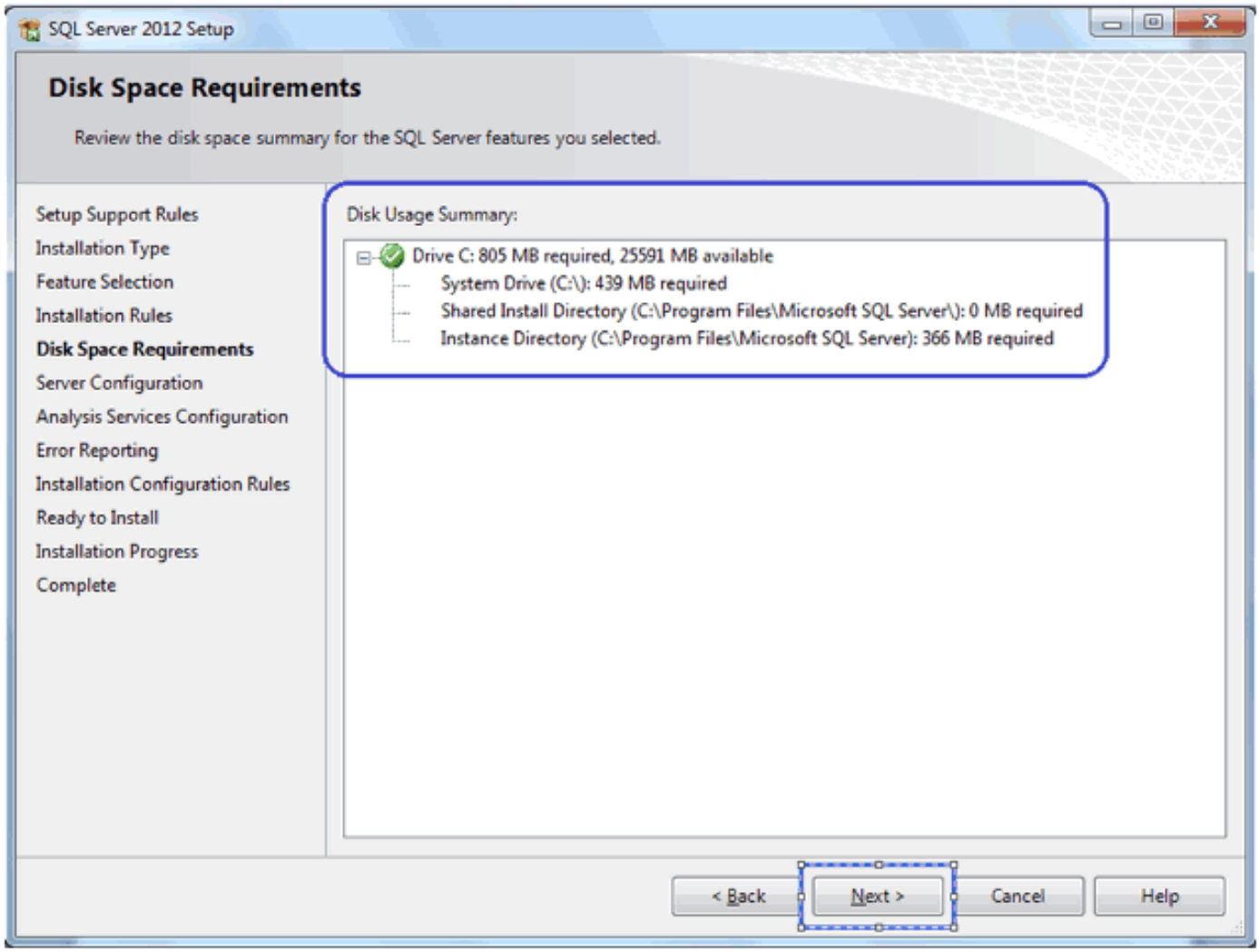
4. در پنجره‌ی بعدی گزینه‌ی Add features to and existing instance of SQL Server 2012 را انتخاب نمایید اگر شما قبلا فقط DataBase Engine را نصب کرده اید و در غیر این صورت Perform a new installation of SQL Server 2012 را انتخاب نماید.
سپس دکمه‌ی Next را بزنید.



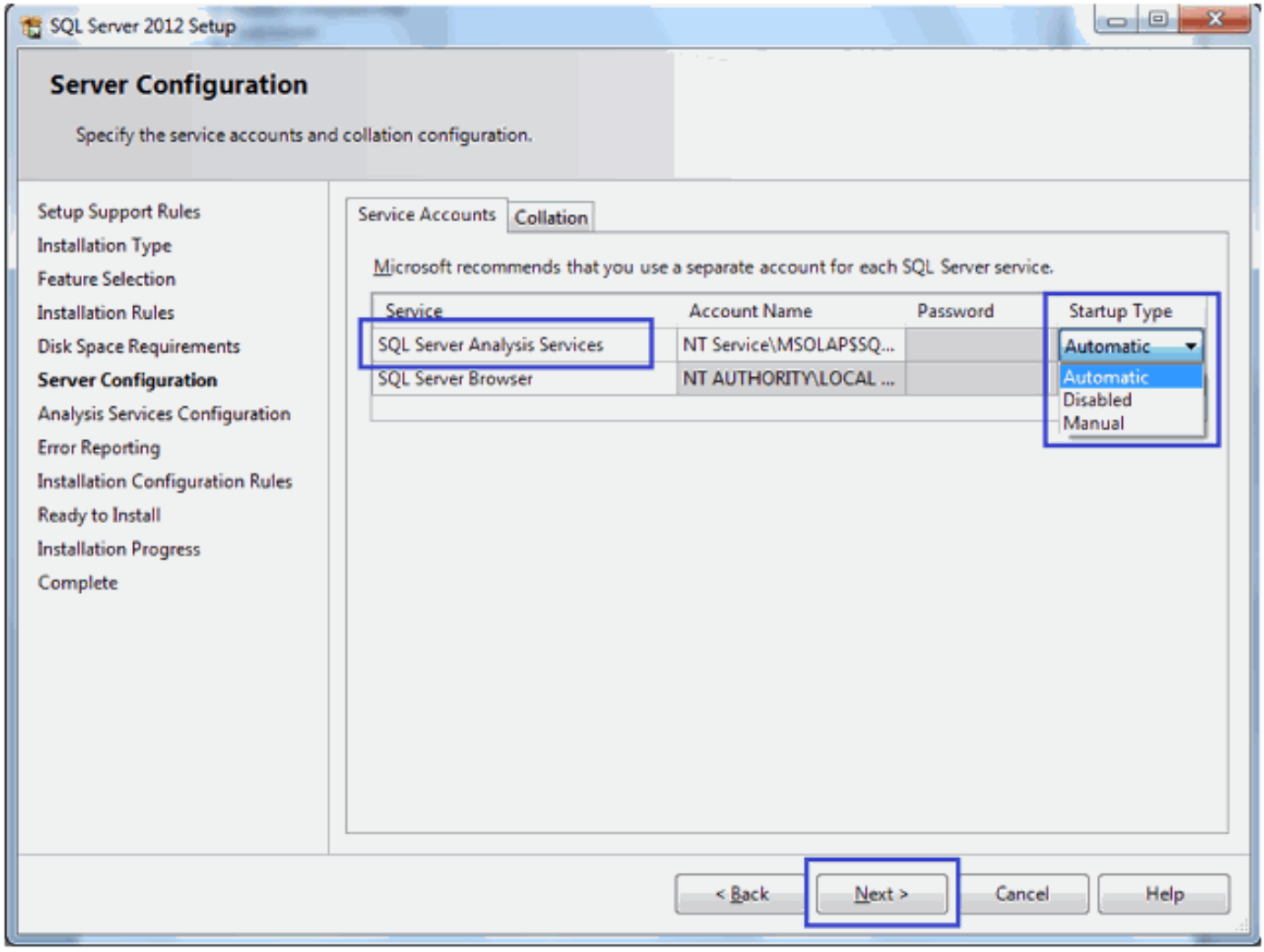
5. در صفحه‌ی Feature Selection گزینه‌ی Analysis Services را مطابق شکل زیر انتخاب نمایید.و سپس دکمه‌ی Next را بزنید .



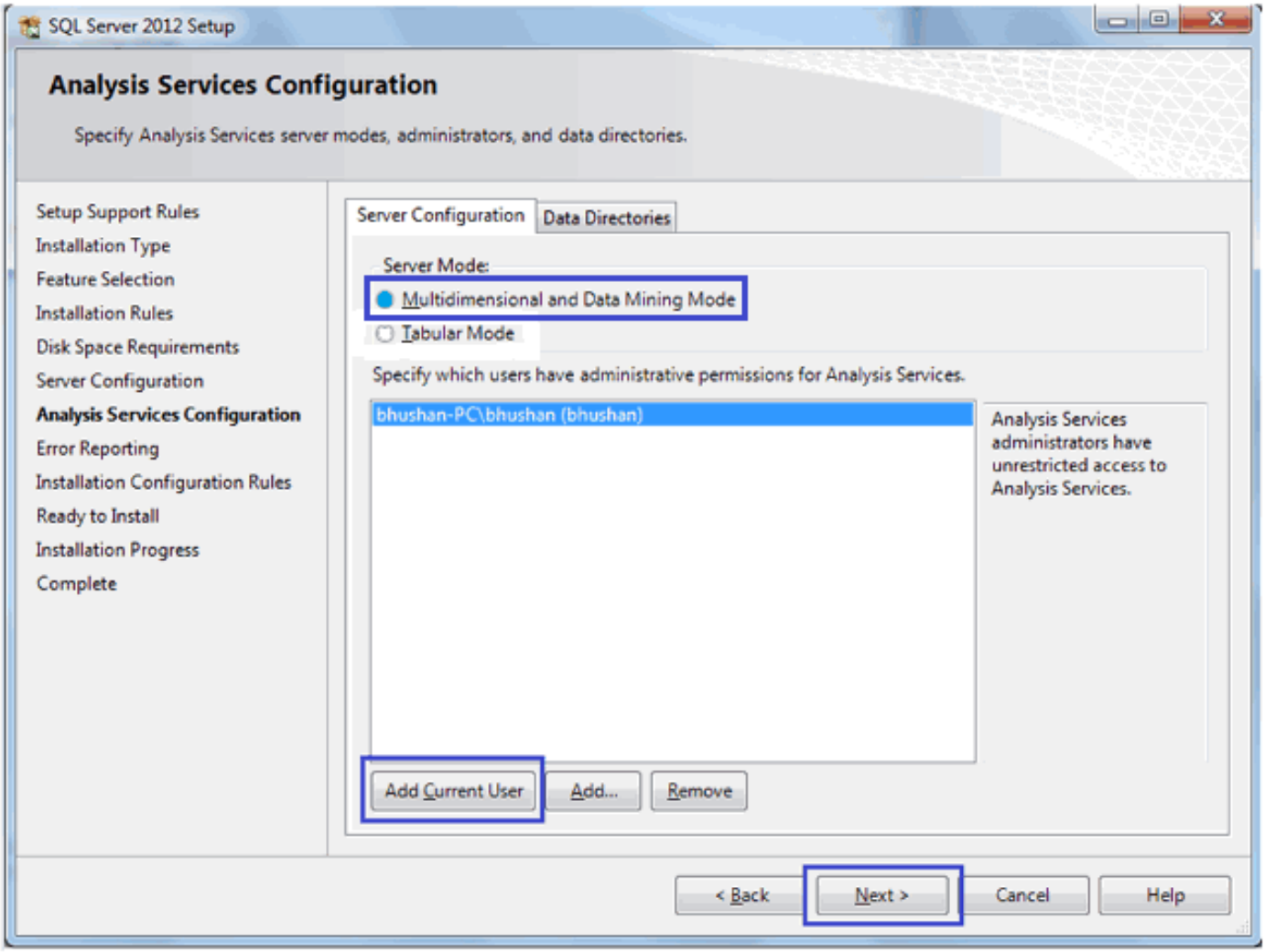
6. در صفحه‌ی بعدی برنامه‌ی نصب به شما توضیحاتی در خصوص مقدار فضای Hard برای نصب سرویس(های) انتخاب شده ، نمایش می‌دهد. دکمه‌ی Next را بزنید



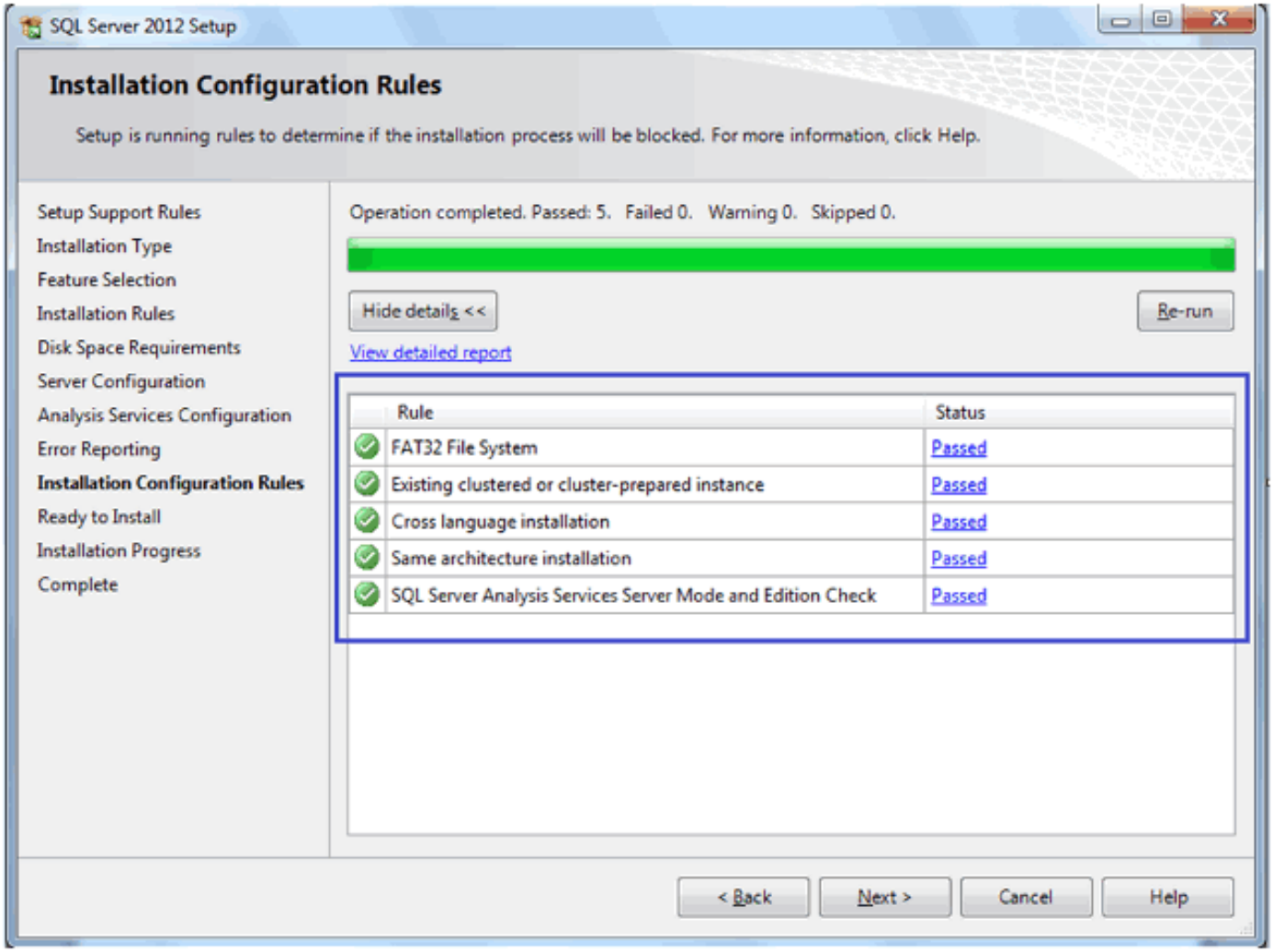
7. در صفحه‌ی Server configuration مد SQL Server Analysis Services را بر روی حالت Automatic تنظیم کنید. سپس دکمه‌ی Next را بزنید.



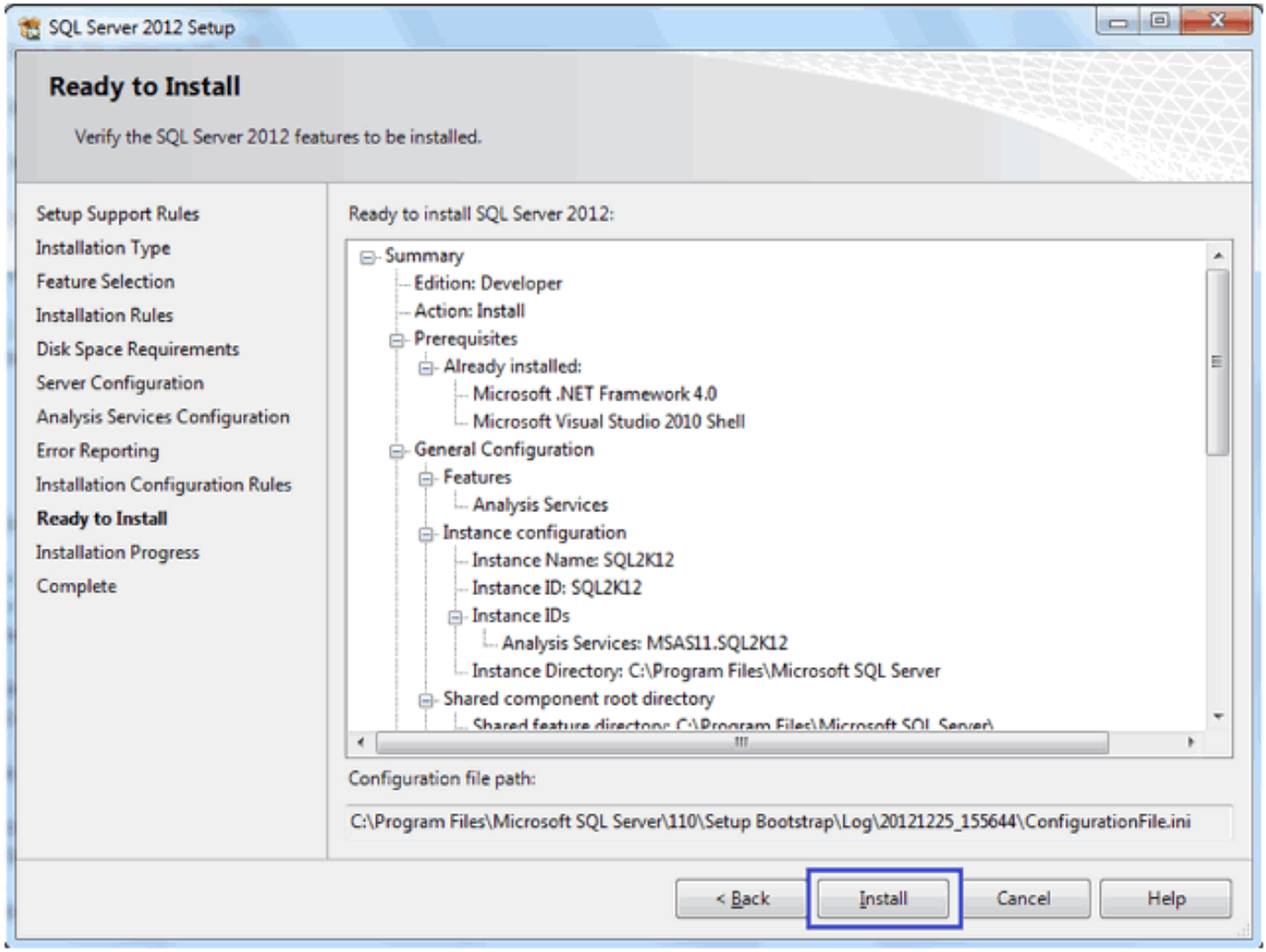
8. سپس در صفحه‌ی Analysis Services Configuration گزینه‌ی Multidimensional and Data Mining Mode را انتخاب نمایید. و همچنین برای مشخص نمودن Administrator سرویس SSAS نام کاربر را در قسمت پایین پنجره وارد نمایید. در صورتی که شما با کاربری که عملا Administrator سرویس SSAS می‌باشد در سیستم عامل ویندوز لاگین نموده اید می‌توانید دکمه‌ی Add Current User را بزنید. سپس دکمه‌ی Next را بزنید .



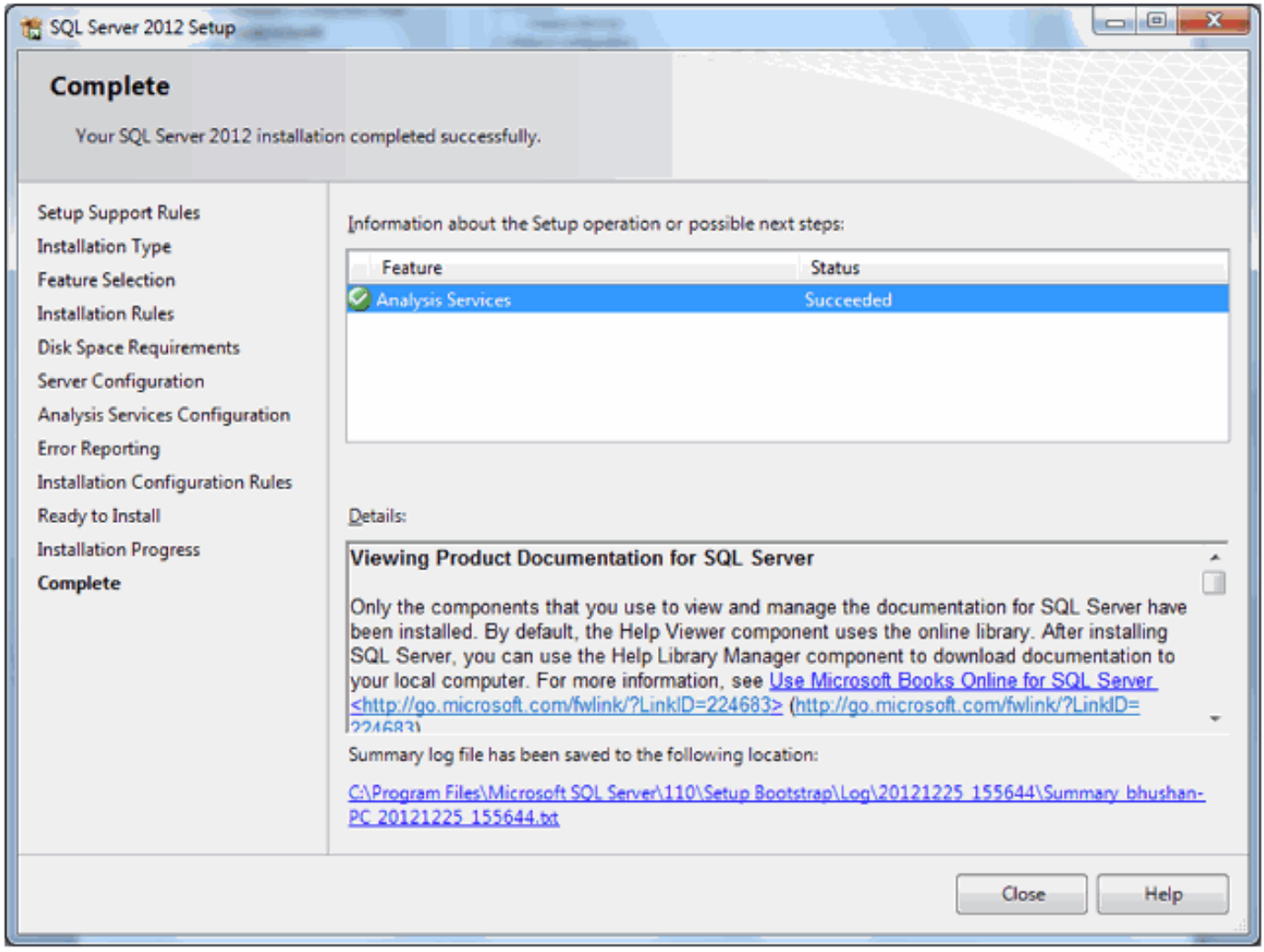
9. در صفحه‌ی بعد بررسی‌های Installation Configuration Rules انجام می‌گردد . دقت داشته باشید که تمامی موارد Passed گردیده باشند. سپس دکمه‌ی Next را بزنید .



10. در صفحه‌ی Ready to install دکمه‌ی Install را بفشارید.

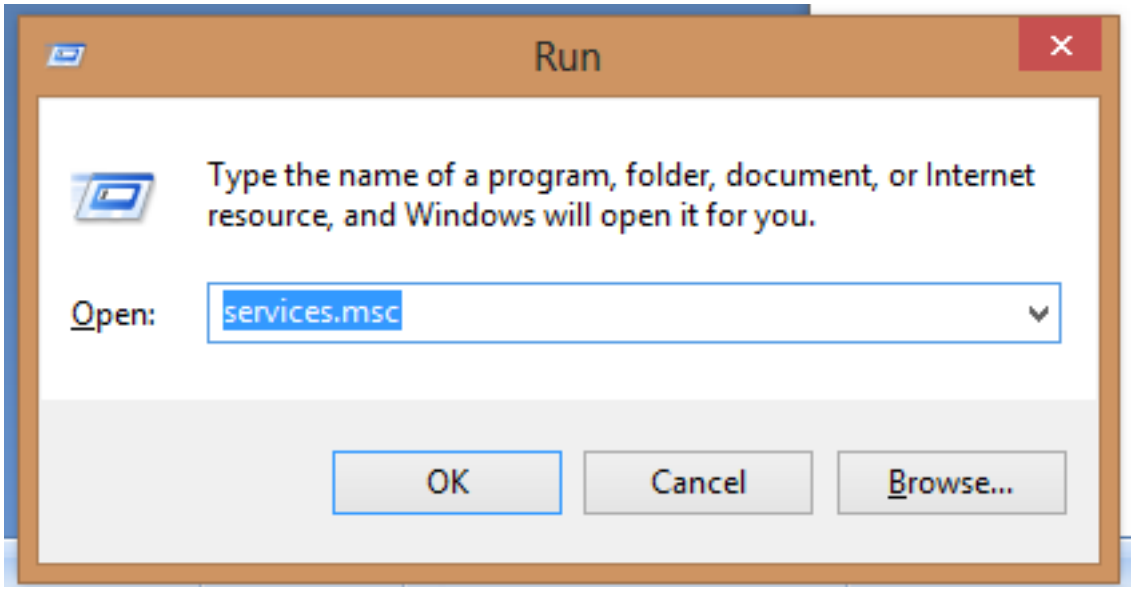


11. در صورتی که نصب با موفقیت انجام شده باشد، صفحه ای به شکل زیر خواهید دید.

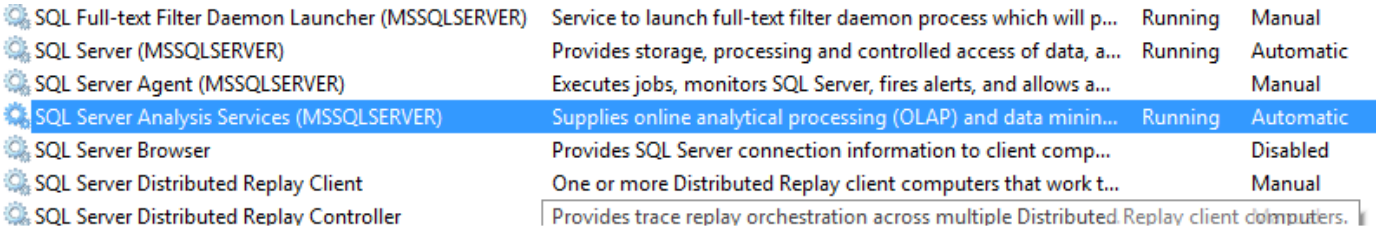


خوب به شما تبریک می‌گوییم شما هم اکنون سرویس SSAS را برروی سرور خود نصب نموده اید. برای اطمینان از تنظیمات Registry توصیه می‌کنم سیستم عامل خود را Restart نمایید.

برای اطمینان از نصب سرویس SSAS بر روی سیستم خود می‌توانید در پنجره‌ی Run عبارت services.msc را وارد کنید .



سپس در قسمت سرویس‌ها شما می‌توانید سرویس SSAS را مشاهده نمایید مطابق شکل زیر.



در قسمت‌های بعدی این سری از آموزش‌های MDX Query تلاش خواهم کرد طریقه‌ی نصب پایگاه داده‌ی Adventure Work DW و همچنین ساخت پایگاه داده‌ی Multidimensional مربوط به Adventure Work DW را آموزش دهم.

عنوان: آموزش MDX Query - قسمت سوم - نصب Adventure Work DW و تهیه ی پایگاه داده ی Multidimensional Database توسط SSAS
نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۰۹/۲۳ ۹:۳۰
آدرس: www.dotnettips.info
گروه‌ها: SQL, OLAP, MDX, SSAS

برای ادامه دادن این سری از مقالات آموزش MDX Query نیاز می‌باشد که پایگاه داده‌ی Advwnture Work DW را نصب کرده و سپس توسط SSAS عمل Deploy را انجام دهیم تا پایگاه داده‌ی Multidimensional Database توسط SSAS ساخته شود .

در ابتدا می‌بایست فایل نصب پایگاه داده ی Advwnture Work را دانلود نمایید برای این منظور به آدرس زیر رفته و فایل AdventureWorks2008R2_SR1.exe را دانلود نمایید .

http://www.general-files.biz/download/gs4ac37d18h17i0/AdventureWorks2008R2_SR1.exe.html

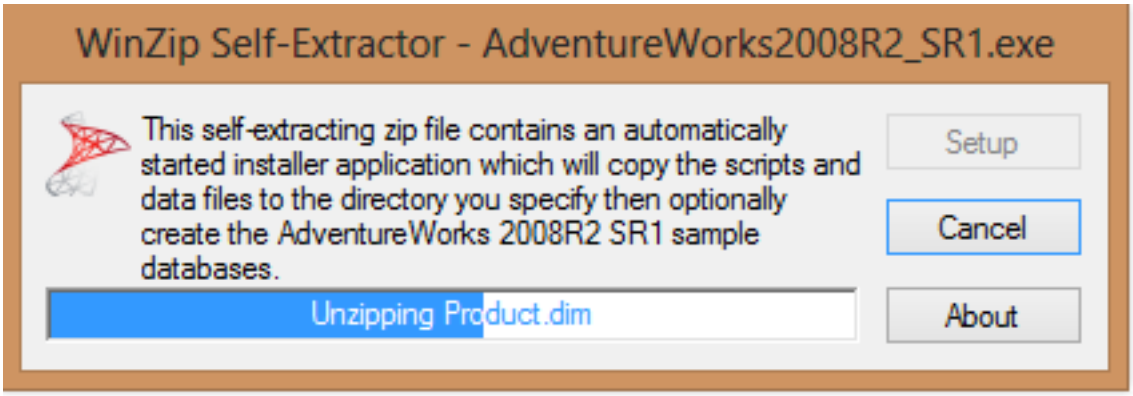
یا به آدرس زیر مراجعه کنید

https://msftdbprodsamples.codeplex.com/releases/view/59211

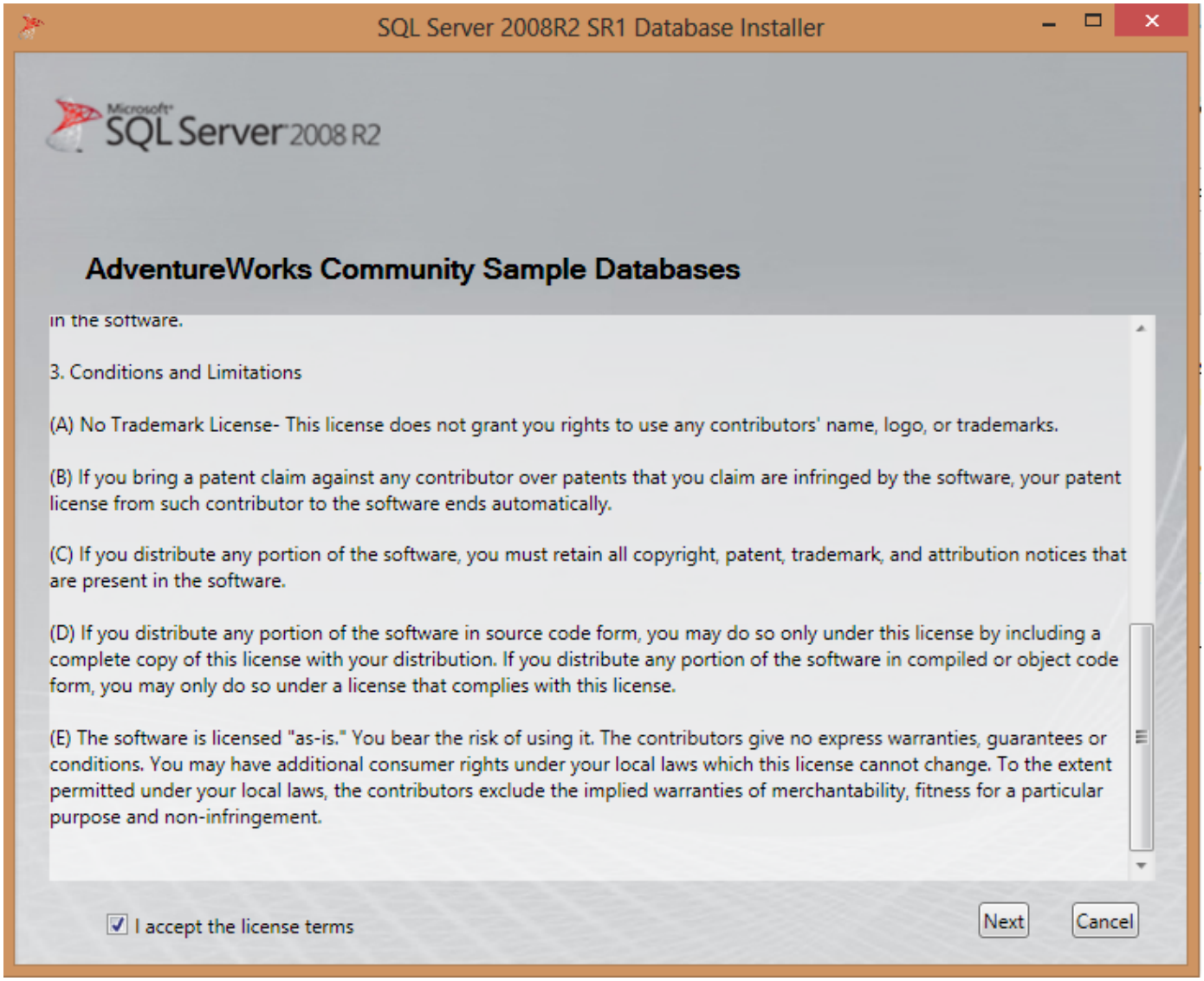
نیاز می‌باشد قبل از شروع به نصب نرم افزار SQL Server Management Studio را ببندید.

سپس مراحل زیر را انجام دهید.

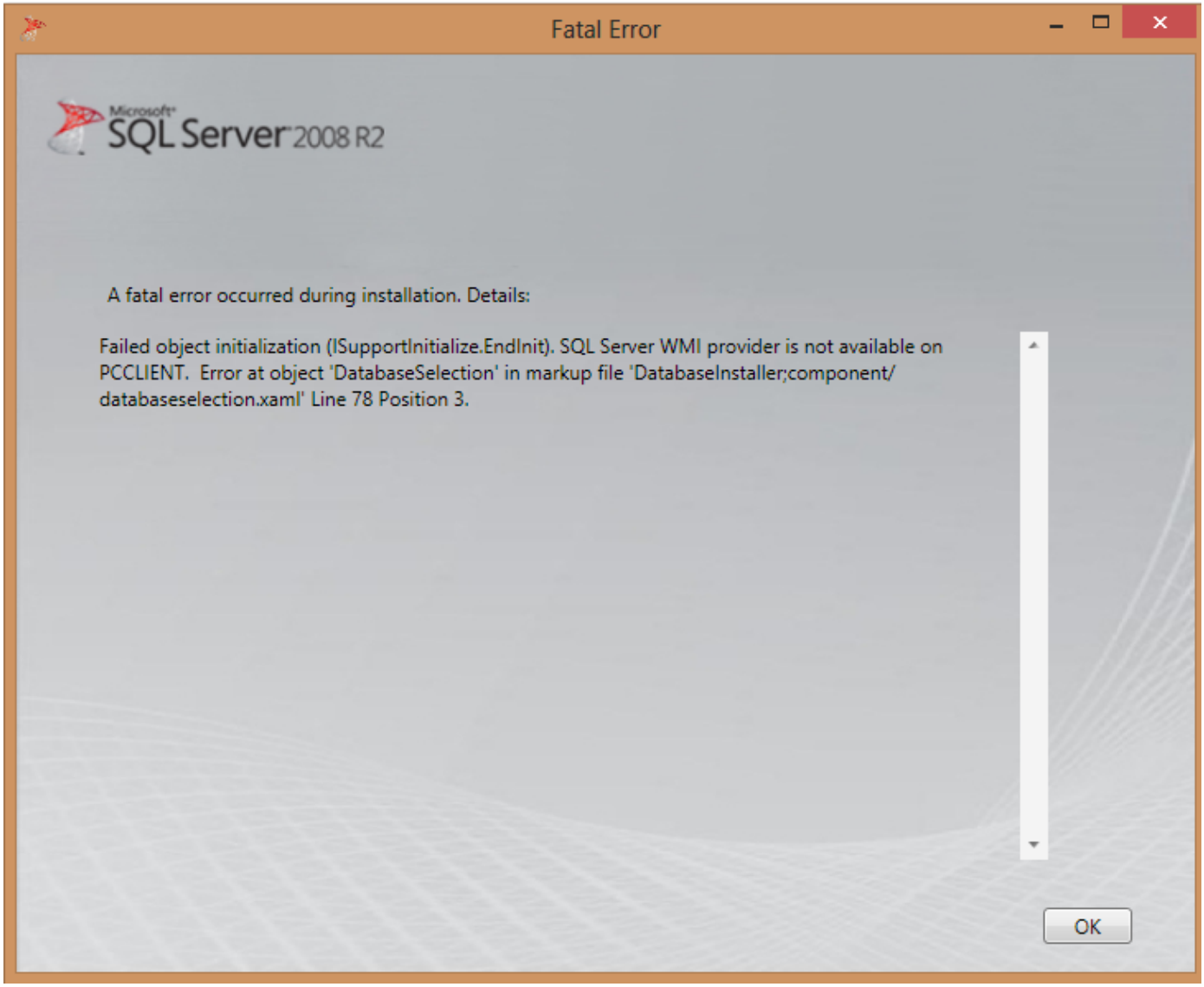
1. فایل AdventureWorks2008R2_SR1.exe را اجرا نمایید.



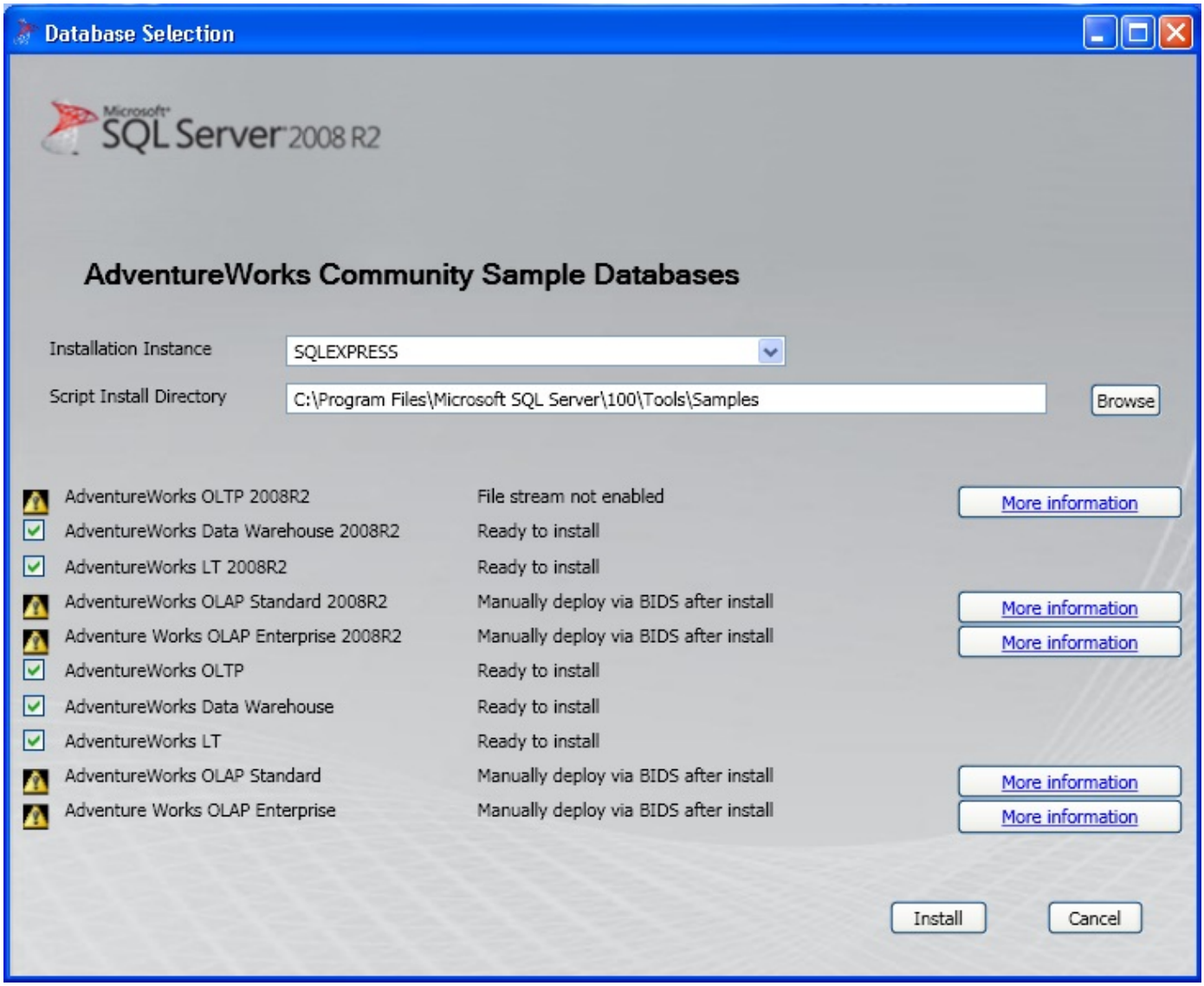
2. کمی صبر کنید تا صفحه‌ی زیر نمایش داده شود. و گزینه‌ی I Accept ... را انتخاب نماید و دکمه‌ی Next را بزنید.



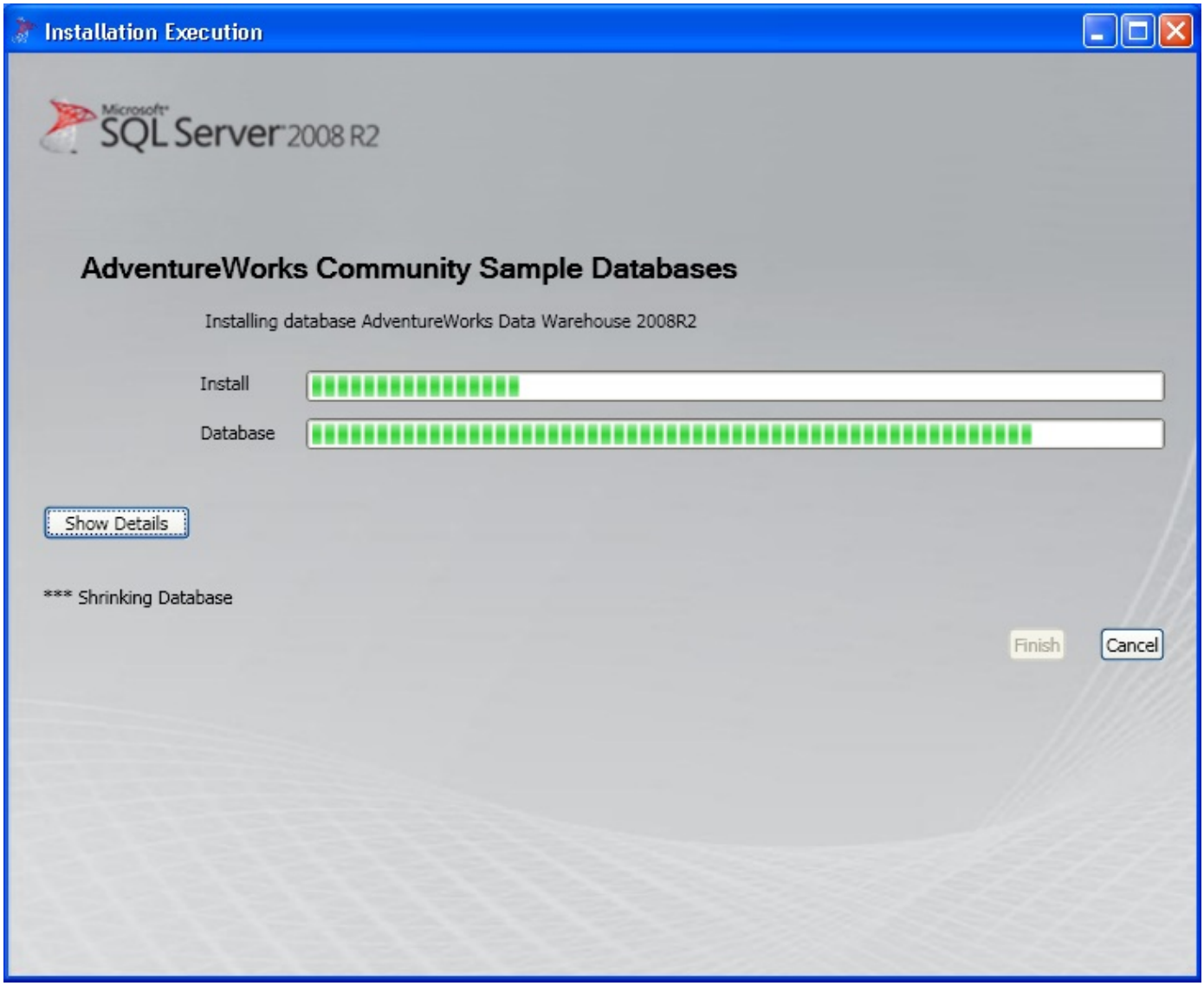
3. در صورتی که از ویندوز 8 استفاده نماید احتمال دارد با خطای زیر مواجه شوید در این صورت به قسمت روش نصب در ویندوز 8 در ادامه‌ی این مقاله مراجعه کنید .

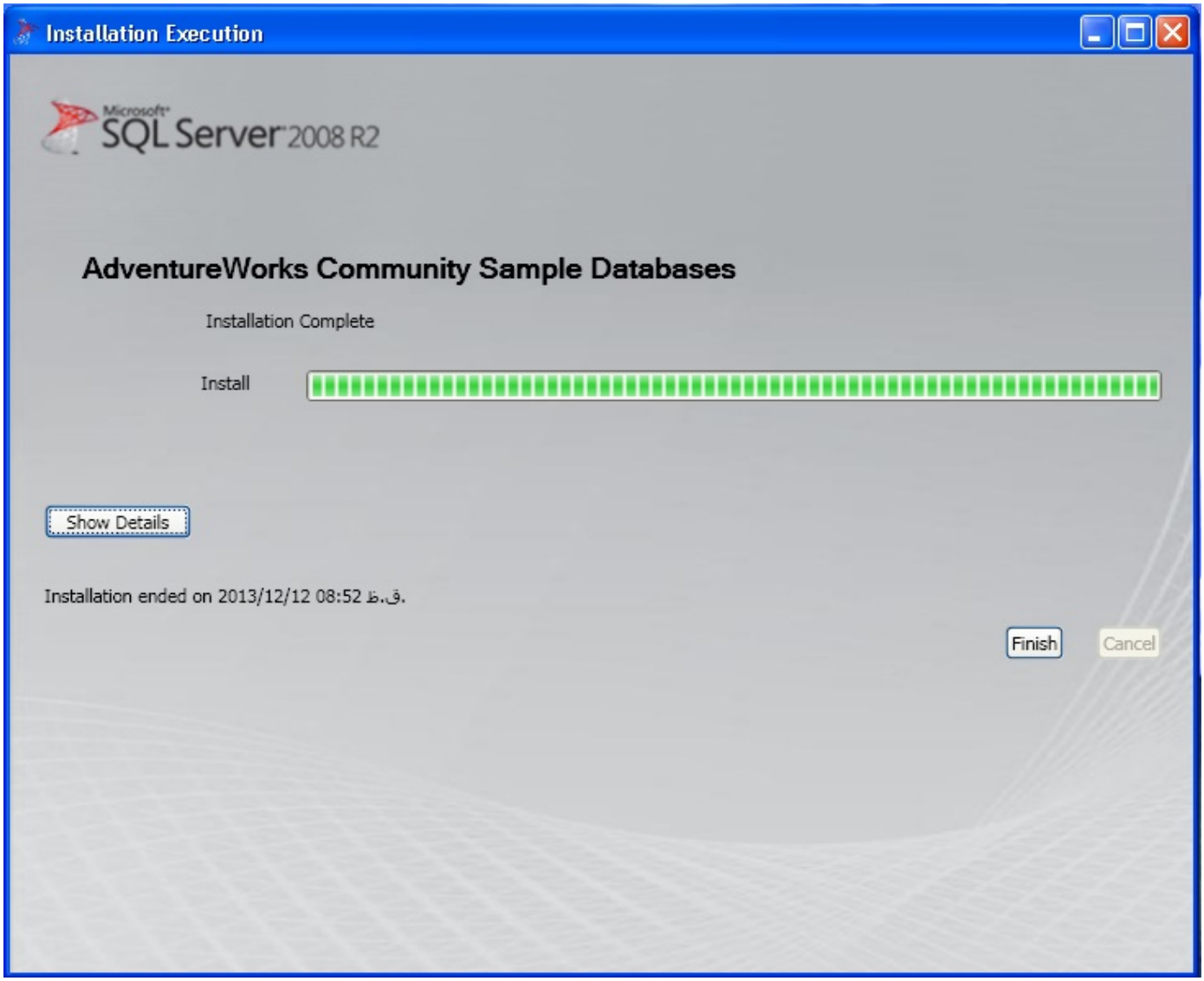


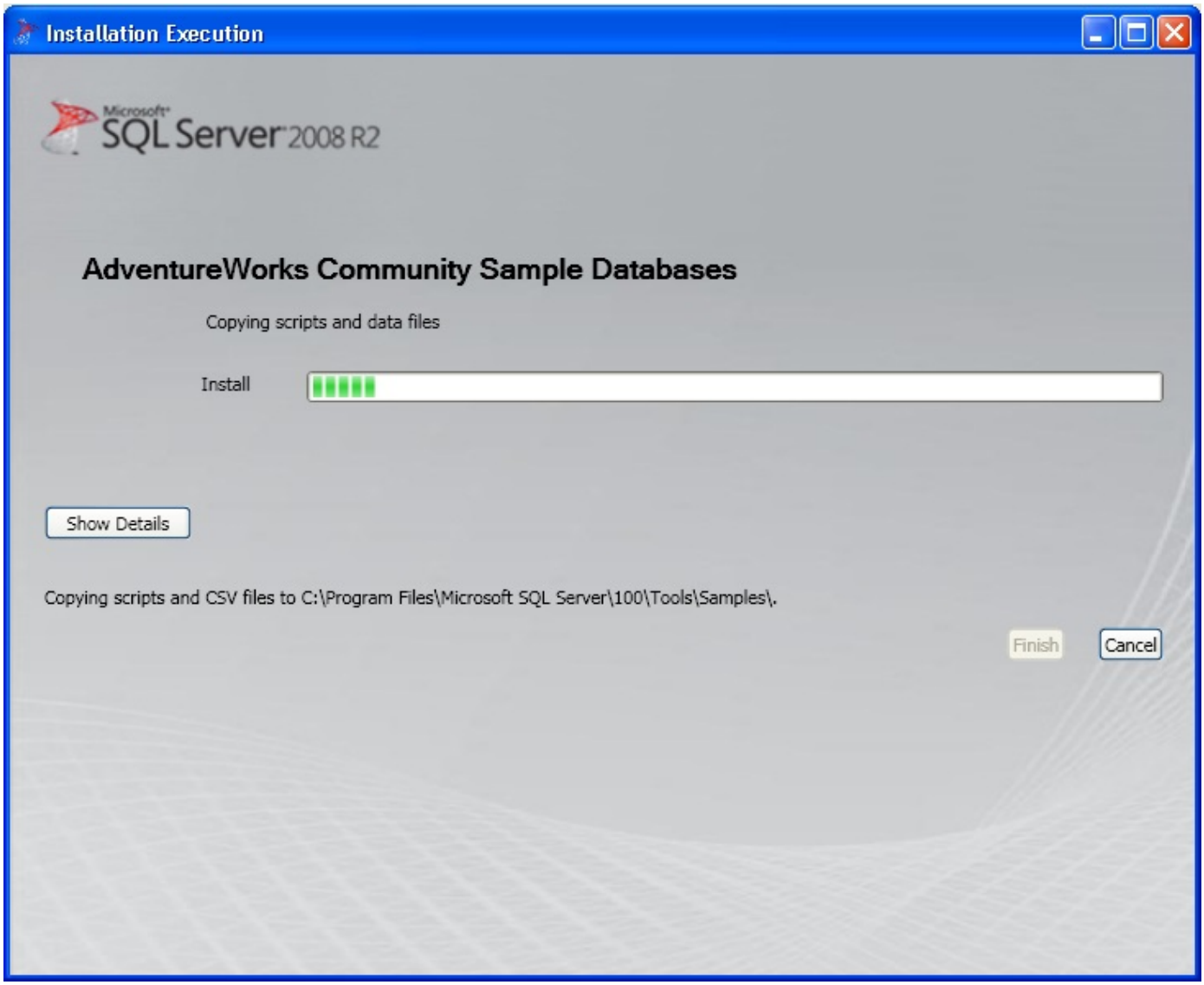
4. در صورتی که از ویندوزهای Win7 , XP, 2003 Server استفاده کنید صفحه‌ی زیر را خواهید دید. در این صفحه ابتدا Instance مربوط به SQL سرور خود را انتخاب نماید (در صورت داشتن چندین Instance روی سرور پایگاه داده) سپس مسیر نصب فایل‌های Sample را مشخص نمایید (بعدا از همین مسیر اقدام به Deploy کردن پایگاه داده‌ی Multidimensional خواهیم کرد) و پیش فرض‌ها را بپذیرید و دکمه‌ی Install را بزنید.



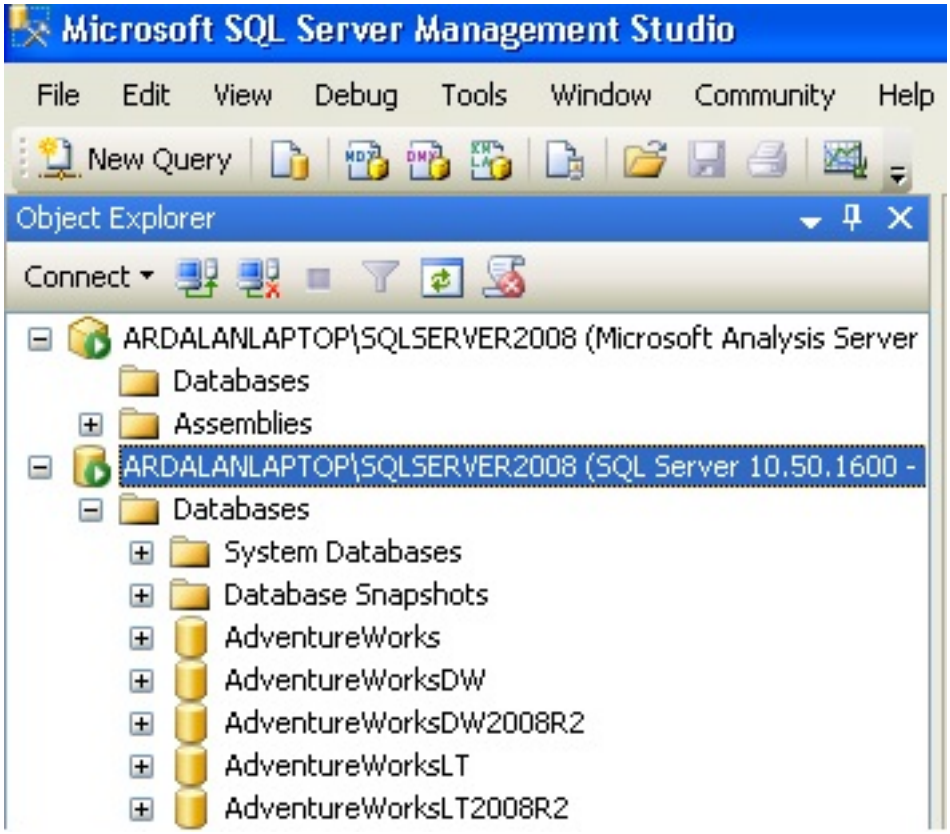
5. کمی صبر کنید تا نصب انجام گردد. و در انتها کلید Finish را بزنید.


Installation Execution
Microsoft SQL Server 2008 R2
AdventureWorks Community Sample Databases
Installing database AdventureWorks Data Warehouse 2008R2
Install
Database
Show Details
*** Shrinking Database
Finish
Cancel

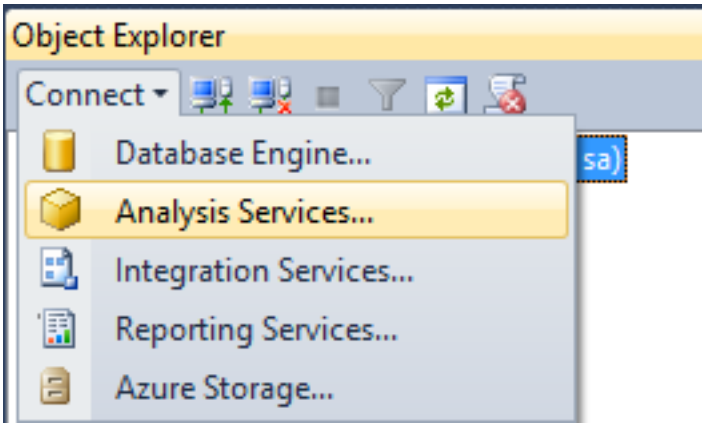

Installation Execution
Microsoft SQL Server 2008 R2
AdventureWorks Community Sample Databases
Installation Complete
Install
Show Details
Installation ended on 2013/12/12 08:52 ق.ظ.
Finish
Cancel



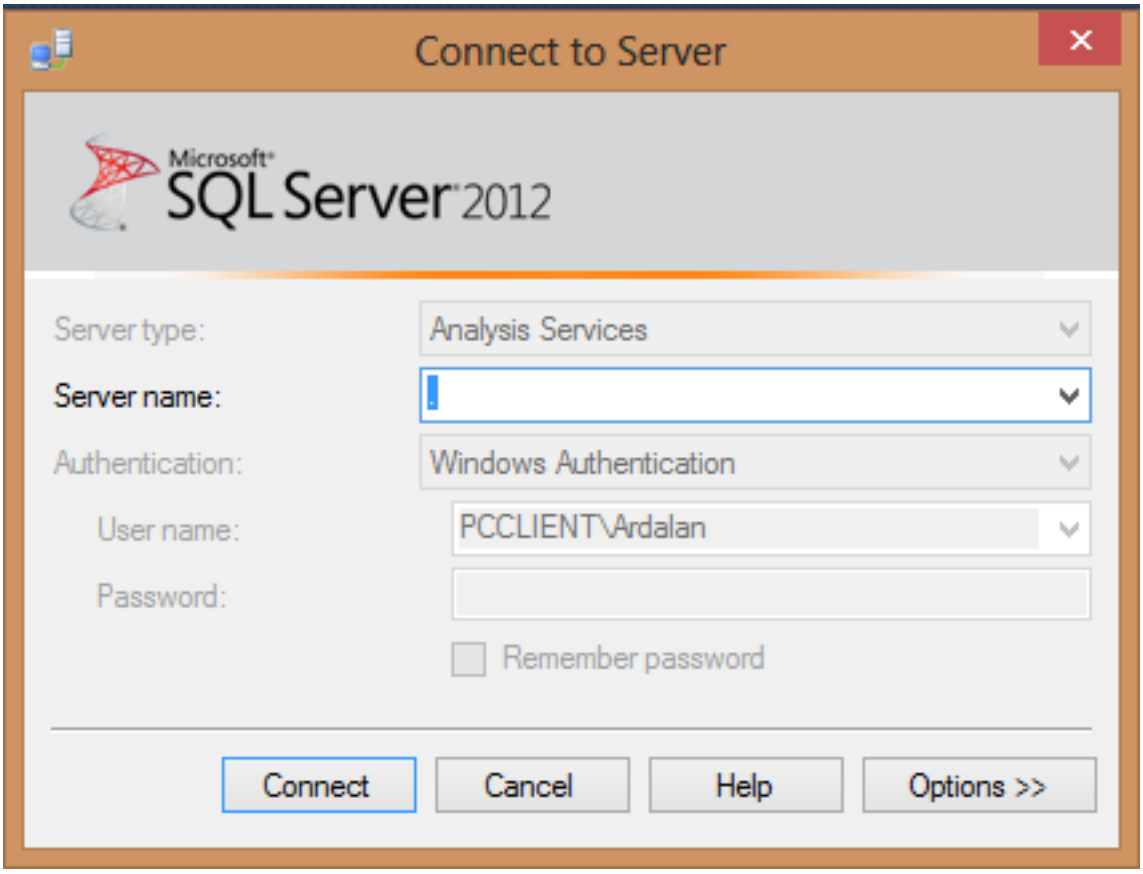
پس از مراحل بالا (به جز ویندوز 8) با باز کردن نرم افزار SQL Server Management Studio و اتصال به سرویس Database Engine در قسمت Database تصویر زیر را خواهید دید (البته امکان دارد شما از قبل دارای پایگاه داده‌های شخصی بوده باشید که بنابر این آنها نیز در لیست شما وجود خواهند داشت)



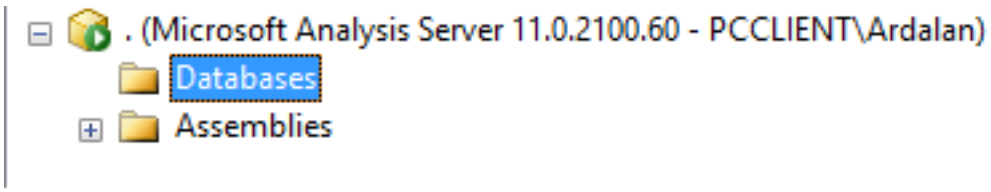
همچنین شما می‌توانید از پنجره‌ی Object Explorer در قسمت Connect اقدام به اتصال به سرویس SSAS نموده .



و در پنجره‌ی باز شده Server Name را انتخاب نمایید (با توجه به اینکه شما در حال حاضر می‌خواهید به SSAS موجود در سیستم Local متصل شوید ، بنابر این انتخاب سرور Local با وارد کردن کاراکتر (.) انجام می‌شود.)



بعد از اتصال شکل زیر را خواهید داشت و در شاخه‌ی Database همچنان هیچ Multidimensional Database ی نخواهید داشت.(بعد از عمل Deploy که در ادامه آموزش داده خواهد شد پایگاه داده‌ی Multidimensional ساخته می‌شود.)



تنها روشی که تاکنون برای نصب پایگاه داده‌ی Adventure Work DW برروی ویندوز 8 یافته ام (البته کمی غیر حرفه ای می‌باشد.) به صورت زیر می‌باشد.

فایل بالا را (AdventureWorks2008R2_SR1.exe) روی سیستم عامل‌های (Server 2003,XP,Win 7) نصب کرده (به عنوان یک سیستم عامل واسط) و سپس سرویس Database Engine را Stop کرده و فایل‌های پایگاه داده را به سیستم عامل ویندوز 8 انتقال داده و به صورت دستی Restore کنیم.

مراحل ایجاد پایگاه داده‌ی Multidimensional در ویندوزهای مختلف ، یکسان می‌باشد.

بعد از نصب پایگاه داده‌ی Adventure Work DW باید به شاخه‌ی نصب Sample بروید (همان مسیری که در مراحل نصب وارد کردیم

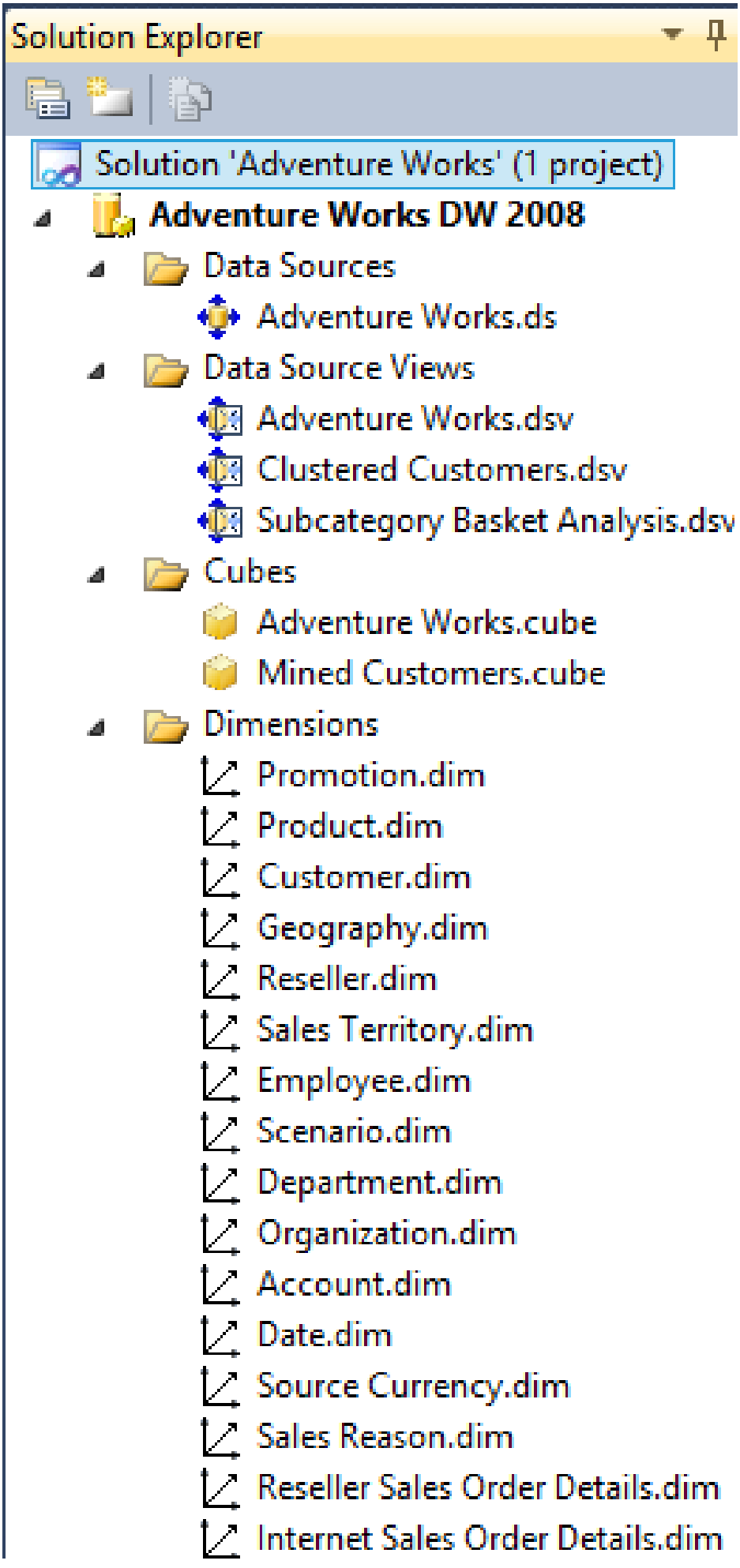
و البته آدرس پیش فرض آن C:\Program Files\Microsoft Sql Server\100\Tools\Sample می‌باشد.)

(در صورتی که در ویندوز 8 مراحل نصب را دنبال می‌کنید مسیر زیر را در سیستم خود درست نمایید و فایل‌ها و پوشه‌های موجود در مسیر فوق در سیستم عامل واسط (همان سیستم عاملی که فایل نصب بر روی آن نصب شده است) را به درون آن انتقال دهید.)

سپس به زیر شاخه‌ی AdventureWorks 2008R2 Analysis Services Project\enterprise \ بروید و فایل Adventure Works.sln را با Visual Studio 2010 باز کنید.

احتمال دارد که نیاز باشد روی کل شاخه‌ی enterprise در قسمت Security کاربر جاری را Add کنید و به آن دسترسی Full Control بدهید تا عملیات Convert این پروژه به درستی انجام شود.

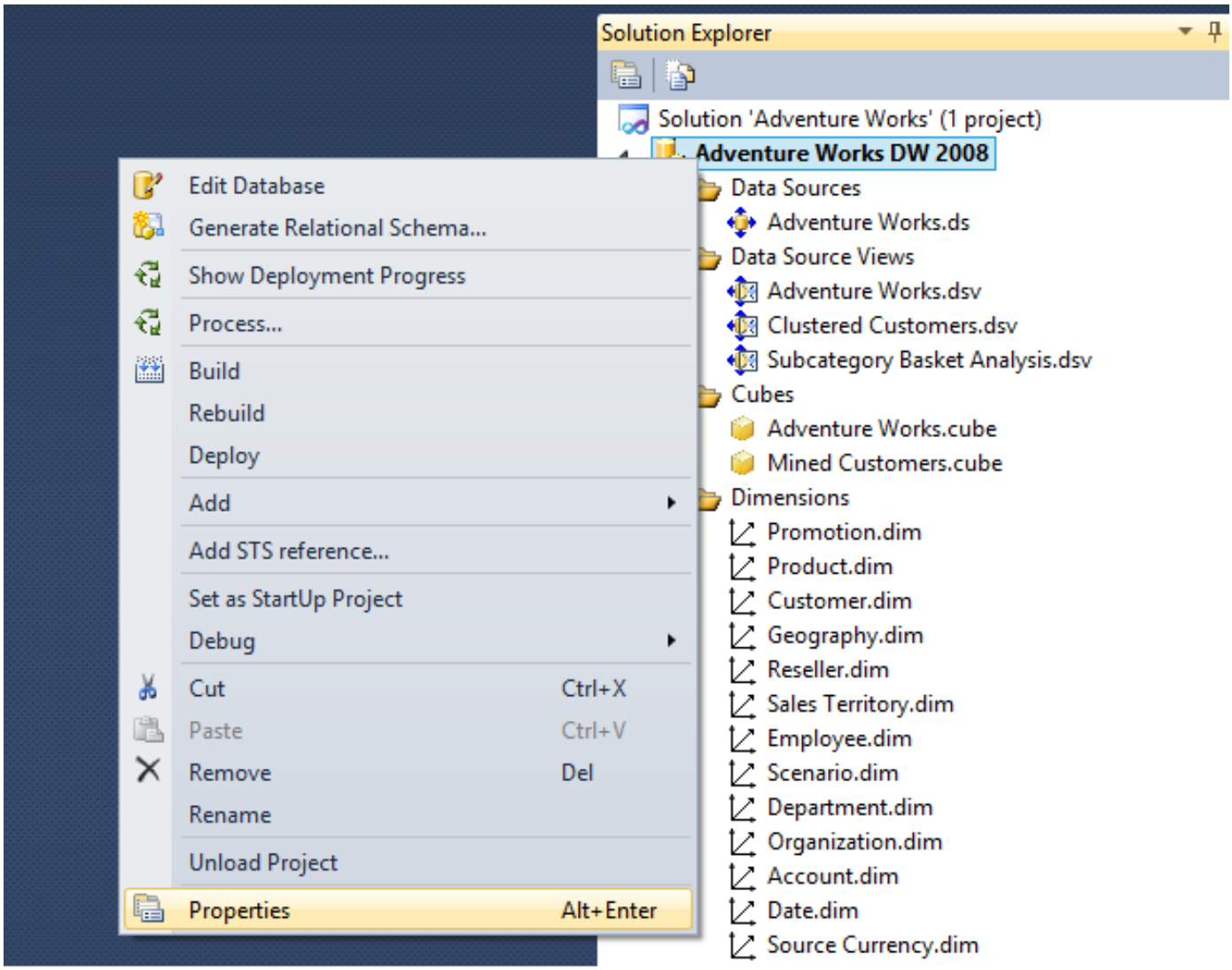
پس از باز کردن پروژه در Visual Studio 2010 صفحه ای مطابق تصویر زیر در پنجره‌ی Solution Explorer خواهید دید.


Solution Explorer
Solution 'Adventure Works' (1 project)
Adventure Works DW 2008
Data Sources
Adventure Works.ds
Data Source Views
Adventure Works.dsv
Clustered Customers.dsv
Subcategory Basket Analysis.dsv
Cubes
Adventure Works.cube
Mined Customers.cube
Dimensions
Promotion.dim
Product.dim
Customer.dim
Geography.dim
Reseller.dim
Sales Territory.dim
Employee.dim
Scenario.dim
Department.dim
Organization.dim
Account.dim
Date.dim
Source Currency.dim
Sales Reason.dim
Reseller Sales Order Details.dim
Internet Sales Order Details.dim

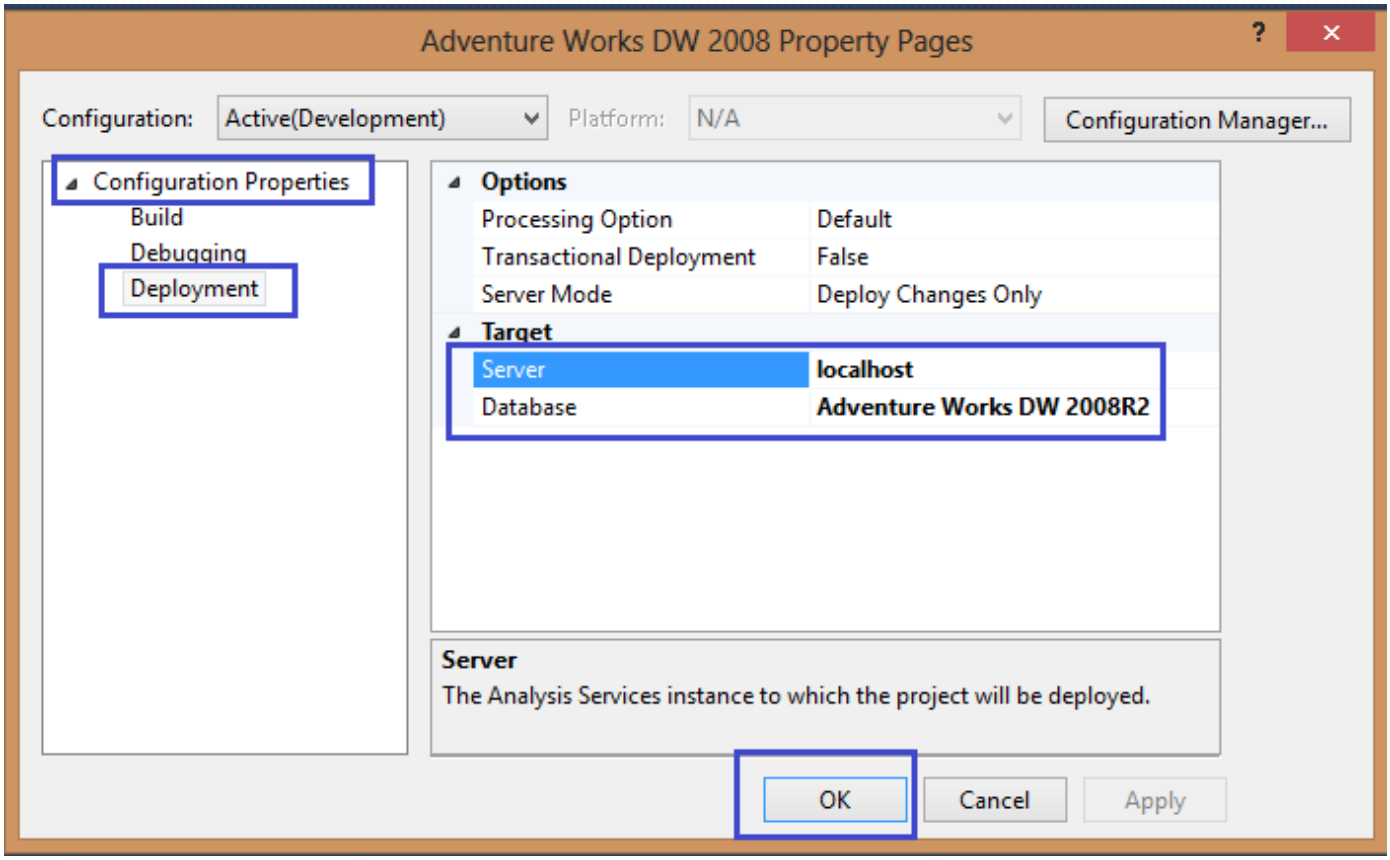
به هیچ عنوان نگران ساختار این پروژه نباشید ، زیرا در مقاله‌های آیند شرح کاملی در این خصوص کار با Business Intelligence Management Studio خواهم داد. فعلا هدف ما ایجاد پایگاه داده‌ی Multidimensional می باشد.

برای ساخت پایگاه داده‌ی Multidimensional مراحل زیر را دنبال نمایید.

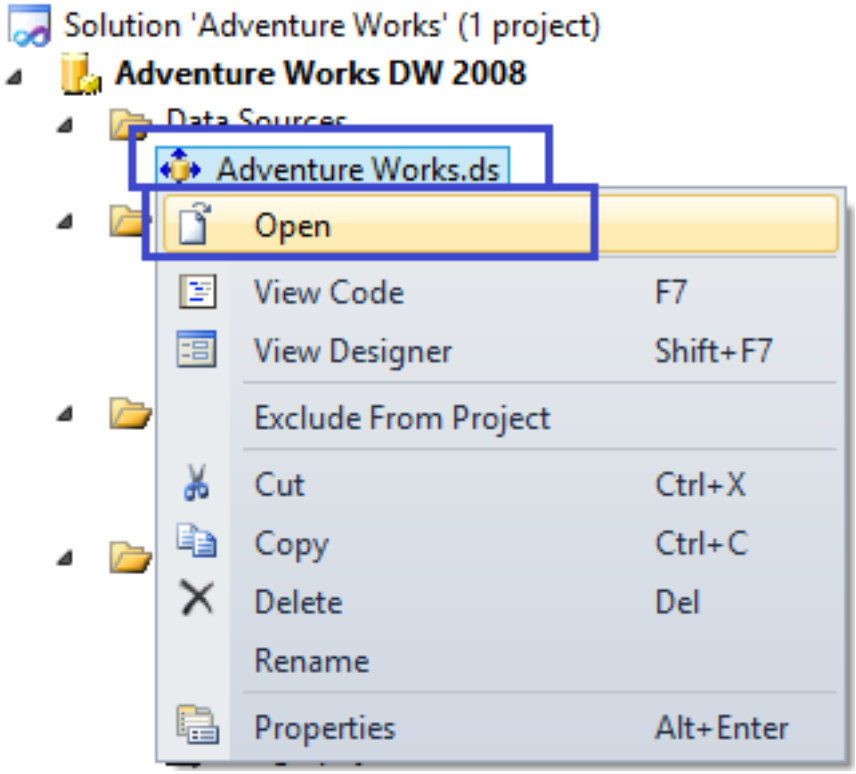
1. در ابتدا روی پروژه کلیک راست کرده و گزینه‌ی Properties را انتخاب نمایید.



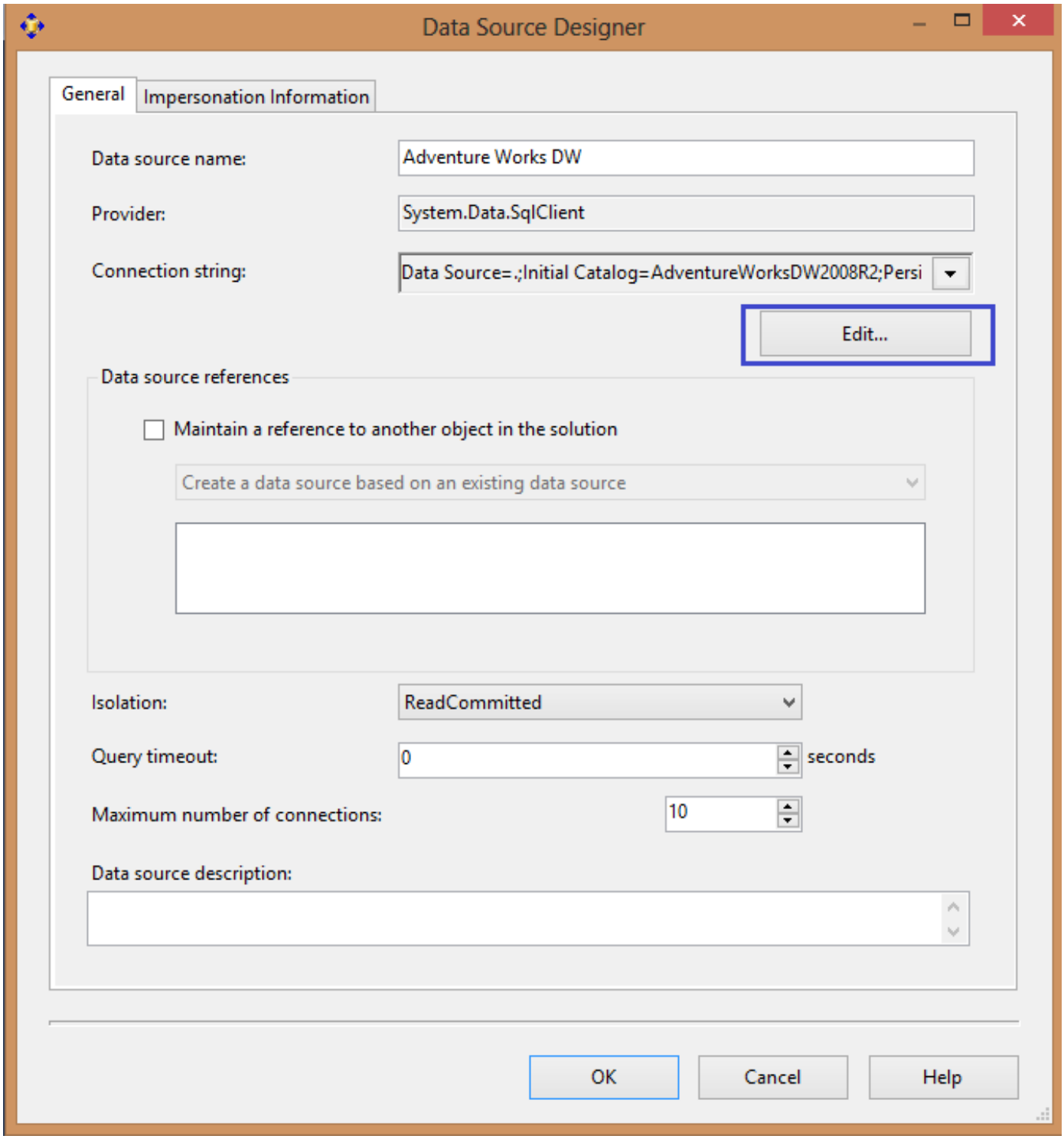
2. در قسمت Configuration Properties منوی Deployment را انتخاب کرده و اطمینان حاصل کنید که سرور شما LocalHost و نام پایگاه داده شما Adventure Works DW 2008R2 باشد.



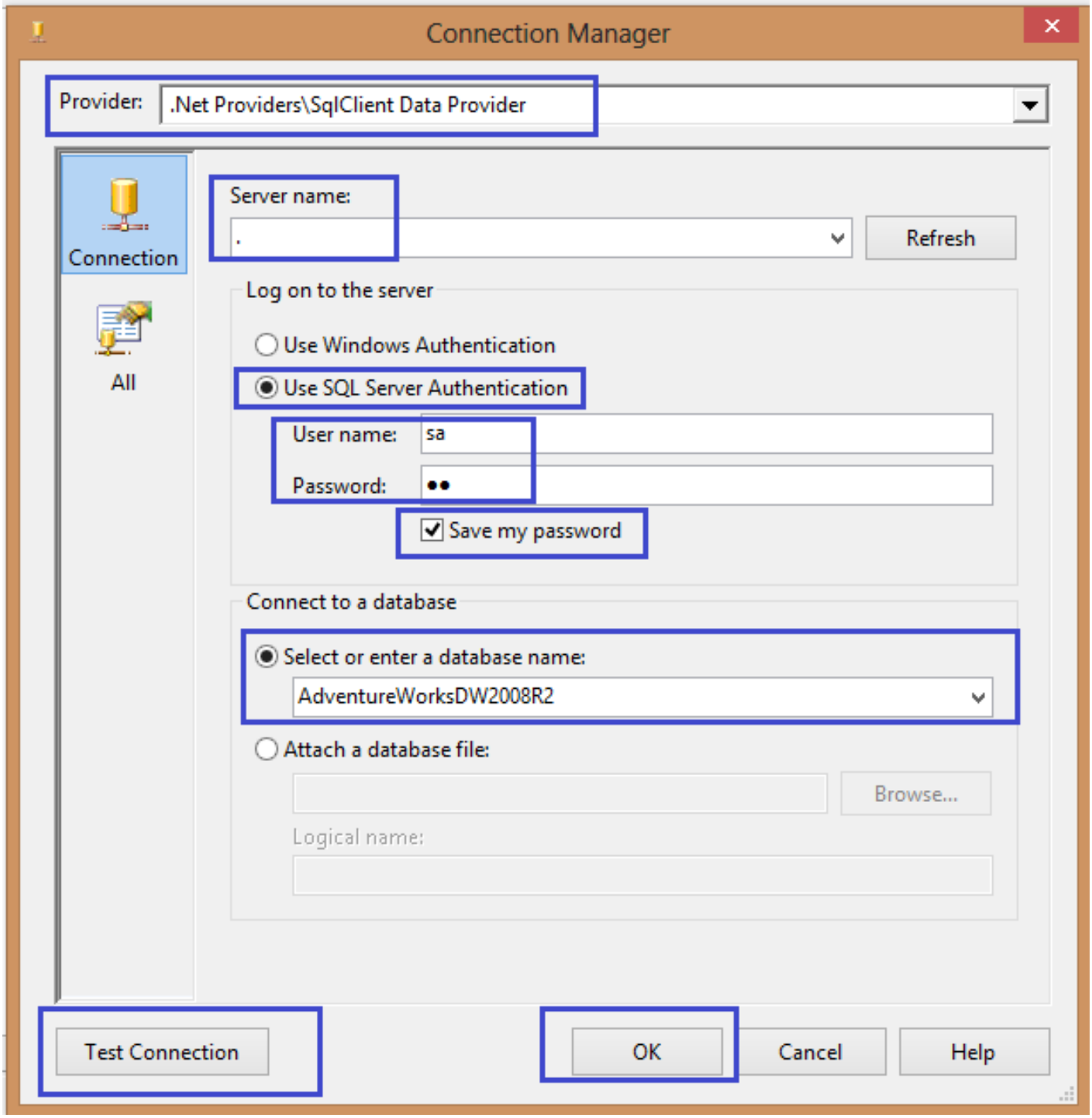
3. سپس روی Adventure Works.ds کلیک راست کنید تا تنظیمات Connection String به DW را انجام دهیم. مطابق شکل زیر



4. سپس در پنجره‌ی باز شده دکمه‌ی Edit را بزنید .



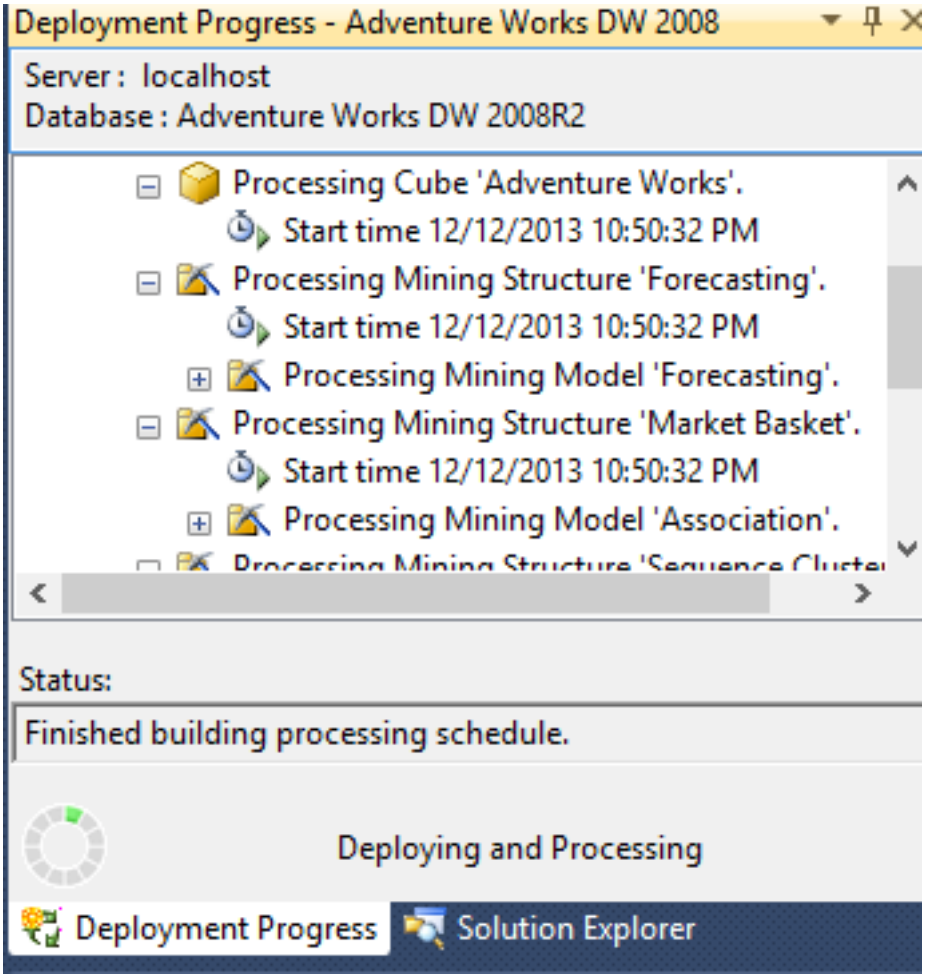
5. و در صفحه باز شده تنظیمات زیر را مطابق تصویر زیر انجام دهید. دقت داشته باشید که تغییرات را از بالا به پایین باید انجام دهید و قبل از زدن دکمه‌ی OK حتما Test Connection را بزنید تا از صحت تنظیمات مطمعا شوید.

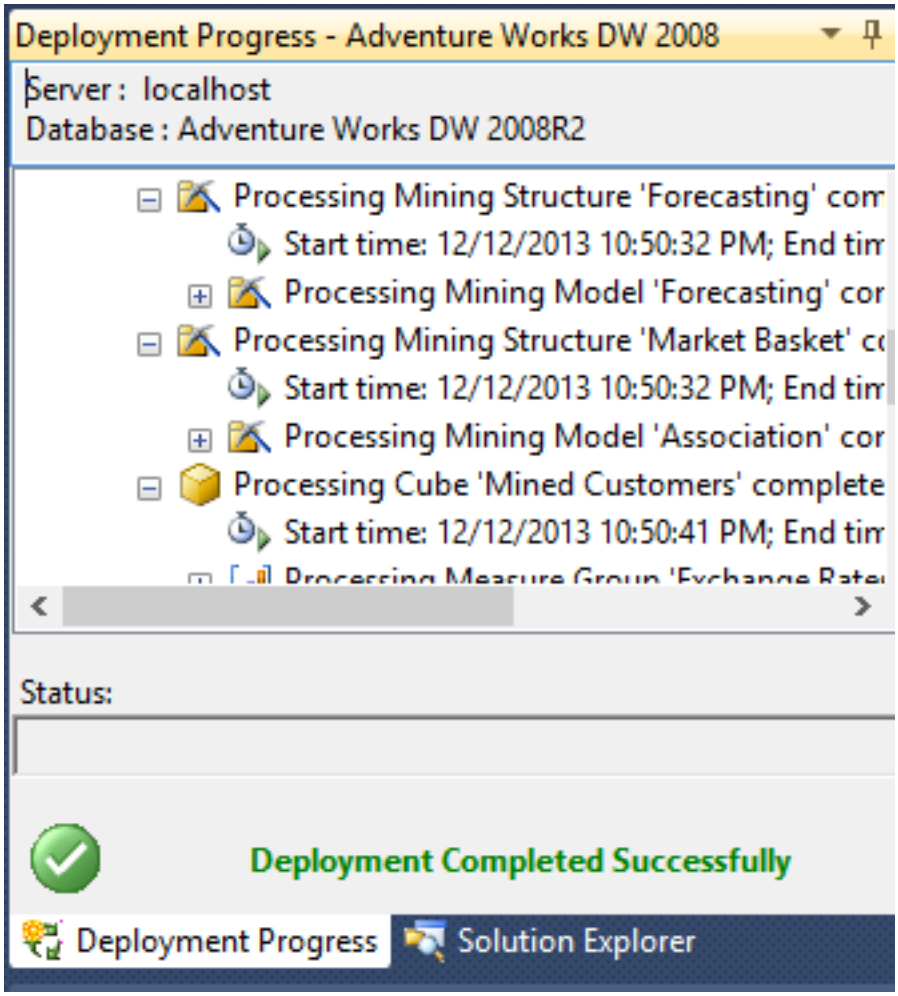


6. سپس دو بار دکمه‌ی OK را در دوصفحه کلیک کنید. (بعد از این مراحل شما آماده‌ی Deploy کردن می‌باشد)

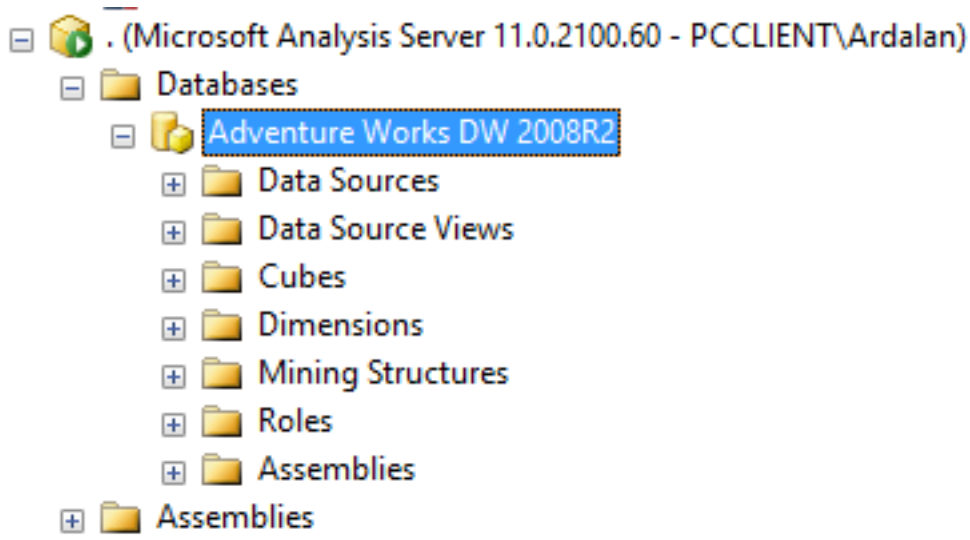
7. در ابتدا پروژه را Build نمایید (CTRl + Shift + B) و اطمینان حاصل کنید که Build با موفقیت انجام می‌شود.

8. در انتها برروی نام پروژه کلیک راست نمایید و گزینه‌ی Deploy را انتخاب نمایید. فرایند Deploy کردن می‌تواند کمی زمان بر باشد بنابر این شکیبا باشید و در انتها پیام Deployment Completed Successfully را دریافت خواهید کرد.


Deployment Progress - Adventure Works DW 2008
Server : localhost
Database : Adventure Works DW 2008R2
Processing Cube 'Adventure Works'.
Start time 12/12/2013 10:50:32 PM
Processing Mining Structure 'Forecasting'.
Start time 12/12/2013 10:50:32 PM
Processing Mining Model 'Forecasting'.
Processing Mining Structure 'Market Basket'.
Start time 12/12/2013 10:50:32 PM
Processing Mining Model 'Association'.
Status:
Finished building processing schedule.
Deploying and Processing
Deployment Progress
Solution Explorer



9. حال به SQL Server Management Studio بروید و به سرویس SSAS کانکت شوید . در قسمت DataBase یک پایگاه داده با نام Adventure Works DW 2008R2 مشاهده خواهید کرد .



به شما تبریک می‌گویم اینک شما یک پایگاه داده‌ی Multidimensional را ساخته اید .

در مقاله‌ی بعدی توضیحاتی در خصوص BIMS (Business Intelligence Management Studio) خواهم داد و همچنین اولین MDX Query را خواهیم نوشت.

عنوان: **استفاده از توابع Scalar بجای case**
نویسنده: فرهود جعفری
تاریخ: ۱۳۹۲/۱۱/۰۶ ۲۱:۲۵
آدرس: www.dotnettips.info
گروه‌ها: T-SQL, querying, SQL

گاهی از اوقات نیاز است در کوئری‌ها از بین چندین مقدار یکی انتخاب و بجای مقدار اصلی، رشته یا عبارتی جایگزین، نوشته شود. پر استفاده‌ترین راه حل پیشنهادی، استفاده از عبارت case در داخل کوئری هست که بر اساس موارد ممکن، عبارتهای برگشتی نوشته می‌شود. این راه حل خوبی به نظر می‌رسد اما اگر تعداد گزینه‌ها زیاد شود باعث شلوغ شدن متن کوئری و اشکال در بازبینی و نگهداری آن خواهد شد.
یک راه حل دیگر استفاده از توابع نوع Scalar می‌باشد؛ به این صورت که میتوان مقدار استخراج شده از جدول را به تابع تعریف شده فرستاد و در ازاء، مقدار بازگشتی مناسبی را در خروجی مشاهده کرد. حال به یک مثال توجه کنید:

```
Select Case Gen when 0 then 'مرد' when 1 then 'زن' end As Gen From Table
```

اکنون استفاده از تابع:

```
CREATE FUNCTION fcGenName
(
@Gen tinyint
)
RETURNS nvarchar(20)
AS
BEGIN
-- Declare the return variable here
DECLARE @gen nvarchar(20)

-- Add the T-SQL statements to compute the return value here
set @gen = (SELECT case @Gen when 0 then 'مرد' when 1 then 'زن' end as d)

-- Return the result of the function
RETURN @gen

END
```

و فراخوانی تابع در متن کوئری :

```
Select fcGenName(Gen) From Table
```

نظرات خوانندگان

نویسنده: م رحمانی
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۲:۲۲

سلام و تشکر
سؤال: این ارجاع به یک تابع تأثیر تو کارایی نداره؟

نویسنده: فرهود جعفری
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۴:۵۳

با سلام
ظاهراً در تعداد رکوردهای پایین مشکلی نداره اما در تعداد رکوردهای بالا احتمال کاهش سرعت اجرا دور از ذهن نیست. به هر حال من دقیق تست نکردم اما روی شیوه دیگه هم دارم کار می‌کنم که اون توابع برگشتی از نوع جدول هست که با این شیوه اساساً فرق داره

نویسنده: زاهدیان فرد
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۱:۴۴

استفاده از function خوبه مزیتش این که میشه جاهای مختلف استفاده کرد! ولی در تعداد رکورد پایین، چون در رکوردهای زیاد سرعت کوئری به شدت افت میکنه! روش اول بنظرم من بهتر

نویسنده: زاهدیان فرد
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۲:۱۰

در SQL server 2012 تابعی اضافه شده به اسم IIF که بجای

```
SELECT CASE @GEN WHEN 0 THEN 'Male' ELSE 'Woman' AS Gender
```

از این می‌توان استفاده کرد

```
SELECT IIF(Gen=0,'Male','Woman')
```

نویسنده: محمد سلیم آبادی
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۵:۳

در نسخه 2012 جهت سهولت در مهاجرت پایگاه داده‌های Access به SQL Server از توابع CHOOSE و IIF حمایت شده.

منتها تابع IIF چندان انعطاف پذیر نیست. مثلا اگر بخواهید به ازای چند حالت مشخص از داده‌های یک فیلد یک مقدار را برگردانید مجبورید چند تابع IIF تودرتو بنویسید. تودرتو بودن این تابع هم به 10 سطح محدود میشه.
اما CASE قابلیت‌ها و انعطاف پذیری بیشتری داره.
سوال میشه گاها کدام یک از این دو Performance یا کارایی بهتر دارد، در جواب میشه گفت هر دو برابر اند. در واقع IIF هنگام اجرا تبدیل به فرم CASE خواهد شد.
فرض کنید یک نظر سنجی تلوزیونی تنظیم کردیم که مردم از طریق پیامک نظر خودشان را به ما اعلام میکنند. شش گزینه هم داریم. برای انتخاب هر گزینه کافیست از اعداد 1 تا 6 استفاده کنیم. حال هنگام نمایش می‌خواهیم به جای اعداد مقدار متناظر ظاهر شود:

```
Use Tempdb
Go

CREATE TABLE [Sample] (value int);
INSERT INTO [Sample] VALUES (1),(2),(3),(4),(5),(6);
Go

--simple CASE Expression
SELECT value,
       CASE Value
        WHEN 1 THEN 'Very Bad'
        WHEN 2 THEN 'Bad'
        WHEN 3 THEN 'Not Bad'
        WHEN 4 THEN 'Good'
        WHEN 5 THEN 'Very Good'
        WHEN 6 THEN 'Excellent'
       ELSE NULL
END AS [Result]
FROM [Sample];

--CHOOSE Scalar Function
SELECT value,
       CHOOSE(value,'Very Bad','Bad','Not Bad','Good','Very Good','Excellent')
FROM [Sample];


--nested IIF Scalr Function
SELECT value,
IIF(value = 1, 'Very Bad',
  IIF(value = 2, 'Bad',
    IIF(value = 3, 'Not Bad',
      IIF(value = 4, 'Good',
        IIF(value = 5, 'Very Good', 'Excellent'
           )
        )
       )
     )
   )
FROM [Sample];
```

عنوان: **اندازه گیری کارآیی پرس و جوها با استفاده از SET STATISTICS TIME**
نویسنده: امید شکری
تاریخ: ۱۳۹۲/۱۲/۱۰ ۲۱:۵۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SQL, SQL Server Performance Tuning

یکی از وظایف اصلی مدیر و یا توسعه دهنده یک بانک اطلاعاتی، نوشتن کدهای T-SQL و اندازه‌گیری عملکرد آنها می‌باشد. ابزارهای مختلفی برای انجام این کار وجود دارد، چه آنهایی که در خود SQL Server بصورت محلی وجود دارند و چه آنهایی که توسط شرکت‌های ثالث ارائه می‌شوند. اما مسئله مهمی که باید در نظر بگیرید چگونگی نوشتن یک پرس و جو (Query) و اندازه گیری کارآیی آن می‌باشد و اینکه باید روی چه مواردی متمرکز شد. در اکثر مواقع گرفتن زمان اجرای یک پرس و جو تا اندازه‌ای خوب می‌باشد. یکی از مواردی که باید روی آن متمرکز شد منابع استفاده شده توسط سرور می‌باشد، درحالیکه زمان اجرای پرس وجو به پارامترهای دیگری همچون بار سرور نیز بستگی دارد. علاوه بر استفاده از پروفایلر و نقشه اجرای کوئری (Execution Plan) ، می‌توانید از **SET STATISTICS TIME** نیز استفاده نمایید.
SET STATISTICS TIME تنظیمی است که برای اندازه گیری منابع مورد نیاز اجرای یک پرس و جو به شما کمک می‌کند. SET STATISTICS TIME آمار مربوط به زمان تجزیه و تحلیل (Parse)، کامپایل و اجرای هر دستور در یک پرس و جو را نمایش می‌دهد. راه‌های مختلف اندکی برای مقایسه آماری دو پرس و جو و انتخاب بهترین آنها برای استفاده وجود دارند.
برای روشن کردن این تنظیم دو راه وجود دارد. ابتدا اینکه از دستور Set برای روشن و خاموش کردن استفاده نمایید و یا اینکه از طریق Query Analyzer اقدام به انجام این کار نمایید.

```
SET STATISTICS TIME ON
```

برای اینکه بتوانید آمارهای این تنظیمات را مشاهده کنید می‌بایست قبل از اجرای پرس و جو تنظیم مورد نظر را روشن نمایید. در نظر داشته باشید که با روشن کردن این تنظیم، برای تمامی پرس و جوهای مربوط به آن جلسه (Session) روشن خواهد ماند تا زمانیکه آنرا خاموش نمایید.

```
SELECT ProductID, StartDate, EndDate, StandardCost
FROM Production.ProductCostHistory
WHERE StandardCost < 500.00;
```

با اجرای دستورات بالا خروجی آن بصورت زیر می‌باشد:

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

(269 row(s) affected)

SQL Server Execution Times:
   CPU time = 1 ms,  elapsed time = 2 ms.
```

زمان صرف شده برای اجرای یک کوئری به دو بخش تقسیم می‌شود:
زمان کامپایل و تجزیه و تحلیل (parse and compile time) : زمانیکه یک کوئری را برای اجرا به SQL Server ارائه می‌دهید، SQL Server آنرا از نظر خطای نحوی بررسی می‌نماید و بهینه ساز یک نقشه بهینه را برای اجرا تولید می‌نماید. اگر به خروجی نگاه کنید، زمان پردازش (CPU time) و زمان سپری شده (elapsed time) را نشان می دهد. منظور از زمان پردازش زمان واقعی صرف شده روی پردازنده می‌باشد و زمان سپری شده اشاره به زمان تکمیل شدن عملیات کامپایل و تجزیه و تحلیل می‌باشد. تفاوت بین زمان پردازش و زمان سپری شده ممکن است زمان انتظار در صف برای گرفتن پردازنده و یا انتظار برای تکمیل عملیات IO باشد.

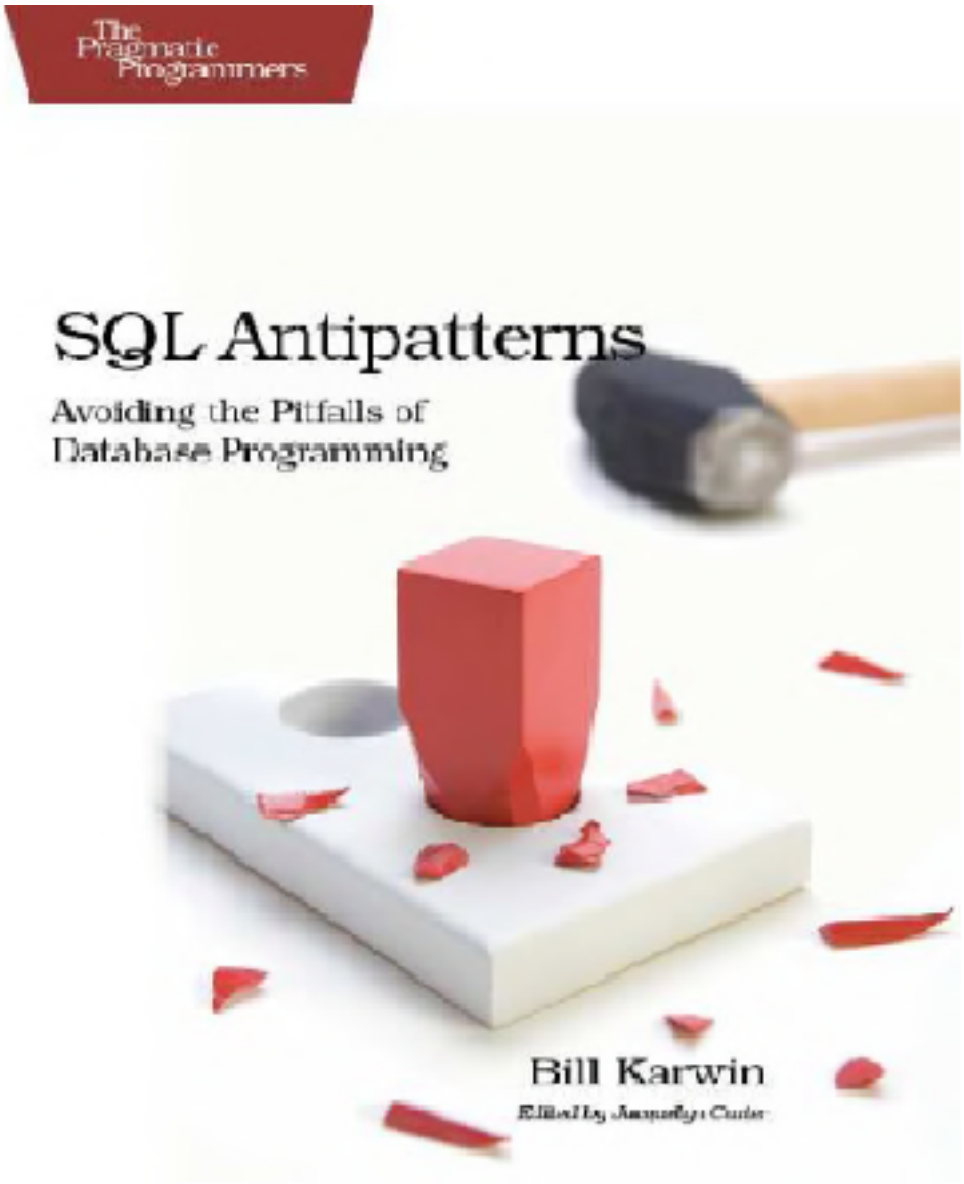
زمان اجرا (Execution Times) **:** این زمان اشاره به زمان سپری شده برای تکمیل اجرای نقشه کامپایل شده دارد. زمان پردازش اشاره به زمان واقعی صرف شده روی پردازنده دارد و زمان سپری شده نیز مجموع زمان صرف شده تا تکمیل اجرای دستور که شامل زمان انتظار برای تکمیل عملیات IO و زمان صرف شده برای انتقال خروجی به کلاینت می‌باشد، دارد. زمان پردازش می‌تواند به عنوان مبنایی برای اندازه‌گیری کارآیی مورد استفاده قرار بگیرد. این مقدار در اجراهای مختلف تفاوت چندانی با هم ندارند جز اینکه کوئری و یا داده‌ها تغییر نمایند. توجه نمایید که زمان براساس میلی ثانیه می‌باشد. زمان سپری شده نیز به فاکتورهایی مانند

بارگذاری روی سرور، بارگذاری IO، و پهنای باند بین سرور و کلاینت وابسته می‌باشد. بنابراین **همیشه زمان پردازش به عنوان مبنایی برای اندازه‌گیری کارایی استفاده می‌شود .**

در این بخش به بررسی **SET STATISTICS TIME** در SQL Server پرداختیم. در بخش بعدی به بررسی **SET STATISTICS IO** برای اندازه گیری کارایی پرس و جوها از نظر میزان استفاده IO خواهیم پرداخت.

عنوان: **SQL Antipattern #1**
نویسنده: سید مرتضی حسینی
تاریخ: ۱۳۹۳/۰۴/۲۹ ۱۳:۲۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SQL, Antipatterns

در این سلسله نوشتار قصد دارم ترجمه و خلاصه چندین فصل از کتاب ارزشمند (SQL Antipatterns: Avoiding the Pitfalls of Database Programming (Pragmatic Programmers که حاصل تلاش گروه IT موسسه هدایت فرهیختگان جوان می‌باشد، را ارائه نمایم.



بخش اول : Jaywalking

در این بخش در حال توسعه ویژگی نرم افزاری هستیم که در آن هرکاربر به عنوان یک کاربر اصلی برای یک محصول تخصیص داده می‌شود. در طراحی اصلی ما فقط اجازه می دهیم یک کاربر متعلق به هر محصول باشد، اما امکان چنین تقاضایی وجود دارد

که چند کاربر نیز به یک محصول اختصاص داده شوند.

در این صورت، اگر پایگاه داده را به نحوی تغییر دهیم که شناسه‌ی حساب کاربران را در لیستی که با کاما از یکدیگر جدا شده‌اند ذخیره نماییم، خیلی ساده‌تر به نظر می‌رسد نسبت به اینکه بصورت جداگانه آنها را ثبت نماییم.

برنامه نویسان معمولا برای جلوگیری از ایجاد جدول واسطی [1] که رابطه‌های چند به چند زیادی دارد از یک لیست که با کاما داده‌هایش از هم جدا شده‌اند، استفاده می‌کنند. بدین جهت اسم این بخش antipatten, jaywalking می‌باشد، زیرا jaywalking نیز عملیاتی است که از تقاطع جلوگیری می‌کند.

1.1 هدف: ذخیره کردن چندین صفت

طراحی کردن جدولی که ستون آن فقط یک مقدار دارد، بسیار ساده و آسان می‌باشد. شما می‌توانید نوع داده‌ایی که متعلق به ستون می‌باشد را انتخاب نمایید. مثلا از نوع (int,date...)

چگونه می‌توانیم مجموعه‌ایی از مقادیری که به یکدیگر مرتبط هستند را در یک ستون ذخیره نماییم ؟

در مثال ردیابی خطای پایگاه داده، ما یک محصول را به یک کاربر، با استفاده از ستونی که نوع آن integer است، مرتبط می‌نماییم. هر کاربر ممکن است محصولاتی داشته باشد و هر محصول به یک contact اشاره کند. بنابراین یک ارتباط چند به یک بین محصولات و کاربر برقرار است. برعکس این موضوع نیز صادق است؛ یعنی امکان دارد هر محصول متعلق به چندین کاربر باشد و یک ارتباط یک به چند ایجاد شود. در زیر جدول محصولات را به صورت عادی آورده شده است:

```
CREATE TABLE Products (
product_id SERIAL PRIMARY KEY,
product_name VARCHAR(1000),
account_id BIGINT UNSIGNED,
-- . . .
FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);

INSERT INTO Products (product_id, product_name, account_id)
VALUES (DEFAULT, 'Visual TurboBuilder' , 12);
```

2.1 Antipattern : قالب بندی لیست هایی که با کاما از یکدیگر جدا شده اند

برای اینکه تغییرات در پایگاه داده را به حداقل برسانید، می‌توانید نوع شناسه‌ی کاربر را از نوع varchar قرار دهید. در این صورت می‌توانید چندین شناسه‌ی کاربر که با کاما از یکدیگر جدا شده‌اند را در یک ستون ذخیره نمایید. پس در جدول محصولات، شناسه‌ی کاربران را از نوع varchar قرار می‌دهیم.

```
CREATE TABLE Products (
product_id SERIAL PRIMARY KEY,
product_name VARCHAR(1000),
account_id VARCHAR(100), -- comma-separated list
-- . . .
);

INSERT INTO Products (product_id, product_name, account_id)
VALUES (DEFAULT, 'Visual TurboBuilder' , '12,34' );
```

اکنون مشکلات کارایی و جامعیت داده‌ها را در این راه حل پیشنهادی بررسی می‌نماییم .

بدست آوردن محصولاتی برای یک کاربر خاص

بدلیل اینکه تمامی شناسه‌ی کاربران که بصورت کلید خارجی جدول Products می‌باشند به صورت رشته در یک فیلد ذخیره

شده‌اند و حالت ایندکس بودن آنها از دست رفته است، بدست آوردن محصولاتی برای یک کاربر خاص سخت می‌باشد. به عنوان مثال بدست آوردن محصولاتی که کاربری با شناسه‌ی 12 خریداری نموده به‌صورت زیر می‌باشد:

```
SELECT * FROM Products WHERE account_id REGEXP '[[:<:]]12[[:>:]]' ;
```

بدست آوردن کاربرانی که یک محصول خاص خریداری نموده اند

Join کردن account_id با Accounts table یعنی جدول مرجع نامناسب و غیر معمول می‌باشد. مساله مهمی که وجود دارد این است که زمانیکه بدین صورت شناسه‌ی کاربران را ذخیره می‌نماییم، ایندکس بودن آنها از بین می‌رود. کد زیر کاربرانی که محصولی با شناسه‌ی 123 خریداری کرده‌اند را محاسبه می‌کند.

```
SELECT * FROM Products AS p JOIN Accounts AS a
ON p.account_id REGEXP '[[:<:]]' || a.account_id || '[[:>:]]'
WHERE p.product_id = 123;
```

ایجاد توابع تجمیعی [2]

مبنای چنین توابعی از قبیل avg ,()count ,()sum() بدین صورت می‌باشد که بر روی گروهی از سطرها اعمال می‌شوند. با این حال با کمک ترفندهایی می‌توان برخی از این توابع را تولید نماییم هر چند برای همه آن‌ها این مساله صادق نمی‌باشد. اگر بخواهیم تعداد کاربران برای هر محصول را بدست بیاوریم ابتدا باید طول لیست شناسه‌ی کاربران را محاسبه کنیم، سپس کاما را از این لیست حذف کرده و طول جدید را از طول قبلی کم کرده و با یک جمع کنیم. نمونه کد آن در زیر آورده شده است:

```
SELECT product_id, LENGTH(account_id) - LENGTH(REPLACE(account_id, ',' , '' )) + 1
AS contacts_per_product
FROM Products;
```

ویرایش کاربرانی که یک محصول خاص خریداری نموده‌اند

ما به راحتی می‌توانیم یک شناسه‌ی جدید را به انتهای این رشته اضافه نماییم؛ فقط مرتب بودن آن را بهم میریزیم. در نمونه اسکریپت زیر گفته شده که در جدول محصولات برای محصولی با شناسه‌ی 123 بعد از کاما یک کاربر با شناسه‌ی 56 درج شود:

```
UPDATE Products
SET account_id = account_id || ',' || 56
WHERE product_id = 123;
```

برای حذف یک کاربر از لیست کافیست دو دستور sql را اجرا کرد: اولی برای fetch کردن شناسه‌ی کاربر از لیست و بعدی برای ذخیره کردن لیست ویرایش شده. بطور مثال تمامی افرادی که محصولی با شناسه‌ی 123 را خریداری کرده‌اند، از جدول محصولات انتخاب می‌کنیم. برای حذف کاربری با شناسه‌ی 34 از لیست کاربران، بر حسب کاما مقادیر لیست را در آرایه می‌ریزیم، شناسه‌ی مورد نظر را جستجو می‌کنیم و آن را حذف می‌کنیم. دوباره مقادیر درون آرایه را بصورت رشته درمیاوریم و جدول محصولات را با لیست جدید کاربران برای محصولی با شناسه‌ی 123 بروزرسانی می‌کنیم.

```
<?php
$stmt = $pdo->query(
"SELECT account_id FROM Products WHERE product_id = 123");
$row = $stmt->fetch();
$contact_list = $row['account_id' ];


// change list in PHP code
$value_to_remove = "34";
$contact_list = split(",", $contact_list);
$key_to_remove = array_search($value_to_remove, $contact_list);
unset($contact_list[$key_to_remove]);
$contact_list = join(",", $contact_list);
$stmt = $pdo->prepare(
"UPDATE Products SET account_id = ?
WHERE product_id = 123");
$stmt->execute(array($contact_list));
```

اعتبارسنجی شناسه‌ی محصولات

به دلیل آنکه نوع فیلد account_id،varchar می‌باشد احتمال این وجود دارد داده‌ای نامعتبر وارد نماییم چون پایگاه داده‌ها هیچ عکس العملی یا خطایی را نشان نمی‌دهد، فقط از لحاظ معنایی دچار مشکل می‌شویم. در نمونه زیر banana یک داده‌ی غیر معتبر می‌باشد و ارتباطی با شناسه‌ی کاربران ندارد.

```
INSERT INTO Products (product_id, product_name, account_id)
VALUES (DEFAULT, 'Visual TurboBuilder' , '12,34,banana' );
```

انتخاب کردن کاراکتر جداکننده

نکته قابل توجه این است که کاراکتری که بعنوان جداکننده در نظر می‌گیریم باید در هیچکدام از داده‌های ورودی ما امکان بودنش وجود نداشته باشد .

محدودیت طول لیست

در زمان طراحی جدول، برای هر یک از فیلدها باید حداکثر طولی را قرار دهیم. به عنوان نمونه ما اگر varchar(30) در نظر بگیریم بسته به تعداد کاراکترهایی که داده‌های ورودی ما دارند می‌توانیم جدول را پرنماییم. اسکریپت زیر شناسه‌ی کاربرانی که محصولی با شناسه‌ی 123 را خریده‌اند، ویرایش می‌کند. با این تفاوت که با توجه به محدودیت لیست، در نمونه‌ی اولی شناسه‌ی کاربران دو کاراکتری بوده و 10داده بعنوان ورودی پذیرفته است در حالیکه در نمونه‌ی دوم بخاطر اینکه شناسه‌ی کاربران شش کاراکتری می‌باشد فقط 4 شناسه می‌توانیم وارد نماییم.

```
UPDATE Products SET account_id = '10,14,18,22,26,30,34,38,42,46'
WHERE product_id = 123;
```

```
UPDATE Products SET account_id = '101418,222630,343842,467790'
WHERE product_id = 123;
```

3.1 موارد تشخیص این Antipattern:

سؤالات زیر نشان می‌دهند که Jaywalking antipattern مورد استفاده قرار گرفته است:

• حداکثر پذیرش این لیست برای داده ورودی چه میزان است؟

• چه کاراکتری هرگز در داده‌های ورودی این لیست ظاهر نمی‌شود؟ **4.1 مواردی که استفاده از این Antipattern مجاز است:**

دی نرمال کردن کارایی را افزایش می‌دهد. ذخیره کردن شناسه‌ها در یک لیست که با کاما از یکدیگر جدا شده‌اند نمونه بارزی از دی نرمال کردن می‌باشد. ما زمانی می‌توانیم از این مدل استفاده نماییم که به جدا کردن شناسه‌ها از هم نیاز نداشته باشیم. به همین ترتیب، اگر در برنامه، شما یک لیست را از یک منبع دیگر با این فرمت دریافت نمایید، به سادگی آن را در پایگاه داده خود به همان فرمت بصورت کامل می‌توانید ذخیره و بازیابی نمایید و احتیاجی به مقادیر جداگانه ندارید. درهنگام استفاده از پایگاه داده‌های دی‌نرمال دقت کنید. برای شروع از پایگاه داده نرمال استفاده کنید زیرا به کدهای برنامه شما امکان انعطاف بیشتری می‌دهد و کمک می‌کند تا پایگاه داده‌ها تمامیت داده(Data integrity) را حفظ کند. **5.1 راه حل: استفاده کردن از جدول واسط**

در این روش شناسه‌ی کاربران را در جدول جداگانهایی که هر کدام از آنها یک سطر را به خود اختصاص داده اند، ذخیره می‌نماییم.

```
CREATE TABLE Contacts ( product_id BIGINT UNSIGNED NOT NULL,
account_id BIGINT UNSIGNED NOT NULL,
PRIMARY KEY (product_id, account_id),
FOREIGN KEY (product_id) REFERENCES Products(product_id),
FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

جدول Contacts یک جدول رابطه ایی بین جداول Products,Accounts

بدست آوردن محصولات برای کاربران و موارد مربوط به آن

ما براحتی می‌توانیم تمامی محصولاتی که مختص به یک کاربر هستند را بدست آوریم. در این شیوه خاصیت ایندکس بودن شناسه‌ی کاربران حفظ می‌شود به همین دلیل queryهای آن برای خواندن و بهینه کردن راحت‌تر می‌باشند. در این روش به کاراکتری برای جدا کردن ورودی‌ها از یکدیگر نیاز نداریم چون هر کدام از آنها در یک سطر جداگانه ثبت می‌شوند. برای ویرایش کردن کاربرانی که یک محصول را خریداری نموده اند، کافیست یک سطر از جدول واسط را اضافه یا حذف نماییم. درنمونه کد زیر، ابتدا در جدول Contacts کاربری با شناسه‌ی 34 که محصولی با شناسه‌ی 456 را خریداری کرده، درج شده است و در خط بعد عملیات حذف با شرط آنکه شناسه‌ی کاربر و محصول به ترتیب 34،456 باشد روی جدول Contacts اعمال شده است.

```
INSERT INTO Contacts (product_id, account_id) VALUES (456, 34);

DELETE FROM Contacts WHERE product_id = 456 AND account_id = 34;
```

ایجاد توابع تجمیعی

به عنوان نمونه در مثال زیر براحتی ما می‌توانیم تعداد محصولات در هر حساب کاربری را بدست آوریم:

```
SELECT account_id, COUNT(*) AS products_per_account
FROM Contacts
GROUP BY account_id;
```

اعتبارسنجی شناسه محصولات

از آنجاییکه مقادیری که در جدول قرار دارند کلید خارجی می‌باشند میتوان صحت اعتبار آنها را بررسی نمود. بعنوان مثال Contacts.account_id به Account.account_id اشاره می‌کند. در ضمن برای هر فیلد نوع آن را می‌توان مشخص کرد تا فقط همان نوع داده را بپذیرد.

محدودیت طول لیست

نسبت به روش قبلی تقریبا در این حالت محدودیتی برای تعداد کاراکترهای ورودی نداریم.

مزیت‌های دیگر استفاده از جدول واسط

کارایی روش دوم بهتر از حالت قبلی می‌باشد چون ایندکس بودن شناسه‌ها حفظ شده است. همچنین براحتی میتوانیم فیلدی را به این جدول اضافه نماییم مثلا (time, date...)

Intersection Table [1]

Aggregate [2]

نظرات خوانندگان

نویسنده: م منفرد
تاریخ: ۱۳۹۳/۰۴/۳۰ ۰:۴۸

سلام
منظورتان از گروه it موسسه هدایت فرهیختگان چه بود؟ ایشان کتاب را ترجمه کردند؟ چگونه می‌توان آن را تهیه کرد؟
با تشکر

نویسنده: فرید
تاریخ: ۱۳۹۳/۰۶/۲۶ ۲۰:۱۸

سلام؛ به نظر میرسه این خصیصه مربوط به mysql است. من نتوانستم آن‌را در sqlserver اجرا کنم.

عنوان: **SQL Antipattern #2**
نویسنده: سید مرتضی حسینی
تاریخ: ۱۳۹۳/۰۴/۳۰ ۱۳:۲۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SQL, Antipatterns

بخش دوم : Naive Trees

فرض کنید یک وب سایت حرفه‌ای خبری یا علمی-پژوهشی داریم که قابلیت دریافت نظرات کاربران را در مورد هر مطلب مندرج در سایت یا نظرات داده شده در مورد آن مطالب را دارا می‌باشد. یعنی هر کاربر علاوه بر توانایی اظهار نظر در مورد یک خبر یا مطلب باید بتواند پاسخ نظرات کاربران دیگر را نیز بدهد. اولین راه حلی که برای طراحی این مطلب در پایگاه داده به ذهن ما می‌رسد، ایجاد یک زنجیره با استفاده از کد sql زیر می‌باشد:

```
CREATE TABLE Comments (
comment_idSERIAL PRIMARY KEY,
parent_idBIGINT UNSIGNED,
comment TEXT NOT NULL,
FOREIGN KEY (parent_id) REFERENCES Comments(comment_id)
);
```

البته همان طور که پیداست بازیابی زنجیره‌ای از پاسخ‌ها در یک پرس‌وجوی sql کار سختی است. این نخ‌ها معمولا عمق نامحدودی دارند و برای به دست آوردن تمام نخ‌های یک زنجیره باید پرس‌وجوهای زیادی را اجرا نمود.

ایده‌ی دیگر می‌تواند بازیابی تمام نظرها و ذخیره‌ی آن‌ها در حافظه‌ی برنامه به صورت درخت باشد. ولی این روش برای ذخیره هزاران نظری که ممکن است در سایت ثبت شود و علاوه بر آن مقالات جدیدی که اضافه می‌شوند، تقریبا غیرعملی است.

1.2 هدف: ذخیره و ایجاد پرس‌وجو در سلسله‌مراتب

وجود سلسله مراتب بین داده‌ها امری عادی محسوب می‌گردد. در ساختار داده‌ای درختی هر ورودی یک گره محسوب می‌گردد. یک گره ممکن است تعدادی فرزند و یک پدر داشته باشد. گره اول پدر ندارد، ریشه و گره فرزند که فرزند ندارد، برگ و گره‌ای دیگر، گره‌های غیربرگ نامیده می‌شوند.

مثال‌هایی که از ساختار درختی داده‌ها وجود دارد شامل موارد زیر است:

Organizational chart : در این ساختار برای مثال در ارتباط بین کارمندان و مدیر، هر کارمند یک مدیر دارد که نشان‌دهنده‌ی پدر یک کارمند در ساختار درختی است. هر مدیر هم یک کارمند محسوب می‌گردد.

Threaded discussion : در این ساختار برای مثال در سیستم نظردهی و پاسخ‌ها، ممکن است زنجیره‌ای از نظرات در پاسخ به نظرات دیگر استفاده گردد. در درخت، فرزندان یک گره‌ی نظر، پاسخ‌های آن گره هستند.

در این فصل ما از مثال ساختار دوم برای نشان دادن Antipattern و راه حل آن بهره می‌گیریم.

2.2 Antipattern : همیشه مبتنی بر یکی از والدین

راه حل ابتدایی و ناکارآمد

اضافه نمودن ستون parent_id . این ستون، به ستون نظر در همان جدول ارجاع داده می‌شود و شما می‌توانید برای اجرای این رابطه از قید کلید خارجی استفاده نمایید. پرس‌وجویی که برای ساخت مثالی که ما در این بحث از آن استفاده می‌کنیم در ادامه آمده است:

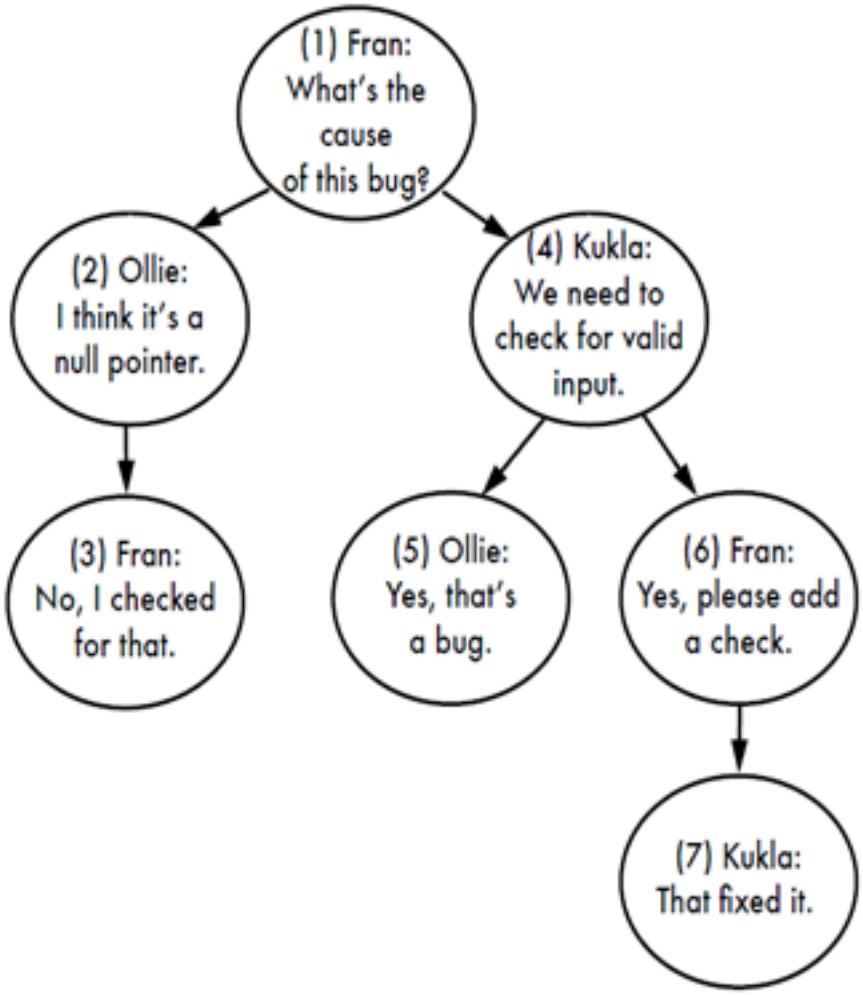
```
CREATE TABLE Comments (  comment_idSERIAL PRIMARY KEY,
parent_idBIGINT UNSIGNED,
bug_idBIGINT UNSIGNED NOT NULL,
author BIGINT UNSIGNED NOT NULL,
comment_dateDATETIME NOT NULL,
comment TEXT NOT NULL,
FOREIGN KEY (parent_id)REFERENCES Comments(comment_id),
FOREIGN KEY (bug_id)         REFERENCES Bugs(bug_id),
FOREIGN KEY(author)          REFERENCES Accounts(account_id)
);
```

مثالی از پرس‌وجوی فوق را می‌توانید در زیر ببینید:

comment_id	parent_id	author	comment
1	NULL	Fran	What's the cause of this bug?
2	1	Ollie	I think it's a null pointer.
3	2	Fran	No, I checked for that.
4	1	Kukla	We need to check for invalid input.
5	4	Ollie	Yes, that's a bug.
6	4	Fran	Yes, please add a check.
7	6	Kukla	That fixed it.

لیست مجاورت :

لیست مجاورت خود می‌تواند به عنوان یک antipattern در نظر گرفته شود. البته این مطلب از آنجایی نشأت می‌گیرد که این روش توسط بسیاری از توسعه‌دهندگان مورد استفاده قرار می‌گیرد ولی نتوانسته است به عنوان راه حل برای معمول‌ترین وظیفه‌ی خود، یعنی ایجاد پرس‌وجو بر روی کلیه فرزندان، باشد.



• با استفاده از پرس‌وجوی زیر می‌توان فرزند بلافاصله‌ی یک "نظر" را برگرداند:

```
SELECT c1.*, c2.*
FROM Comments c1 LEFT OUTER JOIN Comments c2
ON c2.parent_id = c1.comment_id;
```

ضعف پرس‌وجوی فوق این است که فقط می‌تواند دو سطح از درخت را برای شما برگرداند. در حالیکه خاصیت درخت این است که شما را قادر می‌سازد بدون هیچ گونه محدودیتی فرزندان و نوه‌های متعدد (سطوح بی‌شمار) برای درخت خود تعریف کنید.
بنابراین به ازای هر سطح اضافه باید یک join به پرس‌جوی خود اضافه نمایید. برای مثال اگر پرس‌وجوی زیر می‌تواند درختی با چهار سطح برای شما برگرداند ولی نه بیش از آن:

```
SELECT c1.*, c2.*, c3.*, c4.*
FROM Comments c1                        -- 1st level
LEFT OUTER JOIN Comments c2
ON c2.parent_id = c1.comment_id  -- 2nd level
LEFT OUTER JOIN Comments c3
ON c3.parent_id = c2.comment_id  -- 3rd level
LEFT OUTER JOIN Comments c4
ON c4.parent_id = c3.comment_id; -- 4th level
```

این پرس‌وجو به این دلیل که با اضافه شدن ستون‌های دیگر، نوه‌ها را سطوح عمیق‌تری برمی‌گرداند، پرس‌وجوی مناسبی نیست.

در واقع استفاده از توابع تجمیعی ، مانند COUNT() مشکل می‌شود.

راه دیگر برای به دست آوردن ساختار یک زیردرخت از لیست مجاورت برای یک برنامه، این است که سطرهای مورد نظر خود را از مجموعه بازیابی نموده و سلسه‌مراتب مورد نظر را در حافظه بازیابی نماییم و از آن به عنوان درخت استفاده نماییم:

```
SELECT * FROM Comments WHERE bug_id = 1234;
```

نگهداری کردن یک درخت با استفاده از لیست مجاورت

البته برخی از عملکردها با لیست مجاورت به خوبی انجام می‌گیرد. برای مثال اضافه نمودن یک گره (نظر)، مکان‌یابی مجدد برای یک گره یا یک زیردرخت .

```
INSERT INTO Comments (bug_id, parent_id, author, comment)
VALUES (1234, 7, 'Kukla' , 'Thanks!' );
```

بازیابی دوباره مکان یک نود یا یک زیردرخت نیز آسان است:

```
UPDATE Comments SET parent_id = 3 WHERE comment_id = 6;
```

با این حال حذف یک گره از یک درخت در این روش پیچیده است. اگر بخواهیم یک زیردرخت را حذف کنید باید چندین پرس‌وجو برای پیدا کردن تمام نوه‌ها بنویسیم و سپس حذف نوه‌ها را از پایین‌ترین سطح شروع کرده و تا جایی که قید کلید خارجی برقرار شود ادامه دهیم. البته می‌توان از کلید خارجی با تنظیم ON DELETE CASCADE استفاده کرد تا این کارها به طور خودکار انجام گیرد.
حال اگر بخواهیم یک نود غیر برگ را حذف کرده یا فرزندان آن را در درخت جابجا کنیم، ابتدا باید parent_id فرزندان آن نود را تغییر داده و سپس نود مورد نظر را حذف می‌کنیم:

```
SELECT parent_id FROM Comments WHERE comment_id = 6; -- returns 4
UPDATE Comments SET parent_id = 4 WHERE parent_id = 6;
DELETE FROM Comments WHERE comment_id = 6;
```

3.2 موارد تشخیص این Antipattern:

سؤالات زیر نشان می‌دهند که Naive Trees antipattern مورد استفاده قرار گرفته است:
چه تعداد سطح برای پشتیبانی در درخت نیاز خواهیم داشت؟

من همیشه از کار با کدی که ساختار داده‌ی درختی را مدیریت می‌کند، می‌ترسم

من باید اسکریپتی را به طور دوره‌ای اجرا نمایم تا سطرهای یتیم موجود در درخت را حذف کند.

4.2 مواردی که استفاده از این Antipattern مجاز است:

قدرت لیست مجاورت در بازیابی پدر یا فرزند مستقیم یک نود می‌باشد. قرار دادن یک سطر هم در لیست مجاورت کار ساده‌ای است. اگر این عملیات، تمام آن چیزی است که برای انجام کارتان مورد نیاز شما است، بنابراین استفاده از لیست مجاورت می‌تواند مناسب باشد.
برخی از برندهای RDBMS از افزونه‌هایی پشتیبانی می‌کنند که قابلیت ذخیره‌ی سلسله مراتب را در لیست مجاورت ممکن می‌سازد. مثلا SQL-99، پرس‌وجوی بازگشتی را تعریف می‌کند که مثال آن در ادامه آمده است:

```
WITH CommentTree (comment_id, bug_id, parent_id, author, comment, depth)
AS (
SELECT *, 0 AS depth FROM Comments
WHERE parent_id IS NULL
UNION ALL
SELECT c.*, ct.depth+1 AS depth FROM CommentTreect
JOIN Comments c ON (ct.comment_id = c.parent_id)
)
SELECT * FROM CommentTree WHERE bug_id = 1234;
```

Oracle 9i.نیز از پرس‌وجوی بازگشتی پشتیبانی می‌کنند PostgreSQL 8.4 و Microsoft SQL Server 2005، Oracle 11g، IBM DB2 و 10g از عبارت WITH استفاده می‌کنند، ولی نه برای پرس‌وجوهای بازگشتی. در عوض می‌توانید از پرس‌وجوی زیر برای ایجاد پرس‌وجوی بازگشتی استفاده نمایید:

```
SELECT * FROM Comments
START WITH comment_id = 9876
CONNECT BY PRIOR parent_id = comment_id;
```

5.2 راه حل: استفاده از مدل‌های درختی دیگر

جایگزین‌های دیگری برای ذخیره‌سازی داده‌های سلسله مراتبی وجود دارد. البته برخی از این راه حل‌ها ممکن است در لحظه‌ی اول پیچیدتر از لیست مجاورت به نظر آیند، ولی برخی از عملیات درخت که در لیست مجاورت بسیار سخت یا ناکارآمد است، را آسان‌تر می‌کنند. **شمارش مسیر :**

مشکل پرهزینه بودن بازیابی نیاکان یک گره که در روش لیست مجاورت وجود داشت در روش شمارش مسیر به این ترتیب حل شده است: اضافه نمودن یک صفت به هر گره که رشته‌ای از نیکان آن صفت در آن ذخیره شده است.

در جدول Comments به جای استفاده از parent_id، یک ستون به نام path که توع آن varchar است تعریف شده است. رشته‌ای که در این ستون تعریف شده است، ترتیبی از فرزندان این سطر از بالا به پایین درخت است. مانند مسیری که در سیستم عامل UNIX، برای نشان دادن مسیر در سیستم فایل استفاده شده است. شما می‌توانید از / به عنوان کاراکتر جداکننده استفاده نمایید. دقت کنید برای درست کار کردن پرس‌وجوها حتما در آخر مسیر هم این کاراکتر را قرار دهید. پرس‌وجوی تشکیل چنین درختی به شکل زیر است:

```
CREATE TABLE Comments ( comment_id SERIAL PRIMARY KEY,
path VARCHAR(1000),
bug_id BIGINT UNSIGNED NOT NULL,
author BIGINT UNSIGNED NOT NULL,
comment_date DATETIME NOT NULL,
comment TEXT NOT NULL,
FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
FOREIGN KEY (author) REFERENCES Accounts(account_id)
```

comment_id	path	author	comment
1	1/	Fran	What's the cause of this bug?
2	1/2/	Ollie	I think it's a null pointer.
3	1/2/3/	Fran	No, I checked for that.
4	1/4/	Kukla	We need to check for invalid input.
5	1/4/5/	Ollie	Yes, that's a bug.
6	1/4/6/	Fran	Yes, please add a check.
7	1/4/6/7/	Kukla	That fixed it.

در این روش، هر گره مسیری دارد که شماره خود آن گره هم در آنتهای آن مسیر قرار دارد. این به دلیل درست جواب دادن پرس‌وجوهای ایجاد شده است.

می‌توان نیاکان را با مقایسه‌ی مسیر سطر کنونی با مسیر سطر دیگر به دست آورد. برای مثال برای یافتن نیاکان گره (نظر) شماره‌ی 7 که مسیر آن 7/6/4/1/ می‌باشد، می‌توان چنین نوشت:

```
SELECT * FROM Comments AS c
WHERE '1/4/6/7/' LIKE c.path || '%' ;
```

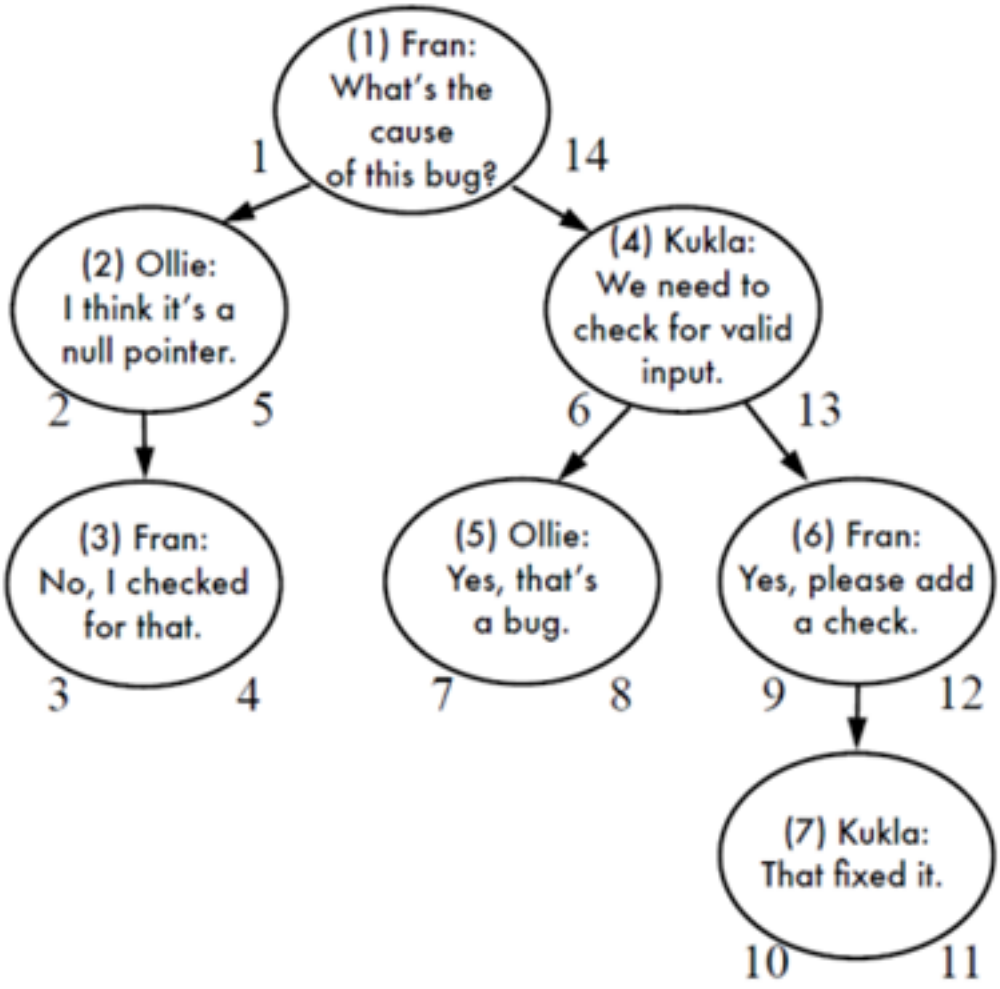
این پرس‌وجو الگوهایی را می‌یابد که از مسیرهای 1/4/6/%، 1/4/1% و 1% نشأت می‌گیرد.
همچنین فرزندان (نوه‌های) یک گره، مثلا گره‌ی 4 را که مسیرش 1/4/ است را می‌توان با پرس‌وجوی زیر یافت:

```
SELECT * FROM Comments AS c
WHERE c.path LIKE '1/4/' || '%' ;
```

الگوی 1/4/% با مسیرهای 1/5/4/، 1/6/4/ و 1/6/7/4/ تطابق می‌یابد.
همچنین می‌توان پرس‌وجوهای دیگری را نیز در این مسیر به سادگی انجام داد؛ مانند محاسبه‌ی مجموع هزینه‌ی گره‌ها در یک زیردرخت یا شمارش تعداد گره‌ها.
اضافه نمودن یک گره هم مانند ساختن خود مدل است. می‌توان یک گره‌ی غیر برگ را بدون نیاز به اصلاح هیچ سطری اضافه نمود. برای این کار مسیر را را از گره‌ی پدر کپی کرده و در انتها شماره‌ی خود گره را به آن اضافه می‌کنیم.
از مشکلات این روش می‌توان به عدم توانایی پایگاه داده‌ها در تحمیل این نکته که مسیر یک گره درست ایجاد شده است و یا تضمین وجود گره‌ای در مسیری خاص، اشاره نمود. همچنین نگهداری رشته‌ی مسیر یک گره مبتنی بر کد برنامه است و اعتبارسنجی آن کاری هزینه‌بر است. این رشته اندازه‌ای محدود دارد و درخت‌هایی با عمق نامحدود را پشتیبانی نمی‌کند.

مجموعه‌های تودرتو :

مجموعه‌های تودرتو، اطلاعات را با هر گره‌ای که مربوط به مجموعه‌ای از نوه‌هایش است، به جای این که تنها مربوط به یک فرزند بلافصلش باشد، ذخیره می‌کنند.



این اطلاعات می‌توانند به وسیله‌ی هر گره‌ای که در درخت با دو شماره‌ی nsleft و nsright ذخیره شده، نمایش داده شوند:

```
CREATE TABLE Comments ( comment_id SERIAL PRIMARY KEY,
nsleft INTEGER NOT NULL,
nsright INTEGER NOT NULL,
bug_id BIGINT UNSIGNED NOT NULL,
author BIGINT UNSIGNED NOT NULL,
comment_date DATETIME NOT NULL,
comment TEXT NOT NULL,
FOREIGN KEY (bug_id) REFERENCES Bugs (bug_id),
FOREIGN KEY (author) REFERENCES Accounts(account_id)
);
```

comment_id	nsleft	nsright	author	comment
1	1	14	Fran	What's the cause of this bug?
2	2	5	Ollie	I think it's a null pointer.
3	3	4	Fran	No, I checked for that.
4	6	13	Kukla	We need to check for invalid input.
5	7	8	Ollie	Yes, that's a bug.
6	9	12	Fran	Yes, please add a check.
7	10	11	Kukla	That fixed it.

شماره‌ی سمت چپ یک گره از تمام شماره‌های سمت چپ فرزندان آن گره کوچک‌تر و شماره‌ی سمت راست آن گره از تمام شماره‌های سمت راست آن گره بزرگ‌تر است. این شماره‌ها هیچ ارتباطی به comment_id مربوط به آن گره ندارند.

یک راه حل ساده برای تخصیص این شماره‌ها به گره‌ها این است که از سمت چپ یک گره آغاز می‌کنیم و اولین شماره را اختصاص می‌دهیم و به همین به گره‌ای سمت چپ فرزندان می‌آییم و شماره‌ها را به صورت افزایشی به سمت چپ آن‌ها نیز اختصاص می‌دهیم. سپس در ادامه به سمت راست آخرین نود رفته و از آن جا به سمت بالا می‌آییم و به همین ترتیب به صورت بازگشتی تخصیص شماره‌ها را ادامه می‌دهیم.

با اختصتص شماره‌ها به هر گره، می‌توان از آن‌ها برای یافتن نیاکان و فرزندان آن گره بهره جست. برای مثال برای بازیابی گره‌ی 4 و فرزندان (نوه‌های) آن باید دنبال گره‌هایی باشیم که شماره‌های آن گره‌ها بین nsleft و nsright گره‌ی شماره4 باشد:

```
SELECT c2.* FROM Comments AS c1
JOIN Comments as c2
ON c2.nsleft BETWEEN c1.nsleft AND c1.nsright
WHERE c1.comment_id = 4;
```

همچنین می‌توان گره‌ی شماره‌ی 6 و نیاکان آن را با دنبال نمودن گره‌هایی به دست آورد که شماره‌های آن‌ها در محدوده‌ی شماره‌ی گره‌ی 6 باشد:

```
SELECT c2.*
FROM Comments AS c1
JOIN Comment AS c2
ON c1.nsleft BETWEEN c2.nsleft AND c2.nsright
WHERE c1.comment_id = 6;
```

یک مزیت مهم روش مجموعه‌ای تودرتو، این است که هنگامی که یک گره را حذف می‌کنیم، نوه‌های آن به طور مستقیم به عنوان فرزندان پدر گره‌ی حذف شده تلقی می‌شوند.
برخی از پرس‌وجوهایی که در روش لیست مجاورت ساده بودند، مانند بازیابی فرزند یا پدر بلافصل، در روش مجموعه‌های تودرتو پیچیده‌تر می‌باشند. برای مثال برای یافتن پدر بلافصل گره‌ی شماره‌ی 6 باید چنین نوشت:

```
SELECT parent.* FROM Comment AS c
JOIN Comment AS parent
ON c.nsleft BETWEEN parent.nsleft AND parent.nsright
LEFT OUTER JOIN Comment AS in_between
ON c.nsleft BETWEEN in_between.nsleft AND in_between.nsright
AND in_between.nsleft BETWEEN parent.nsleft AND parent.nsright
WHERE c.comment_id = 6
AND in_between.comment_id IS NULL;
```

دست‌کاری درخت، اضافه، حذف و جابجا نمودن گره‌ها در آن درروش مجموعه‌های تودرتو از مدل‌های دیگر پیچیده‌تر است.
هنگامی که یک گره‌ی جدید را اضافه می‌کنیم، باید تمام مقادیر چپ و راست بزرگ‌تر از مقدار سمت چپ گره‌ی جدید را مجددا محاسبه کنیم؛ که این شامل برادر سمت راست گره‌ی جدید، نیاکان آن و برادر سمت راست نیاکان آن می‌باشد. همچنین اگر گره‌ی جدید به عنوان گره‌ی غیربرگ اضافه شده باشد، شامل فرزندان آن هم می‌شود. برای مثال اگر بخواهیم گره‌ی جدیدی به گره‌ی 5 اضافه نماییم، باید چنین بنویسیم:

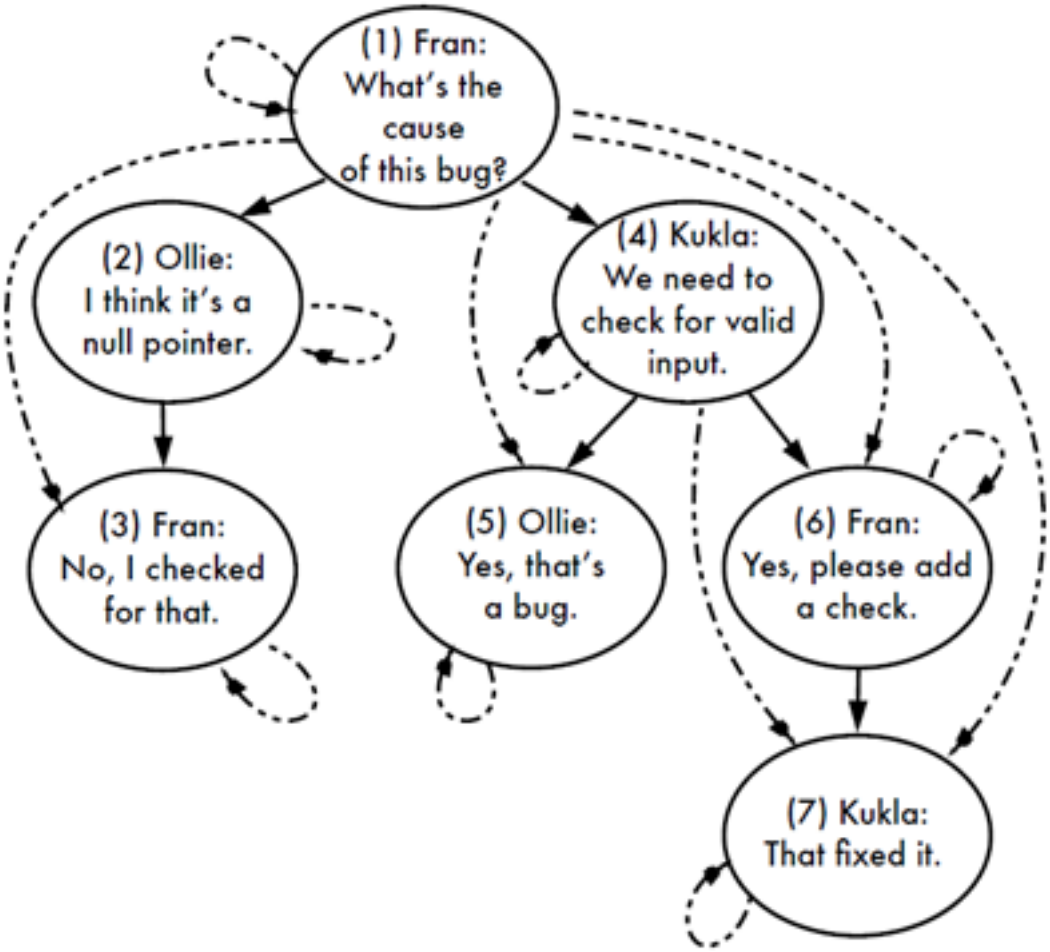
```
-- make space for NS values 8 and 9
UPDATE Comment
SET nsleft = CASE WHEN nsleft >= 8 THEN nsleft+2 ELSE nsleft END,
nsright = nsright+2
WHERE nsright >= 7;

-- create new child of comment #5, occupying NS values 8 and 9
INSERT INTO Comment (nsleft, nsright, author, comment)
VALUES (8, 9, 'Fran' , 'Me too!' );
```

تنها مزیت این روش نسبت به روش‌های قبلی ساده‌تر و سریع‌تر شدن ایجاد پرس‌وجوها برای پیدا کردن فرزندان یا پدران یک درخت است. اگر هدف استفاده از درخت شامل اضافه نمودن متعدد گره‌ها است، مجموعه‌های تودرتو انتخاب خوبی نیست.

Closure Table

راه حل closure table روشی دیگر برای ذخیره‌ی سلسه‌مراتبی است. این روش علاوه بر ارتباطات مستقیم پدر- فرزندی، تمام مسیرهای موجود در درخت را ذخیره می‌کند.



این روش علاوه بر داشتن یک جدول نظرها، یک جدول دیگر به نام TreePaths با دو ستون دارد که هر کدام از این ستون‌ها یک کلید خارجی به جدولComment هستند:

```
CREATE TABLE Comments ( comment_id SERIAL PRIMARY KEY,
bug_id BIGINT UNSIGNED NOT NULL,
author BIGINT UNSIGNED NOT NULL,
comment_date DATETIME NOT NULL,
comment TEXT NOT NULL,
FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
FOREIGN KEY (author) REFERENCES Accounts(account_id)
);
CREATE TABLE TreePaths (
ancestor BIGINT UNSIGNED NOT NULL,
descendant BIGINT UNSIGNED NOT NULL,
PRIMARY KEY(ancestor, descendant),
FOREIGN KEY (ancestor) REFERENCES Comments(comment_id),
FOREIGN KEY (descendant) REFERENCES Comments(comment_id)
);
```

ancestor	descendant	ancestor	descendant	ancestor	descendant
1	1	1	7	4	6
1	2	2	2	4	7
1	3	2	3	5	5
1	4	3	3	6	6
1	5	4	4	6	7
1	6	4	5	7	7

به جای استفاده از جدول Comments برای ذخیره‌ی اطلاعات مربوط به یک درخت از جدول TreePath استفاده می‌کنیم. به ازای هر یک جفت گره در این درخت یک سطر در جدول ذخیره می‌شود که ارتباط پدر فرزندی را نمایش می‌دهد و الزاما نباید این دو پدر فرزند بلافصل باشد. همچنین یک سطر هم به ازای ارتباط هر گره با خودش به جدول اضافه می‌گردد.

پرس‌وجوهای بازیابی نیاکان و فرزندان (گره‌ها) از طریق جدول TreePaths ساده‌تر از روش مجموعه‌های تودرتو است. مثلا برای بازیابی فرزندان (نوه‌های) گره‌ی شماره‌ی 4، سطرهایی که نیاکان آن‌ها 4 است را به دست می‌آوریم:

```
SELECT c.*  FROM Comments AS c
JOIN TreePaths AS t ON c.comment_id = t.descendant
WHERE t.ancestor = 4;
```

برای به دست آوردن نیاکان گره‌ی شماره‌ی 6، سطرهایی از جول TreePaths را به دست می‌آوریم که فرزندان آن‌ها 6 باشد:

```
SELECT c.*
FROM Comments AS c
JOIN TreePaths AS t ON c.comment_id = t.ancestor
WHERE t.descendant = 6;
```

برای اضافه کردن گره‌ی جدید، برای مثال به عنوان فرزند گره‌ی شماره‌ی 5، ابتدا سطری که به خود آن گره برمی‌گردد را اضافه می‌کنیم، سپس یک کپی از سطوری که در جدول TreePaths، به عنوان فرزندان (نوه‌های) گره‌ی شماره5 هستند (که شامل سطری که به خود گره‌ی 5 به عنوان فرزند اشاره می‌کند) به جدول اضافه نموده و فیلد descendant آن را با شماره‌ی گره‌ی جدید جایگزین می‌کنیم:

```
INSERT INTO TreePaths (ancestor, descendant) SELECT t.ancestor, 8
FROM TreePaths AS t
WHERE t.descendant = 5
UNION ALL
SELECT 8, 8;
```

در این جا می‌توان به اهمیت ارجاع یک گره به خودش به عنوان پدر (یا فرزند) پی برد.
برای حذف یک گره، مثلا گره‌ی شماره‌ی 7، تمام سطوری که فیلد descendant آن‌ها در جدول TreePaths برابر با 7 است حذف می‌کنیم:

```
DELETE FROM TreePaths WHERE descendant = 7;
```

برای حذف یک زیردرخت کامل، برای مثال گره‌ی شماره‌ی 4 و فرزندان (نوه‌های) آن، تمام سطوری که در جدول TreePaths دارای فیلد descendant با مقدار 4 هستند، حذف می‌کنیم. علاوه بر این باید نودهایی که به عنوان descendant به فیلد descendant گره‌ی 4، ارجاع داده می‌شوند نیز باید حذف گردد:

```
DELETE FROM TreePaths
WHERE descendant IN (SELECT descendant
FROM TreePaths
WHERE ancestor = 4);
```

دقت کنید وقتی گره‌ای را حذف می‌کنیم، بدان معنی نیست که خود گره (نظر) را حذف می‌کنیم. البته این برای مثال نظر و پاسخ آن مقداری عجیب است ولی در مثال کارمندان در چارت سازمانی امری معمول است. هنگامی که ارتباطات یک کاربر را تغییر می‌دهیم، از حذف در جدول TreePaths استفاده می‌کنیم و این قضیه که ارتباطات کارمندان در جدول جداگانه‌ای ذخیره شده است به ما انعطاف‌پذیری بیشتری می‌دهد.

برای جابجایی یک زیردرخت از مکانی به مکان دیگری در درخت، سطرهایی که ancestor گره‌ی بالایی زیردرخت را برمی‌گردانند و فرزندان آن گره را حذف می‌کنیم. برای مثال برای جابجایی گره‌ی شماره‌ی 6 به عنوان فرزند گره‌ی شماره‌ی 4 و قرار دادن آن به عنوان فرزند گره‌ی شماره‌ی 3، این چنین عمل می‌کنیم. فقط باید حواسمان جمع باشد سطری که گره‌ی شماره‌ی 6 به خودش ارجاع داده است را حذف نکنیم:

```
DELETE FROM TreePaths
WHERE descendant IN (SELECT descendant
                                        FROM TreePaths
                                        WHERE ancestor = 6)
AND ancestor IN (SELECT ancestor
                              FROM TreePaths
                              WHERE descendant = 6
                                  AND ancestor != descendant);
```

آن‌گاه این زیردرخت جدا شده را با اضافه کردن سطرهایی که با ancestor مکان جدید و descendant زیردرخت، منطبق هستند، به جدول اضافه می‌کنیم:

```
INSERT INTO TreePaths (ancestor, descendant)
SELECT supertree.ancestor, subtree.descendant
FROM TreePaths AS supertree
CROSS JOIN TreePaths AS subtree
WHERE supertree.descendant = 3
AND subtree.ancestor = 6;
```

روش Closure Table آسان‌تر از روش مجموعه‌های تودرتو است. هر دوی آن‌ها روش‌های سریع و آسانی برای ایجاد پرس‌وجو برای نیاکان و نوه‌ها دارند. ولی Closure Table برای نگهداری اطلاعات سلسله مراتب آسان‌تر است. در هر دو طراحی ایجاد پرس‌وجو در فرزندان و پدر بلافصل سرراست‌تر از روش‌ای لیست مجاورت و شمارش مسیر می‌باشد.
می‌توان عملکرد Closure Table را برای ایجاد پرس‌وجو روی فرزندان و پدر بلافصل را آسان‌تر نیز نمود. اگر فیلد path_length را به جدول TreePaths اضافه نماییم این کار انجام می‌شود. path_length گره‌ای که به خودش ارجاع می‌شود، صفر است. path_length فرزند بلافصل هر گره 1، path_length نوه‌ی آن 2 می‌باشد و به همین ترتیب path_lengthها را در هر سطر مقداردهی می‌کنیم. اکنون یا فتن فرزند گره‌ی شماره‌ی 4 آسان‌تر است:

```
SELECT *
FROM TreePaths
WHERE ancestor = 4 AND path_length = 1;
```

از کدام طراحی استفاده نماییم؟

در این جا این سؤال مطرح است که ما باید از کدام طراحی استفاده نماییم. در پاسخ به این سؤال باید گفت که هر کدام از این روش‌ها نقاط قوت و ضعفی دارند که ما باید نسبت به عملیاتی که می‌خواهیم انجام دهیم از این طراحی‌ها استفاده کنیم. جدولی که در ادامه آمده است، مقایسه‌ای است میان میزان سهولت اجرای این طراحی‌ها در استفاده از پرس‌وجوهای متفاوت.

Design	Tables	Query Child	Query Tree	Insert	Delete	Referential Integration
Adjacency List	1	Easy	Hard	Easy	Easy	Yes
Recursive Query	1	Easy	Easy	Easy	Easy	Yes
Path Enumeration	1	Easy	Easy	Easy	Easy	No
Nested Sets	1	Hard	Easy	Hard	Hard	No
Closure Table	2	Easy	Easy	Easy	Easy	Yes

لازم به ذکر است در اینجا ستون سوم (Query Child) به معنای پرس‌وجوهایی است که با فرزندان کار می‌کند و ستون چهارم (Query Tree) به معنای پرس‌وجوهایی است که با کل درخت کار می‌کنند، می‌باشد.

نظرات خوانندگان

نویسنده: علیرضا صبوری
تاریخ: ۱۳۹۳/۰۵/۰۵ ۱۴:۴۷

فکر میکنم عموما پرس‌وجوی بازگشتی اگر ساپورت بشه توسط دیتابیس بهترین روش همان لیست مجاورت هستش که مدیریت درخت رو برامون ساده میکنه و دیتابیس کنترل بشتری رو هر نود ما داره. البته به غیر از مواردی خاص... ممنون از مطلب مفیدتون ولی سوالی که دارم اینه از نظر Performance مقایسه ای انجام شده که آیا استفاده از لیست بازگشتی چقدر از نظر سرعت در بازیابی اطلاعات با سایر روش‌ها تفاوت داره ؟ مبنعی اگر سراغ دارید ممنون میشم معرفی کنین.